ICF Catalog Backup and Recovery: Catalog RecoveryPlus Update

Learn how to be prepared when catalogs break
Know the tools for debugging catalog errors
Explore case studies of common error situations

Mary Lovelace
Eileen McClintock

ibm.com/redbooks
First Edition (December 2006)

This edition applies to Mainstar Catalog RecoveryPlus (product number 5620-FGY).

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

Notices ...................................................................................... v
Trademarks ............................................................................... vi

Preface ........................................................................................ vii
The team that wrote this Redpaper ................................................ vii
Become a published author .......................................................... viii
Comments welcome .................................................................. viii

Chapter 1. Using Integrated Catalog Forward Recover Utility ............................. 1
1.1 ICFRU basics ......................................................................... 2
1.2 Recovering an ICF catalog using ICFRU .................................... 2
1.3 Post recovery actions .............................................................. 6

Chapter 2. Using Catalog RecoveryPlus ....................................................... 7
2.1 Introduction to CR+ facilities .................................................... 8
2.2 CR+ ISPF interface ............................................................... 9
2.3 SAF FACILITY class profiles ................................................ 14
2.4 BACKUP command ................................................................ 15
  2.4.1 Specifying BCSs to be backed up ..................................... 16
  2.4.2 Specifying VVDSs to be backed up .................................. 18
  2.4.3 Diagnosing BCS or VVDS integrity prior to backup .......... 19
  2.4.4 Restricting access to the BCS or volume during backup .... 20
  2.4.5 Creating multiple backup copies .................................... 20
  2.4.6 Alias processing with BACKUP BCS ............................. 22
  2.4.7 Bypassing index processing with BACKUP BCS ............. 22
  2.5 RECOVER command ......................................................... 23
    2.5.1 Restore versus forward recovery ................................... 23
    2.5.2 Specifying the BCS or VVDS to restore .......................... 24
    2.5.3 Explicit/implicit DEFINE of the BCS or VVDS to restore.. 24
    2.5.4 Locking versus unlocking the BCS ............................... 25
    2.5.5 Alias handling during BCS restore/recovery .................... 25
    2.5.6 Renaming the BCS during restore ................................. 25
    2.5.7 SMF data for forward recovery .................................... 26
    2.5.8 Printing BCS records during a RECOVER ....................... 29
    2.5.9 Simulation as a BCS or VVDS recovery testing tool ......... 30
    2.5.10 Resizing the VVDS during RECOVER processing ........... 31
  2.6 DIAGNOSE command ....................................................... 31
    2.6.1 DIAGNOSE ALIAS command ..................................... 32
    2.6.2 DIAGNOSE BCS-VVDS and VOLUME-BCS commands .. 34
  2.7 MERGE CAT command ...................................................... 36
    2.7.1 Alias handling .......................................................... 37
    2.7.2 Using the journal file .................................................. 37
    2.7.3 RESTART and BACKOUT capability ............................ 38
    2.7.4 SIMULATE capability ............................................... 38
    2.7.5 Catalog activity during MERGE CAT processing ............. 38
    2.7.6 Creating an alternate master catalog .............................. 39
  2.8 SUPERCLIP command ....................................................... 39
    2.8.1 Changing a volser .................................................... 39
    2.8.2 Caution when using SUPERCLIP ................................. 40
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:

- @server®
- Redbooks (logo) ™
- z/OS®
- AIX®
- DB2®
- DFMSdfp™
- DFSMSdss™
- DFSMSshsm™
- DFSMSrmm™
- DFSORT™
- IBM®
- MVS™
- Redbooks™
- RACF®
- S/360™
- S/390®
- TotalStorage®

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

ICF catalogs are essential system data sets. Even with the high availability storage subsystems and processors that are available today, there are still situations where you need to recover your catalogs. You should keep your catalogs healthy and be prepared for a recovery situation. Also, you must be sure that your recovery procedures do not have a big impact on your production environment. To minimize the recovery process, you should have a clear backup and recovery strategy and the proper utilities.

IBM® Integrated Catalog Forward Recovery Utility (ICFRU) and Mainstar Catalog RecoveryPlus (CR+) are two of the available tools you can use to recover your catalogs. ICFRU is a basic tool to help you in a forward recovery situation. It does not offer a wide range of features, but it is useful for catalog recovery. Mainstar Catalog RecoveryPlus offers a set of features to help you with your ICF catalog environment maintenance.

This IBM Redpaper explains how to use each of these products in a catalog recovery situation. It also offers storage administrators useful recommendations for implementing a catalog backup and recovery plan. It does not compare the two products but instead shows how each of them work. This paper also provides practical tests and examples that can help you with the different error scenarios that you might find in your daily production activities.

The team that wrote this Redpaper

This Redpaper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Mary Lovelace is a Consulting IT specialist at the International Technical Support Organization. She has more than 20 years of experience with IBM in large systems, storage, and storage networking product education, system engineering and consultancy, and systems support. She has written many IBM Redbooks™ about TotalStorage® Productivity Center and z/OS® storage products.

Eileen McClintock is a Catalog RecoveryPlus Product Support Specialist for Mainstar Software in Glendale, CA. She also works as an independent contractor, teaching z/OS courses for IBM. She has 30 years of experience in IBM mainframe operating systems and worked for IBM for 14 years. Her last assignment at IBM was large systems and storage support in the Western Area System Center. Her areas of expertise include ICF catalogs, VSAM, and SMP/E.

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

Robert Haimowitz
ITSO Raleigh

Mark Thomen
IBM San Jose

Elliot Hamilton
Dave Roese
Mainstar Software
Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll 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'll 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 Redpaper or other Redbooks in one of the following ways:

► Use the online Contact us review redbook form found at:
  ibm.com/redbooks

► Send your comments in an email to:
  redbook@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
Chapter 1. Using Integrated Catalog Forward Recover Utility

This chapter explains how to use the IBM Integrated Catalog Forward Recovery Utility (ICFRU), an ICF catalog recovery tool. With ICFRU, you can forward recover ICF catalogs using selected SMF records and an error-free backup of the ICF catalog being recovered. The backup must be created by the IDCAMS EXPORT command.

ICFRU (5798-DXQ) has been incorporated into the z/OS base and is documented in Appendix A of DFSMS Managing Catalogs, SC26-7409.
1.1 ICFRU basics

ICFRU is a catalog recovery tool that helps you recover a damaged catalog to a correct and current status. It supports all types of catalog entries that can exist in the basic catalog structure (BCS), including those for VSAM, non-VSAM, and generation data groups (GDGs). A catalog to be recovered can be shared by multiple systems. You can recover a master catalog if it is not in use as a master catalog.

The base for recovery is an error-free backup of an undamaged BCS. SMF records log the catalog changes and are selected and applied to this backup to create a new image of the catalog to a current state. To produce a current BCS, you then reload the image.

1.2 Recovering an ICF catalog using ICFRU

ICFRU needs a backup copy of the BCS, created by an IDCAMS EXPORT command, as a base file. If this portable copy is not available, then you cannot use ICFRU.

To forward recover an ICF catalog, you also must have a file that contains the SMF record types 61, 65, and 66 from all systems to which the ICF catalog that is being recovered is connected. These SMF records have the information that is required to update the ICF catalog to the point of failure.

ICFRU uses these records to create a new file that contains the updated catalog records. You can then use the new file as input to the IDCAMS IMPORT command to create an error-free copy of the ICF catalog.

ICFRU uses two programs to create the error-free file that can be used to restore the ICF catalog to the point of failure. The first program, Integrated Catalog Forward Recovery Record Selection and Validation (CRURRSV), extracts the SMF records that contain the information that is required to create an updated version of the ICF catalog being recovered. CRURRSV makes the SMF record selections based on the name of the ICF catalog being recovered, the start date and time that is passed to the program, and the end date and time. The selected SMF records are sorted, and this sorted file is used as input to the second program.

The second program, Integrated Catalog Forward Recovery Record Analysis and Processing (CRURRAP), uses the sorted SMF file and the backup file that was created by the IDCAMS EXPORT command to create a new file containing the updated ICF catalog records.

Figure 1-1 on page 3 shows the ICFRU forward recovery process.
To create an error-free copy of the ICF catalog, use the file that was created by ICFRU as input to the IDCAMS IMPORT command. This new version of the ICF catalog is now error free.

ICFRU does not provide any mechanism for locking an ICF catalog. To lock an ICF catalog prior to commencing the recovery, use the IDCAMS ALTER command, specifying the lock parameter. This ensures that no updates take place to the ICF catalog during the recovery process.
Figure 1-2 and Figure 1-3 show an example of the JCL that is required to run ICFRU to forward recover an ICF user or master catalog.

```jcl
/YCJRESIZ JOB MSGCLASS=X,NOTIFY=&SYSUID,RESTART=*  
/SET0   SET IFCAT='UCAT.CRPLUS2'  
/SET1   SET SYSNAME1=SC63  
/SET2   SET SYSNAME2=SC64  
/SET3   SET SYSNAME3=SC65  
/SET4   SET STARTDTE='09/21/01'  
/SET5   SET STARTIME='20:16:47'  
/SET6   SET ENDDATE='09/21/01'  
/SET7   SET ENDTIME='20:30:00'  
/SET8   SET GAPTIME=30  
/SET9   SET CLOCKDIFF=0  
/S1     EXEC PGM=CRURRSY,  
// PARM=('&ICFCAT.,&STARTDTE.,&STARTIME.,&ENDDATE.,&ENDTIME.,&GAPTIME.',  
// ' &CLOCKDIFF.' )  
//SYSPRINT DD SYSOUT=*  
/YSYSLOG DD SYSOUT=*  
/SMFIN DD DSN=YPJRES1.&SYSNAME1..SMFDATA,DISP=SHR  
// DD DSN=YPJRES1.&SYSNAME2..SMFDATA,DISP=SHR  
// DD DSN=YPJRES1.&SYSNAME3..SMFDATA,DISP=SHR  
//SMFOUT DD DSN=YPJRES1.SMFCAT.DATA,DISP=(,CATLG,DELETE),  
// SPACE=(CYL,(10,10),RLSE),UNIT=SYSALLDA  
// IF (S1.RC = 0) THEN  
//S2 EXEC PGM=CRURRAP  
// PARM=('&ICFCAT.,&STARTDTE.,&STARTTIME.,&ENDDATE.,&ENDTIME.,&GAPTIME.',  
// ' &CLOCKDIFF.' )  
//SYSPRINT DD SYSOUT=*  
/YSYSLOG DD SYSOUT=*  
//EXPIN DD DSN=YPJRES1.SMFCAT.DATA.SORTED,DISP=(OLD,DELETE)  
//EXPOUT DD DSN=YPJRES1.SMFCAT.DATA,SORTED,DISP=(,CATLG,DELETE),  
// UNIT=SYSALLDA,SPACE=(CYL,(10,10),RLSE)  
// IF (S2.RC = 0) THEN  
//S3 EXEC PGM=CRURRAP,  
// PARM=('&ICFCAT.,&STARTDTE.,&STARTTIME.,&ENDDATE.,&ENDTIME.,&GAPTIME.',  
// ' &CLOCKDIFF.' )  
//SYSPRINT DD SYSOUT=*  
/YSYSLOG DD SYSOUT=*  
//SMFIN DD DSN=YPJRES1.SMFCAT.DATA.SORTED,DISP=(OLD,DELETE)  
//EXPIN DD DSN=YPJRES1.SMFCAT.DATA,SORTED,DISP=(,CATLG,DELETE),  
// UNIT=SYSALLDA,SPACE=(CYL,(10,10),RLSE)  
// IF (S3.RC = 0) THEN  
/*  
// ENDIF
```

Figure 1-2  Sample JCL for running ICFRU (Part 1 of 2)

```jcl
// IF (S1.RC = 0) THEN  
//S2 EXEC PGM=CRURRAP,  
// PARM=('&ICFCAT.,&STARTDTE.,&STARTTIME.,&ENDDATE.,&ENDTIME.,&GAPTIME.',  
// ' &CLOCKDIFF.' )  
//SYSPRINT DD SYSOUT=*  
/YSYSLOG DD SYSOUT=*  
//SMFIN DD DSN=YPJRES1.SMFCAT.DATA.SORTED,DISP=(OLD,DELETE)  
//EXPIN DD DSN=YPJRES1.SMFCAT.DATA,SORTED,DISP=(,CATLG,DELETE),  
// UNIT=SYSALLDA,SPACE=(CYL,(10,10),RLSE)  
// IF (S2.RC = 0) THEN  
//S3 EXEC PGM=CRURRAP,  
// PARM=('&ICFCAT.,&STARTDTE.,&STARTTIME.,&ENDDATE.,&ENDTIME.,&GAPTIME.',  
// ' &CLOCKDIFF.' )  
//SYSPRINT DD SYSOUT=*  
/YSYSLOG DD SYSOUT=*  
//SMFIN DD DSN=YPJRES1.SMFCAT.DATA.SORTED,DISP=(OLD,DELETE)  
//EXPIN DD DSN=YPJRES1.SMFCAT.DATA,SORTED,DISP=(,CATLG,DELETE),  
// UNIT=SYSALLDA,SPACE=(CYL,(10,10),RLSE)  
// IF (S3.RC = 0) THEN  
/*  
// ENDIF
```

Figure 1-3  Sample JCL for running ICFRU (Part 2 of 2)

The catalog to recover is UCAT.CRPLUS2. The result of the ICFRU program is the file YCJRES1.CRPLUS1.EXPORT.BACKUP(+1).
Figure 1-4 shows the output report from CRURRSV. You can see that the SMF record selection start date was 09/21/01, with a commencing time of 14:24:26 and an end date of 09/21/01 and end time of 15:15:00.

```
INTEGRATED CATALOG FORWARD RECOVERY UTILITY RIMO
CRURRSV SYSPRINT                 09/21/01 (01.264)  17:04:05    PAGE 01
RECORD SELECTION AND VALIDATION REPORT
EXECUTION PARAMETERS
CATALOG NAME               UCAT.CRPLUS2
RECORD SELECTION START     09/21/01 (01.264)  14:24:26
RECORD SELECTION STOP      09/21/01 (01.264)  15:15:00
REPORT FOR ALL SYSTEMS
RECORD SELECTION AND VALIDATION CONDITION CODE IS 0
  274 RECORDS SELECTED FOR UCAT.CRPLUS2
  273 DEFINE (TYPE 61) RECORDS SELECTED
  0 DELETE (TYPE 65) RECORDS SELECTED
  1 ALTER (TYPE 66) RECORDS SELECTED
RECORD SELECTION AND VALIDATION CONDITION CODE IS 0
```

Figure 1-4  ICFRU CRURRSV output report

Figure 1-5 shows the output report from CRURRAP. You can see that 272 records were selected from the SMF data and used in the creation of the new export data set.

```
CRURRAP SYSPRINT                 09/21/01 (01.264) 17:04:07     PAGE 01
REPORT OF RECORDS BY DATA SET
7,868 TOTAL RECORDS IN THE NEW EXPORT DATA SET (EXP.OUT)
  10 CONTROL RECORDS
  7,858 CATALOG RECORDS
7,596 RECORDS FORWARDED FROM THE OLD EXPORT DATA SET (EXP.IN)
  10 CONTROL RECORDS
  7,586 CATALOG RECORDS
272 CATALOG RECORDS SELECTED FROM THE SMF DATA SET (SMFIN)
7,598 TOTAL RECORDS FROM THE OLD EXPORT DATA SET (EXP.IN)
7,596 RECORDS CARRIED FORWARD TO THE NEW EXPORT DATA SET
  10 CONTROL RECORDS
  7,586 CATALOG RECORDS
  2 RECORDS SUPERSEDED OR DELETED (BASED ON SMF DATA)
  0 RECORDS REJECTED BECAUSE OF ERRORS
    0 INVALID LENGTH
    0 UNRECOGNIZED CATALOG RECORD TYPE
274 TOTAL RECORDS FROM THE SMF DATA SET (SMFIN)
272 CATALOG RECORDS SELECTED FROM THE SMF DATA SET
  2 RECORDS SUPERSEDED OR DELETED BY NEWER SMF RECORDS
  0 RECORDS REJECTED
    0 NOT AN MVS SMF RECORD
    0 NOT AN SMF CATALOG RECORD
    0 NOT AN SMF CATALOG RECORD FOR THIS CATALOG
    0 DATE/TIME EARLIER THAN EFFECTIVE START TIME
    0 DATE/TIME LATER THAN EFFECTIVE STOP TIME
7,868 TOTAL OF ALL OUTPUT RECORDS
  4 TOTAL OF ALL RECORDS DISCARDED
7,872 TOTAL OF ALL INPUT RECORDS
```

Figure 1-5  ICFRU CRURRAP output report
1.3 Post recovery actions

After you recover the ICF catalog using ICFRU, you should complete the following tasks:

1. List the self-describing record for the recovered ICF catalog using the IDCAMS LISTCAT command. The format of the LISTCAT command is:
   
   LISTCAT ENT(UCAT.CRPLUS2) ALL CAT(UCAT.CRPLUS2)
   
   If the job completes with a condition code of zero (0), then you have successfully read a catalog record from the recovered ICF catalog.

2. Unlock the ICF catalog using the IDCAMS ALTER command. If you do not unlock the recovered ICF catalog, all users and batch jobs are denied access to any file that is cataloged in it. The format of the ALTER command is:
   
   ALTER UCAT.CRPLUS2 UNLOCK
   
3. Ensure that the structural integrity of the recovered ICF catalog is intact by using the IDCAMS EXAMINE INDEXTEST command. This validates that the vertical and horizontal pointers of the recovered ICF catalog in the index component are correct. The format of the EXAMINE command is:
   
   EXAMINE NAME(UCAT.CRPLUS2) INDEXTEST
   
4. Create a current backup copy of the recovered catalog.
Using Catalog RecoveryPlus

This chapter explains how to use Catalog RecoveryPlus (CR+), an ICF catalog management tool from Mainstar Software Corporation (more information about this product is located at http://www.mainstar.com). CR+ can be licensed directly from Mainstar, or from IBM, Program Number 5620-FGY. It is the catalog management tool that IBM recommends.

With CR+, you can back up and recover ICF catalogs. You can back up both the BCS and VSAM volume data set (VVDS), including forward recovery with SMF records. You can also perform catalog diagnostic and troubleshooting functions and day-to-day catalog maintenance.

This chapter provides basic information about CR+ commands. It also briefly discusses the execution of all CR+ commands through its ISPF interface and the SAF FACILITY class profiles that are available.

The information in this chapter about CR+ does not necessarily include the most recently issued product features. You should consult Mainstar Catalog RecoveryPlus User Guide, document number 006-0202, for complete details. This document can be obtained from Mainstar Software Corporation, for use during trial evaluation and subsequent licensing of the software product.
2.1 Introduction to CR+ facilities

As a reminder, catalogs do break. If you are not prepared for this, your system could have a serious outage while accessing thousands, tens of thousands, or hundreds of thousands of data sets. ICF catalogs, including BCSs and their related VVDSs, are software structures. They are accessed and updated by catalog management software, which is very complicated.

Catalogs are often shared among multiple systems, where in-storage control blocks and buffers are maintained and accessed by each individual system during open, close, and update processing of your data sets. This creates a very complex catalog-management environment. All it takes is the tiniest failure in the software code, and you risk a catalog failure.

Catalog backups, taken frequently and accurately, are your first level of insurance against a failure. Your second level of insurance is knowing how to recover a catalog from a failure and completing that recovery in the shortest possible time. To accomplish this, you must ensure that all catalogs in your environment are being backed up, as frequently as you can possibly afford to.

At a minimum, back up your catalogs at least once a day. For heavily updated catalogs, consider backups that are taken multiple times a day.

The BCS is not the only part of your ICF catalog environment that should be backed up. If you have a complex IBM DB2® environment, for example, that spans across dozens of disk volumes, you should consider backing up the VVDSs in those volumes. This safeguards against a VVDS failure in one volume that would otherwise require you to recover the databases throughout all of those volumes and then bring them up to date from those backups. VVDS backups with CR+ are very fast compared to the time it would take to recover from the failure of a VVDS.

The only way to know how to recover an ICF catalog, whether it is a BCS or VVDS, is through practice. Your installation almost certainly practices a disaster recovery plan, and you should practice catalog recovery as well.

You should not practice this recovery while at your disaster recovery site, but rather, practice recovery at your primary site and on your production catalogs and volumes. This can most easily be done through simulation capabilities, and that is possible with CR+.

The CR+ commands covered in this chapter are:

- BACKUP
- RECOVER
- DIAGNOSE
- MERGECAT
- SUPERCLIP
- ZAP (PRINT, DELETE, PATCH functions)
- REORG
- MAP
- EXPLORE
- GENERATE
- CATSCRUB
2.2 CR+ ISPF interface

All CR+ commands can be built interactively from ISPF panels, or from provided sample batch jobs.

Figure 2-1 shows the Main Menu where you can specify the command that you want to execute.

![CR+ ISPF Main Menu panel](image)

With the CR+ Submit Backup BCS and Backup VVDS panels, you can specify all keywords and parameters as on a batch-submitted command. After the panel is completed, CR+ creates a batch job that is ready for the submission of the BACKUP command. Figure 2-2 on page 10 and Figure 2-3 on page 11 show examples of these panels.
<table>
<thead>
<tr>
<th>Menu</th>
<th>Diagnostics</th>
<th>Preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command ===&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(S) SELECT AN OPTION:
Build JCL/Edit for Submit

Required keywords:
(S) Selected List => No catalogs selected
OUTPUT Data Set Name => 'MHLRES1.THRRE.BACKUP'
OUTFILE (DDName) => OUTFILE OUTPUT dataset must not exist

Optional keywords:
ALIAS => Y Y or N
DIAGNOSE-BCS => Y Y or N
- ACCEPT-DIAGNOSE => W I -Informational, W -Warnings, E -Errors
- DIAGNOSE-ERROR-LIMIT => 16 0-4096
- IGNORE-DIAGNOSE=WARN => N Y or N
DISPOSITION => O O -OLD, S -SHR
EXAMINE
- ACCEPT-EXAMINE => W I -Informational, W -Warnings, E -Errors, D -Disaster
- DATATEST => Y Y or N
- INDEXTEST => Y Y or N
- EXAMINE-ERROR-LIMIT => 16 0-4096
- IGNORE-EXAMINE=WARN => N Y or N
EXCLUDE-BCS(...) => N Y or N
EXCP-MODE => N Y or N - SPAN only valid when EXCMODE=Y
- SPAN-DATASET Name =>
- SPAN-FILE => SPAN-DATASET dataset must not exist
EXPLICIT-VERIFY => N Y or N
FATAL-CATALOG-ERROR => C -Continue, or E -End
MASTERCATALOG => 'MCAT.SANDBOX.Z17.SBOX00'
MESSAGE-TEXT => F N -None, F -Full
PARALLEL => 1-16
PRINT => N N -None, D -Data, K -Key
RESERVE => N Y or N
SIMULATE => N Y or N

Figure 2-2  CR+ BACKUP BCS command panel
Figure 2-3   CR+ BACKUP VVDS command panel

Figure 2-4 on page 12 and Figure 2-5 on page 13 show the CR+ Submit Recover BCS and VVDS panels that create a batch job for the submission of the RECOVER command.
Menu  Diagnostics  Preferences
---------------------------------------------------------------------
Command ===>

(S) SELECT AN OPTION:
    Build JCL/Edit for Submit
Required keywords:
BCS Name =>
INFILE (DDName) =>
INDATASET =>
ALIAS => A -ALIAS, N -NOALIAS, O -ALIASONLY
Optional keywords - General:
    ECSHARING => A -Add, R -Remove, or blank
    EXCLUDE-BCS => N Y or N
    INTO-EMPTY => N Y or N
    NDEFINE => N Y or N
    LOCK => Y Y -LOCK, N -NOLOCK, L -LEAVE-LOCKED
    MASTER-CATALOG => N Y or N
    PRINT => N N -None, D -Data, K -Key
    SIMULATE => D -DEFINE, R -RESTORE, F -FORWARD, or blank
    Add GENDEFCC => N Y or N Delete and Define commands
    Assign to SYSOUT? => N Y or N
    GENDEFCC DSName =>
Optional keywords - FORWARD recovery:
    Recover FORWARD => N Y or N
    SMF Data Set =>
    FROM Date/Time => 20000101 000000 yyyymmdd hhmmss
    TO Date/Time => 20060908 161118 yyyymmdd hhmmss
    SMFERR Data Set =>
Optional keywords - DASD management:
    EXTENT-CONSOLIDATION => N Y or N
    NEW-DATA-PRIMARY =>
    NEW-DATA-SECONDARY =>
    NEW-INDEX-PRIMARY =>
    NEW-INDEX-SECONDARY =>
    NEW-VOLSER =>
Optional keywords - renaming BCS:
    DELETE-OLD-BCS => N Y or N
    JOURNAL Data Set =>
    Generate DEFINE => Y Y or N
    NEW-NAMEx Data Set =>
    VVDS-UPDATE => Y Y or N
Optional keywords - SMS management (do not enter class for default):
    NEW-DATACLAS => N C or N Class:
    NEW-MGMTCLASS => N C or N Class:
    NEW-STORCLASS => N C or N Class:
Optional keywords - VSAM management:
    NEW-BUFND => 3 - 255
    NEW-BUFNI => 2 - 255
    NEW-CI-FSPC => 0 - 100
    NEW-CI-FSPC => 0 - 100
    NEW-DATA-CISZ => Numeric and valid CI size
    NEW-INDEX-CISZ => Numeric and valid CI size
    NEW-STRNO => 2 - 255
Optional keywords - Selective Recovery:
    INCLUDE-TYPE => N Y or N
    EXCLUDE-TYPE => N Y or N
    INCLUDE-DSN => N Y or N
    EXCLUDE-DSN => N Y or N
    INCLUDE-VOLSER => N Y or N
    EXCLUDE-VOLSER => N Y or N

Figure 2-4  CR+ RECOVER BCS command panel
Figure 2-5   CR+ RECOVER VVDS command panel

Figure 2-6 on page 14 shows the panel used to build a job to reorganize a catalog while it is open. See 2.10, “REORG command” on page 42 for more information about this feature.
2.3 SAF FACILITY class profiles

You can control user access to command execution with CR+ FACILITY class profiles that are recognized by RACF® and other security products.

The profiles have been established at the command level (and sometimes at the subcommand level) for maximum flexibility in controlling who will be authorized to use CR+, or, for the potentially destructive command functions, who can execute certain commands or subcommands.

You can examine the permissions that have been granted to your user ID with the SAF Facility Permissions dialog option, which is available as a menu choice option under DIAGNOSTICS.

![Menu Diagnostics Preferences](image)

<table>
<thead>
<tr>
<th>Required keywords:</th>
<th>Optional keywords - General:</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCS</td>
<td>BACKUP =&gt; Y Y or N</td>
</tr>
<tr>
<td></td>
<td>- OUTPUT Data Set Name =&gt; 'MHLRES1.RWO.BKUPS'</td>
</tr>
<tr>
<td></td>
<td>- OUTFILE (DDName) =&gt; OUTDD OUTPUT dataset must not exist</td>
</tr>
<tr>
<td></td>
<td>EXTENT-RELEASE =&gt; Y Y or N</td>
</tr>
<tr>
<td></td>
<td>NO-REPLY =&gt; Y Y or N</td>
</tr>
<tr>
<td></td>
<td>REPAIR =&gt; Y Y or N</td>
</tr>
<tr>
<td></td>
<td>SIMULATE =&gt; Y Y or N</td>
</tr>
<tr>
<td></td>
<td>WHILE-OPEN =&gt; Y Y or N</td>
</tr>
<tr>
<td>Optional keywords - Data Component Attributes:</td>
<td></td>
</tr>
<tr>
<td>NEW-BUFND =&gt; 3 - 255</td>
<td></td>
</tr>
<tr>
<td>NEW-CA-FSPC =&gt; 0 - 100</td>
<td></td>
</tr>
<tr>
<td>NEW-CI-FSPC =&gt; 0 - 100</td>
<td></td>
</tr>
<tr>
<td>NEW-DATA-CISZ =&gt; 512 - 8192, multiples of 512 8192 - 32768, multiples of 2048</td>
<td></td>
</tr>
<tr>
<td>NEW-DATA-PRIMARY =&gt; 1 - 9999999</td>
<td></td>
</tr>
<tr>
<td>NEW-DATA-SECONDARY =&gt; 0 - 9999999</td>
<td></td>
</tr>
<tr>
<td>NEW-DATA-SPACETYPE =&gt; CYL TRK KB MB</td>
<td></td>
</tr>
<tr>
<td>NEW-STRNO =&gt; 2 - 255</td>
<td></td>
</tr>
<tr>
<td>Optional keywords - Index Component Attributes:</td>
<td></td>
</tr>
<tr>
<td>NEW-BUFNI =&gt; 2 - 255</td>
<td></td>
</tr>
<tr>
<td>NEW-INDEX-CISZ =&gt; 512 - 8192, multiples of 512 8192 - 32768, multiples of 2048</td>
<td></td>
</tr>
<tr>
<td>NEW-INDEX-PRIMARY =&gt; 1 - 9999999</td>
<td></td>
</tr>
<tr>
<td>NEW-INDEX-SECONDARY =&gt; 0 - 9999999</td>
<td></td>
</tr>
<tr>
<td>NEW-INDEX-SPACETYPE =&gt; CYL TRK KB MB</td>
<td></td>
</tr>
</tbody>
</table>

---

Figure 2-6  CR+ REORG command panel
2.4 BACKUP command

You can create backup copies of both parts of an ICF catalog (BCS and VVDS) with the CR+ BACKUP command. You can specify one or more BCSs or VVDSs in each BACKUP command, and with mask filtering values, groups of BCSs and VVDSs can be selected in a single command. If you desire, you can back up all connected catalogs, including the system master catalog, with one execution of the BACKUP command. In all cases, all selected BCSs or VVDSs in one command are “stacked” in a single output file.

The BACKUP output file is designated by either the OUTFILE or OUTDATASET keyword. If OUTFILE is coded, it points to an external DD statement, in which the output file data set name and other attributes are specified. If the output data set has been pre-allocated and cataloged, it can be dynamically allocated with the OUTDATASET keyword.

The BACKUP command has several features for analyzing the integrity of BCSs and VVDSs just prior to their backup that you can invoke with IDCAMS EXAMINE or DIAGNOSE commands “under the covers.” To run these commands, you specify keywords, and the output reports from these commands are subsequently inserted into the user output. Also, you can use a return code keyword to designate whether backup processing should continue, based on the level of return code returned from these IDCAMS commands.
In addition to backing up the BCS or VVDS records, the BACKUP command backs up the information necessary to delete and define the BCS or VVDS automatically at the time of recovery.

2.4.1 Specifying BCSs to be backed up

The BACKUP BCS command allows you to specify a single BCS (Figure 2-8) for back up to the output file that is designated by the OUTFILE keyword.

```
//BCKCPY DD DSN=BKUP.CATBAKS(+1),DISP=(,CATLG),
   UNIT=SYSALLDA,SPACE=(CYL,(10,10),RLSE)
BACKUP BCS(UCAT.CRPLUS1) OUTFILE(BCKCPY)
```

Figure 2-8  Back up a single BCS

Or, you can back up multiple, explicitly-specified BCSs, all of which are backed up to the single output file that is designated by the OUTFILE keyword (Figure 2-9).

```
//BCKCPY DD DSN=BKUP.CATBAKS(+1),DISP=(,CATLG),
   UNIT=SYSALLDA,SPACE=(CYL,(10,10),RLSE)
BACKUP BCS(UCAT.CRPLUS1 UCAT.CRPLUS2 UCAT.CRPLUS3) OUTFILE(BCKCPY)
```

Figure 2-9  Back up multiple BCSs, explicitly specified

You can also back up multiple BCSs by specifying a mask filter value. Again, all of the BCSs selected by the mask filter are backed up to the single output file. In Figure 2-10, all BCSs that match the mask value UCAT.CRPLUS* are backed up.

```
//BCKCPY DD DSN=BKUP.CATBAKS(+1),DISP=(,CATLG),
   UNIT=SYSALLDA,SPACE=(CYL,(10,10),RLSE)
BACKUP BCS(UCAT.CRPLUS*) OUTFILE(BCKCPY)
```

Figure 2-10  Back up multiple BCSs, with a mask filter

When using mask filter values to identify BCSs, you can also use the EXCLUDEBCS keyword (Figure 2-11) to specifically exclude those BCSs that you do not want to be in the selection list that the mask filter creates.

```
//BCKCPY DD DSN=BKUP.CATBAKS(+1),DISP=(,CATLG),
   UNIT=SYSALLDA,SPACE=(CYL,(10,10),RLSE)
BACKUP BCS(UCAT.CRPLUS*) EXCLUDEBCS(UCAT.CRPLUS10) OUTFILE(BCKCPY)
```

Figure 2-11  Back up multiple BCSs, excluding one
Back up all BCSs in the system

A valuable feature of the BACKUP command allows you to specify an all-powerful mask of ** (double asterisk), where the ** value indicates all catalogs in the system. This technique, which Figure 2-12 illustrates, includes in its selection list all “connected” catalogs in the master catalog in the system that is running the BACKUP command. This includes the master catalog itself and all other master catalogs that are connected to this master catalog as user catalogs.

![Figure 2-12 Back up all BCSs in the environment](image)

The reason this technique is so valuable is that you can add and remove user catalogs or change names of existing catalogs without having to worry about updating the catalog backup jobs.

A real-world example of how things can go wrong can illustrate the value of this technique. A system programmer at an installation discovered a “broken” catalog after running an EXAMINE INDEXTEST on all of their catalogs. The EXAMINE error message indicated duplicate key and out-of-sequence records in a data control interval (CI) of his catalog. A search for the backup of that particular BCS, so that a recovery could be started, revealed that there was no backup. Two years earlier, when the catalog was first allocated, someone forgot to add the appropriate new EXPORT step in the catalog backup job for this catalog, and all this time passed without any backups being taken. With the double-asterisk facility in the CR+ BACKUP command, any new, deleted, or renamed catalogs are automatically put into or removed from the selection list the next time BACKUP is run.

Back up BCSs at different frequencies

In many ICF environments, some BCSs are volatile. Others are relatively static, depending on the frequency of allocations and deletions of data sets. Also, the speed of recovery that is required for one BCS might be significantly different than that for other BCSs.

For these reasons, you should consider setting up multiple backup jobs, where some are run at different frequencies than others, and each job identifies the specific BCSs and VVDSs to be backed up at that time. The ease with which this can be done depends on how good your BCS naming conventions are, because to do the backups, you can specify mask values for specific groups of BCSs to be backed up in each job.

For example, you might want all BCSs that catalog application development data sets (assume these are highly active) backed up every four hours and those BCSs that catalog stable production data sets backed up once a day. You could specify the following mask in your command for the backups that occur every four hours:

```
BACKUP BCS(UCAT.DEV.**) ...
```

Or, you code the following mask in your command that backs up once a day:

```
BACKUP BCS(UCAT.PROD.**) ...
```
2.4.2 Specifying VVDSs to be backed up

With the BACKUP VVDS command, you have similar options for specifying one or more VVDSs to be backed up in a single command. All selected VVDSs are backed up to a single output file.

To make coding simpler, the actual data set name of the VVDS is not specified, but rather, you specify the *volser* for the VVDS to be backed up. Since the standard (and required) data set name of a VVDS is SYS1.VVDS.V*volser*, it is a simple matter for CR+ to convert the specified *volser* list into VVDS names.

You can specify a single VVDS to be backed up to the output file that is designated by the OUTFILE keyword (Figure 2-13).

![Figure 2-13 Back up a single VVDS](image)

Or, you can specify multiple, explicit VVDS names (Figure 2-14), all of which are backed up to the single output file that is designated by the OUTFILE keyword.

![Figure 2-14 Back up multiple VVDSs, explicitly specified](image)

Or, you can specify a *volser* mask so that all the VVDSs that are selected by the mask filter are be backed up to the single output file. In Figure 2-15, all VVDSs that match the mask value VBOX* are backed up.

![Figure 2-15 Back up multiple VVDSs, with a mask filter](image)

Or, you can back up the VVDSs in all the volumes in your entire system (Figure 2-16).

![Figure 2-16 Back up all VVDSs in the environment](image)

When you are identifying VVDSs with mask filter values, you can also use the EXCLUDEVVDS keyword. This keyword specifically excludes those VVDSs that you do not want to include in the selection list that the mask filter creates.
Figure 2-17 shows an example of how to exclude one VVDS.

```
//BCKCPY DD DSN=BKUP.VVDSBAKS(+1),DISP=(,CATLG),
// UNIT=SYSALLDA,SPACE=(CYL,(10,10),RLSE)
BACKUP VVDS(VBOX*) EXCLUDEVVDS(VBOX10) OUTFILE(BCKPY)
```

Figure 2-17  Back up multiple VVDSs, excluding one

2.4.3 Diagnosing BCS or VVDS integrity prior to backup

The structural integrity of BCSs and VVDSs is vital to successful operation of your system. Yet, people frequently say, “Catalogs do not break anymore,” or “I have RAID storage now, and with a mirrored copy, I never have catalog failures.” This ignores the fact that many catalog failures are “soft” in nature, where there are problems in the BCS or VVDS, but you simply do not know about them yet.

With some catalog errors, you can run for weeks or months without even realizing that there is a problem. Eventually, however, some required process is likely to fail as a result of that error. Worse, when the error finally manifests itself, it could take the form of a catastrophic failure, and it could occur at the worst possible time. The sooner you identify the failure and fix it, the more likely you are to recover with less difficulty or to recover more successfully.

For that reason, the absolute minimum you should perform is an IDCAMS EXAMINE, with the INDEXTEST keyword, on every BCS prior to backing it up. You should also run a DIAGNOSE ICFCATALOG at regular intervals without the COMPAREVVDS keyword specified. This performs an inside-the-BCS record structure and relationship check to determine if data content within the BCS is becoming corrupted.

Using the CR+ BACKUP command, you can issue these commands at your discretion just prior to starting the backup. The ACCEPTEXAMINE and ACCEPTDIAGNOSE keywords control whether an error should stop your backup process. The warning and error messages from the diagnostic commands that you see in your output listing give you a clearer picture of whether the BCSs and VVDSs that you are backing up are sound:

- **EXAMINE(INDEXTEST | DATATEST)**
  The IDCAMS EXAMINE command performs its structure test about as well as any software can, so by default, CR+ internally issues an IDCAMS EXAMINE INDEXTEST just prior to backing up each BCS. If you request it with the keyword EXAMINE(DATATEST), the CR+ BACKUP internally issues EXAMINE with this option. If neither is desired, you always have the option to specify NOEXAMINE.

- **DIAGNOSEBCS**
  This keyword indicates that the IDCAMS DIAGNOSE ICFCATALOG command, without the COMPAREVVDS keyword, is to be invoked by the CR+ BACKUP BCS command just prior to backing up each BCS. When specified, DIAGNOSE ICFCATALOG reads through every VSAM and non-VSAM catalog record in the BCS, checking for consistency and accuracy of all data fields in each record and for relevant relationships across all records throughout the BCS.

- **DIAGNOSEVVDS**
  This keyword indicates that the IDCAMS DIAGNOSE VVDS command, without the COMPAREBCS keyword, is to be invoked by the CR+ BACKUP VVDS command just prior to backing up each VVDS. When specified, DIAGNOSE VVDS reads through every record in the VVDS, checking for consistency and accuracy of all data fields in each VSAM volume record (VVR) and non-VSAM volume record (NVR).
2.4.4 Restricting access to the BCS or volume during backup

IDCAMS EXPORT processing issues an exclusive ENQ for update throughout the time that a BCS is being backed up. By contrast (and by default), the CR+ BACKUP BCS and VVDS commands do not issue a similar ENQ, and therefore, do not restrict update or locate access during backup processing. This means that while BACKUP is running, DELETE, DEFINE, and ALTER processing can occur coincidentally, with the slight possibility that a small amount of “fuzziness” occurs in the backup. This fuzziness can be identified by running two-way CR+ DIAGNOSE commands following any catalog recovery. You can then correct it with any “fix” commands that are generated from DIAGNOSE.

The benefit of this logic with CR+ is that you can schedule catalog backups more frequently with less impact on your environment. For this reason, we recommend that you run BACKUP BCS with NORESERVE in effect (which is the default), unless you can specifically identify a risk factor that is unacceptable to you.

If you prefer to restrict catalog access during backup, this can be done with the RESERVE keyword in the BACKUP command. The RESERVE keyword, in the case of the BACKUP BCS command, does not issue a volume reserve (as its name implies), but rather the same exclusive ENQ for update that EXPORT issues.

When backing up the VVDS, similar considerations apply with one difference. When RESERVE is specified for VVDS backup processing, an ENQ with the RESERVE option is in effect throughout the time of the backup. This locks up the entire volume from any I/O for the duration of the VVDS backup (which is only 1 to 2 seconds), and is something you should take into account.

2.4.5 Creating multiple backup copies

With the CR+ BACKUP command, you can automatically create multiple backup copies for as many copies as the output files that are included in your OUTDATASET or OUTFILE keyword specification. Figure 2-18 shows how easy it is to code a BACKUP command for multiple copies of a backup.

```
//BCKCPY1 DD DSN=BKUP1.CATBACKS(+1),DISP=(,CATLG),
   UNIT=SYSALLDA,SPACE=(CYL,(10,10),RLSE)
//BCKCPY2 DD DSN=BKUP2.CATBACKS(+1),DISP=(,CATLG),
   UNIT=SYSALLDA,SPACE=(CYL,(10,10),RLSE)
BACKUP BCS(**) OUTFILE(BCKCPY1,BCKCPY2)
```

*Figure 2-18 Sample duplex BACKUP command*

Do not forget that backup files need to be cataloged so that they can be located when a recovery is necessary. If the catalog that needs to be restored happens to be the one where the backup is cataloged, you could be in big trouble.

One solution is to establish a fundamental catalog backup methodology that plans ahead for this situation. To do this, create two dedicated catalogs to be used just for cataloging the backups of your ICF catalog environment. Assign each catalog a single alias that identifies the catalog backups that will be cataloged in it. With two catalogs, where you catalog only the catalog backups in each catalog, there is less risk of failure to your backup catalog. You would then run the BACKUP command, specifying backup copies with different high-level qualifiers in their data set names, so that each backup copy is cataloged in its corresponding catalog.
With this technique of multiple backup copies, each cataloged in a different catalog, if a failure occurs in one catalog where your BCS or VVDS backups are cataloged, you can use the other copy to recover.

Cataloging of your SMF data is another important consideration. Because these files are required for forward recovery of a BCS or VVDS, what if the failing catalog points to your SMF files? To ensure accessibility, you should create your backup methodology so that you can always access your cataloged backup files and SMF data.

One concern you might have is how difficult this is to set up. The command in Figure 2-18 on page 20 shows how to use CR+ to code a duplex BACKUP step that matches our logic flow diagram in Figure 2-19.

All BCSs throughout the entire system are backed up in a single command, creating two copies: one routed to the file on the BCKCPY1 DD statement and the other routed to the BCKCPY2 DD statement. The high-level qualifier in the data set name for BKUP1 and BKUP2 are the aliases of different user catalogs, causing each backup copy to be cataloged in a different location.

Another concern might be how much additional processing a duplex backup adds to your BCS or VVDS backup job. As you would expect, there will be a relatively small increase in EXCPs, but other performance-related factors should not change. To illustrate this, we created two sets of tests in our lab: the first with duplex BCS backups and the second with duplex VVDS backups.
In our duplex BCS backup test, we backed up 28 BCSs, where Test1 created a single backup copy and Test2 created two backup copies. The 28 BCSs had a cumulative total of 63,216 catalog records (which is not very large, but this example is intended to give you relative differences). As you can see in Table 2-1, the EXCPs only increased from 4,008 in Test 1 to 4,253 in Test 2, or slightly more than 5%, for a test that was a complete duplication of the backup files.

Table 2-1  Duplex BCS backup overhead statistics

<table>
<thead>
<tr>
<th>Test</th>
<th>Backup copies</th>
<th>Elapsed time</th>
<th>CPU time</th>
<th>EXCPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>1</td>
<td>15 sec</td>
<td>.02 sec</td>
<td>4,008</td>
</tr>
<tr>
<td>#2</td>
<td>2</td>
<td>15 sec</td>
<td>.02 sec</td>
<td>4,253</td>
</tr>
</tbody>
</table>

In our duplex VVDS backup test, we backed up 66 VVDSs, where Test 3 created a single backup copy and Test 4 created two backup copies. During BACKUP selection processing, the specified volser mask encountered and identified 29 volumes that would not be included because they did not have a VVDS allocated on them. Our test BACKUP job automatically ran the IDCAMS DIAGNOSE VVDS on each VVDS (and in fact, encountered one VVDS that had errors, so the test times included dumping the VVDS record to SYSPRINT). All VVDSs were allocated at the default size of TRK(10 10). As you can see in Table 2-2, the elapsed time was exactly the same (recording accuracy was only to a second), CPU time was the same (recording accuracy of 1/100th of a second), and EXCPs increased by less than 1%.

Table 2-2  Duplex VVDS backup overhead statistics

<table>
<thead>
<tr>
<th>Test</th>
<th>Backup copies</th>
<th>Elapsed time</th>
<th>CPU time</th>
<th>EXCPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>#3</td>
<td>1</td>
<td>1 min, 54 sec</td>
<td>.11 sec</td>
<td>33,667</td>
</tr>
<tr>
<td>#4</td>
<td>2</td>
<td>1 min, 54 sec</td>
<td>.11 sec</td>
<td>33,926</td>
</tr>
</tbody>
</table>

2.4.6  Alias processing with BACKUP BCS

If the default keyword ALIAS is in effect for the BACKUP command, all related aliases for the BCS are copied from the current master catalog to the backup copy at the start of backup for each BCS. If there is a subsequent RECOVER process, these aliases are restored to the master catalog for the system running the RECOVER command. It is also possible to recover ALIASONLY, which allows you to use this facility to replicate aliases to multiple master catalogs. If, for some reason, you do not want aliases backed up at all, you can specify NOALIAS on the BACKUP command.

2.4.7  Bypassing index processing with BACKUP BCS

The CR+ BACKUP command automatically opens the BCS as a data set. In other words, the Access Method Control Block (ACB) indicates that you are opening an ICF catalog but the open process takes place “outside of” catalog management. The command specifically opens the data component of the BCS for read access, without using the index. With this logic, BACKUP reads through the data component of the BCS in physical sequential order, deblocking records from the CIs in the BCS, reblocking them, and writing them out to the backup.

This is unlike most other BCS backup tools, such as EXPORT, which open the BCS as a catalog, that is, as a KSDS. If there are corruptions in the index, the result can be unrecoverable backup failures.
We illustrate the importance of this logic in 3.1, “Case study #1: Ensuring good backups” on page 48, where the specific type of BCS corruption prevented IDCAMS EXPORT from backing up all of the records in the BCS, but the CR+ BACKUP backed up the entire BCS. The difference was specifically due to this processing logic.

2.5 RECOVER command

With the CR+ RECOVER command, you can restore or forward recover either a BCS or a VVDS from a backup copy that was created either by IDCAMS EXPORT (BCS only) or the CR+ BACKUP command.

If the backup was created by CR+ BACKUP, the records are first sorted and then written to the BCS or VVDS being recovered. This logic alerts you (by way of informational or error messages) to any duplicate key records.

The RECOVER input file is designated by either the INFILE or INDATASET keyword. INFILE points to an external DD statement that includes the input file data set name and other attributes. INDATASET specifies the data set name, and a dynamic allocation is performed. Because the backup file contains a copy of all BCSs or VVDSs that are backed up by a single BACKUP command, this file is accessed to the point where the relevant records for the BCS or VVDS to be recovered are found.

2.5.1 Restore versus forward recovery

The RECOVER command is fundamentally a restore BCS or VVDS facility, although it can also forward recover. It might help to explain these terms, restore and recover, in some detail.

**Restore**

The term *restore* means that RECOVER is to retrieve backup records from the input file and write them to the output BCS or VVDS exactly as they are in the backup file, without updating them with SMF records.

If that is all you want to do, the restored BCS or VVDS then has records in it that are identical to those in the BCS or VVDS at the exact time of BACKUP processing. This is usually an appropriate function to perform in situations where your BACKUP and RECOVER command are executed in quick succession and no BCS or VVDS updates have occurred between the time of the BACKUP and the time of the RECOVER. Valid examples of when you can perform just a restore include:

- For a BCS, if you decide to reorganize it
- For a BCS, if you need to change any of the allocation parameters (such as allocated size, volser, CI size, the FREESPACE parameter), remove the IMBED parameter, or make other changes
- For a BCS, when you are renaming it (see 2.5.6, “Renaming the BCS during restore” on page 25, for more details about this facility)
- For a BCS, when you determine that the error condition is a structural index failure, and to repair it, you must backup and restore, to rebuild the index (see 3.1, “Case study #1: Ensuring good backups” on page 48, for a real-life situation where this occurred)
- For a VVDS, if you need to change the physical allocated size (see 3.5, “Case study #5: Enlarging VVDS” on page 60, for details and an example of how to do this)
**Forward recovery**
Forward recovery of a BCS or VVDS is the process of applying all updates to the BCS or VVDS that occurred between the backup start time and the restore, at the time the restore is taking place. When this process is to be performed, it is typical for the FROMBACKUPDATE and TOCURRENTDATE options to be in effect. The backup time is retrieved from the backup file and used as the starting time from which SMF records are analyzed. The forward recovery ending time is the time that the RECOVER command was initiated. If desired, alternate FROM and TO dates and times can be specified.

### 2.5.2 Specifying the BCS or VVDS to restore

RECOVER allows only a single BCS or VVDS to be specified in one execution of the command.

With RECOVER BCS (Figure 2-20), the specification must be the fully-qualified name of the BCS as it was at the time of BACKUP (for details about how to rename the BCS, see 2.5.6, “Renaming the BCS during restore” on page 25).

```
//BCKCPY DD DSN=BKUP.CATBAKS(0),DISP=SHR
RECOVER BCS(UCAT.CRPLUS1) INFILE(BCKCPY)
```

*Figure 2-20  RECOVER command for a BCS*

For RECOVER VVDS (Figure 2-21), the specification must be the full *volser* of the volume where the VVDS is to be recovered (and not the actual VVDS data set name).

```
//BCKCPY DD DSN=BKUP.VVDSBAKS(0),DISP=SHR
RECOVER VVDS(VBOX01) INFILE(BCKCPY)
```

*Figure 2-21  RECOVER command for a VVDS*

### 2.5.3 Explicit/implicit DEFINE of the BCS or VVDS to restore

Prior to beginning the restore process, a check is made to determine if a BCS or VVDS of the same name already exists (and, in the case of a BCS, if it is connected to the current master catalog). If one is found and INTOEMPTY is not specified, RECOVER issues an IDCAMS DELETE command (with RECOVERY specified) that physically deletes the BCS or VVDS.

The RECOVERY keyword in the DELETE command tells IDCAMS that a BCS or VVDS is being deleted for *recovery purposes* and that all data sets defined in it are to remain untouched. Then, using the self-describing information that was written to the backup file, an IDCAMS DEFINE USERCATALOG (in the case of RECOVER BCS) or DEFINE CLUSTER command (in the case of RECOVER VVDS) is constructed and issued.

The RECOVER command allows you to specify new values for primary and secondary allocation quantities and any of the VSAM attributes, such as FREESPACE or CISZ. However, if you want, you can issue your own DELETE/DEFINE of the BCS or VVDS. Then, you can specify INTOEMPTY on the RECOVER command. You can also use this technique if you want to change a BCS name. Refer to 2.5.6, “Renaming the BCS during restore” on page 25, for more details.
2.5.4 Locking versus unlocking the BCS

RECOVER BCS, by default of the LOCK keyword, issues an IDCAMS ALTER LOCK command on the named BCS prior to recovery so that it is in effect throughout the time that the restore and forward recovery processing is performed. We strongly recommend that you use this facility because it ensures exclusive access by your recovery job to the BCS while it is being recovered. You do not want application production jobs issuing data set searches or data set cataloging (and uncataloging) while the recovery is in progress. Locking makes certain that no other accesses are made to the BCS while the RECOVER command is running.

For the LOCK to be successful, you must define the RACF profile IGG.CATLOCK prior to issuing the RECOVER command. If you do not define it, an IDC3009I message is issued with a return code 186, reason code 2, and RECOVER terminates without further processing.

You can also issue the IDCAMS ALTER LOCK command to the BCS prior to recovery and then specify the NOLOCK keyword in the RECOVER command.

2.5.5 Alias handling during BCS restore/recovery

BACKUP processing dumps all aliases for the BCS to the output file. By default, they are redefined to the current master catalog during RECOVER processing. As a prerequisite of RECOVER processing, whether you explicitly issue the commands yourself or let RECOVER do it implicitly, the existing version of the BCS is deleted and disconnected from the master catalog, and all aliases for the BCS are deleted from the master catalog. Restoring them to the master catalog from the BCS backup file is the easiest and safest way to establish the aliases for the newly restored/recovered BCS.

If your RECOVER command also performs a forward recovery, any new or deleted aliases for this BCS that are identified by SMF records that are different from those on the backup file are also applied during the processing of the recovery.

If your environment has shared access to the BCS that is being restored or recovered, you can issue the CR+ DIAGNOSE ALIAS command after RECOVER is complete to compare the aliases for this BCS with those in another master catalog. Alternatively, if you want to update the aliases for other master catalogs in your environment directly, you can run the RECOVER command with the MASTERCATALOG keyword specified. Specifying this keyword restores all aliases for the specified BCS in up to 32 named master catalogs.

2.5.6 Renaming the BCS during restore

For many installations, new BCS allocations have occurred over a long period of time, with various personnel assigned to catalog management. In other installations, corporate mergers or data center consolidations have caused catalogs to be imported into the environment from other locations. As a result, BCS naming convention standards are in great disorder, and it is virtually impossible to use BCS mask values for day-to-day catalog maintenance functions.

With the Catalog RecoveryPlus RECOVER BCS ... NEWNAME keyword, it is possible to rename any BCS relatively quickly and easily. You simply allocate a BCS with the new name and then “point” the RECOVER BCS command to that BCS by specifying the name in the NEWNAME keyword. All VVRs and NVRs are changed to reference the new BCS name.
Figure 2-22 illustrates how to code this and the full steps involved in accomplishing a BCS rename.

```
//S1 EXEC -- Catalog RecoveryPlus, with all of its regular JCL --
//BCKCPY DD DSN=BKUP.CRPLUS1,DISP=(,CATLG),
//  UNIT=SYSALLDA,SPACE=(CYL,(10,10),RLSE)

BACKUP BCS(UCAT.CRPLUS1) OUTFILE(BCKCPY)

//S2 EXEC PGM=IDCAMS
//SBOX02 DD VOL=SER=SBOX02,UNIT=SYSALLDA,DISP=SHR
//SBOX36 DD VOL=SER=SBOX36,UNIT=SYSALLDA,DISP=SHR
//SYSPRINT DD SYSOUT=*  
//SYSSIN DD *

DELETE UCAT.CRPLUS1 -
  USERCATALOG -
  FILE(SBOX02) -
  RECOVERY

DEFINE USERCATALOG -
  (NAME(UCAT.CRPLUS5) -
   IFCATALOG -
   CYL(10) -
   BUFND(27) -
   STRNO(8) -
   VOLUME(SBOX36) -
   DATA(CISZ(4096)) -
   INDEX(CISZ(1024)) -
   BUFNI(4))

//S3 EXEC -- Catalog RecoveryPlus, with all of its regular JCL --
//BCKCPY DD DSN=BKUP.CRPLUS1,DISP=SHR

RECOVER BCS(UCAT.CRPLUS1) -
  INFILE(BCKCPY) -
  INTOEMPTY -
  NEWNAME(UCAT.CRPLUS5) -
  JOURNAL(YCJRES2.CRPLUS1.RECOVER.JOURNAL)
```

Figure 2-22  RECOVER BCS with NEWNAME

To illustrate an additional idea of what you can do, this example also moves the BCS from volume SBOX02 to volume SBOX36. As you can see, a forward recovery with SMF is not required, provided that, during the time between BACKUP and RECOVER, there are no updates to the BCS that is being renamed (in other words, you should ensure that the BCS is quiesced throughout this entire period).

Note that we specified a journal file with this version of the RECOVER command. It is used for the VVDS update process, which updates all BCS back-pointers from the old to new BCS name.

### 2.5.7 SMF data for forward recovery

Forward recovery of a BCS uses SMF record types 61, 65, and 66, which represent:

- Type 61: A data set DEFINE
- Type 65: A data set DELETE
- Type 66: A data set ALTER

Forward recovery of a VVDS uses SMF record type 60, which represents a VVDS record insert, update, or delete. It applies to both VVR changes for VSAM data set components and NVR changes for system managed non-VSAM data sets.
Only “dumped” SMF data is acceptable to RECOVER. Therefore, you must issue an I SMF “switch” command for the in-use MANX/Y file and then dump the SMF data to a file that can be used as input to the RECOVER process.

With the CR+ RECOVER command, it is not necessary to pre-extract the SMF records that are to be used. Instead, you can simply gather up all SMF dump data from all systems sharing the BCS or VVDS. As illustrated in Figure 2-23 and Figure 2-24, you then concatenate all of the dumped SMF files in the DD statement that is identified in the FORWARD(SMFFILE(ddname)) keyword.

```
//BCKCPY DD DSN=BKUP.CATBAKS(0),DISP=SRH
//SMFINDD DD DSN=YCJRES1.SC63.SMFDATA,DISP=SHR
// DD DSN=YCJRES1.SC64.SMFDATA,DISP=SHR
// DD DSN=YCJRES1.SC65.SMFDATA,DISP=SHR
RECOVER BCS(UCAT.CRPLUS1) INFILE(BCKCPY) -
   FORWARD(SMFFILE(SMFINDD))
```

Figure 2-23  RECOVER BCS FORWARD(SMFFILE) command

```
//BCKCPY DD DSN=BKUP.VVDBAKS(0),DISP=SRH
//SMFINDD DD DSN=YCJRES1.SC63.SMFDATA,DISP=SHR
// DD DSN=YCJRES1.SC64.SMFDATA,DISP=SHR
// DD DSN=YCJRES1.SC65.SMFDATA,DISP=SHR
RECOVER VVDS(VBOX01) INFILE(BCKCPY) -
   FORWARD(SMFFILE(SMFINDD))
```

Figure 2-24  RECOVER VVDS FORWARD(SMFFILE) command

Figure 2-25 on page 28 shows the BACKUP/RECOVER BCS logic flow.
In the first step, the BACKUP BCS command creates a backup copy of multiple BCSs in a single file. This file is used in the second step to forward recover BCS4. You can see how the SMF dump files for all the systems are used along with the backup copy of BCS4.

**Note:** Sample BACKUP BCS commands are shown in 2.4.1, “Specifying BCSs to be backed up” on page 16.

Figure 2-26 on page 29 shows the BACKUP/RECOVER VVDS logic flow.
In the first step, the BACKUP VVDS command creates a backup copy of multiple VVDSs into a single file. This file is used in the second step to forward recover the VVDS in VOL003. You can see how the SMF dump files for all the systems are used along with the backup copy of the VVDS.

**Note:** Sample BACKUP VVDS commands are shown in 2.4.2, “Specifying VVDSs to be backed up” on page 18.

### 2.5.8 Printing BCS records during a RECOVER

If you want to see a process log printout of the RECOVER command execution, you can use the PRINT keyword as follows:

```
PRINT (NONE | DATA | KEY)
```

The operand parameters indicate:

- **NONE:** This is the default. It specifies that records from the backup file and, in the event of a forward recovery, that merged records from the SMF file will not be printed.
- **DATA:** This specifies the complete record, including key and data from the backup file. In the event of a forward recovery, merged records from the SMF file are printed.
- **KEY:** This specifies that only the key from the backup file, and, in the event of a forward recovery, that merged records from the SMF file will be printed. KEY is not a valid specification for RECOVER VVDS, but if coded, it changes to DATA.
2.5.9 Simulation as a BCS or VVDS recovery testing tool

An important key to successful BCS or VVDS recovery—helping you to achieve recovery that is completed in the shortest possible time frame—is practicing and testing your recovery procedures. It is important to note, however, that meaningful testing is often difficult to perform.

Many people tell us that they frequently “test” their catalog recovery procedures at the beginning of their overall disaster recovery test. This is definitely not meaningful because they are most likely testing the successful restore of a catalog and not a forward recovery. Other people tell us that they rarely test catalog recovery at all, presumably because they believe that catalogs never break. As a result, they never test to determine if their knowledge of catalog recovery is sufficiently good, nor do they test whether they have procedures in place to locate and pull the necessary SMF data together and other technical issues that affect successful recovery.

A keyword, SIMULATE, is provided with the CR+ RECOVER command for both BCS and VVDS recovery. Its FORWARD subkeyword allows you to fully test a complete forward recovery of either a BCS or a VVDS. You can test against very realistic situations, including large, highly active production BCSs or volumes that would otherwise not be available for testing. You can also use the SIMULATE facility to see if you have good procedures for collecting all SMF data from all systems that share the BCS or VVDS.

SIMULATE(FORWARD) performs a complete simulation of the restore and forward recovery of the specified BCS (Figure 2-27) or VVDS (Figure 2-28).

```
RECOVER BCS(UCAT.CRPLUS1) INFILE(BCKCPY) -
FORWARD(SMFFILE(SMFINDD)) -
SIMULATE(FORWARD)
```

*Figure 2-27 RECOVER BCS SIMULATE(FORWARD) command*

```
RECOVER VVDS(VBOX01) INFILE(BCKCPY)
FORWARD(SMFFILE(SMFINDD)) -
SIMULATE(FORWARD)
```

*Figure 2-28 RECOVER VVDS SIMULATE(FORWARD) command*

The output report from the command execution is exactly the same as it would be if the actual forward recovery was done. As you would expect, SIMULATE does not delete and define the output BCS or VVDS, nor does it actually write any records to the output BCS or VVDS. (As a result of this last statement, you should not use SIMULATE for timing tests of recovery.)
2.5.10 Resizing the VVDS during RECOVER processing

In certain circumstances, an already allocated VVDS could prove to be too small, and you want to enlarge it. This can be particularly true for volumes where the VVDS was automatically allocated when the first data set was allocated on the volume that needed the VVDS. In that situation, the default allocation size for a VVDS is TRK(10 10).

If there are data sets already on the volume that have records in the VVDS, your only option with standard MVS™ facilities is to unload all data sets that have records in the VVDS from the volume. Then you delete and re-define the VVDS at your chosen size, and finally reload the data sets.

While this sounds like an easy process, it can be time-consuming and tedious for volumes with large numbers of existing data sets. With system-managed volumes, both VSAM and non-VSAM data sets have records in the VVDS, making this process even more difficult.

With the CR+ BACKUP and RECOVERY commands, you can accomplish this with the following command sequence without having to remove any data sets from the volume:

1. BACKUP the VVDS.
2. Run an IDCAMS DELETE (with RECOVERY specified) of the VVDS to physically remove it from the volume.
3. Run an IDCAMS DEFINE CLUSTER to re-allocate the VVDS at your desired larger size.
4. RECOVER the VVDS from the backup (forward recovery is not required).

To see actual commands and job stream output from this process, refer to 3.5, “Case study #5: Enlarging VVDS” on page 60.

Note: Before you attempt to enlarge a VVDS, first determine from a LISTCAT the current total allocated size of the VVDS, and factor that into the new, larger size that you are contemplating. Also, determine whether the existing VVDS has acquired multiple extents, and make certain that you include all of these into your size calculation. The reason for this is that, during CR+ RECOVER VVDS processing, it is not possible to allocate secondary extents, and if there is not sufficient space in the VVDS’s primary allocation, RECOVER fails with an error message, and you must DELETE/DEFINE, and start over.

2.6 DIAGNOSE command

CR+ provides its own version of a DIAGNOSE command with certain functions that appear to be similar to those in IDCAMS, but with significant differences. The CR+ DIAGNOSE command does not call IDCAMS DIAGNOSE but instead processes the selected BCSs and VVDSs with its own logic.

The primary design of the CR+ DIAGNOSE command is to provide an easy-to-code and extremely high-performance ICF catalog diagnostic facility that can be run on a frequent and scheduled basis, to help you keep your BCSs and VVDSs as clean and error-free as possible. Each DIAGNOSE command can generate “fix” commands for all of the types of error situations that it looks for. These fix commands are in the form of IDCAMS commands that are stored in a physical sequential (flat) file. You can browse, edit, and then issue these commands when you are ready and comfortable with them.
2.6.1 DIAGNOSE ALIAS command

The DIAGNOSE ALIAS command is designed to compare two master catalogs to determine if their user catalog connections are consistent and if alias relationships for their connected user catalogs are synchronized. In 3.3, “Case study #3: Synchronizing aliases” on page 54, we show the DIAGNOSE ALIAS command with some of the fix commands that it generates.

This command is most likely executed:

- After master catalog maintenance, where aliases have been updated in one master catalog and you want to replicate that maintenance to one or more system master catalogs
- After a user or master catalog recovery
- After a MERGECAT of one or more alias groups from one user catalog to another

PEER versus NONPEER processing

There are two fundamental ways to run the DIAGNOSE ALIAS command using the PEER or NONPEER keywords.

In PEER mode (Figure 2-29), both specified master catalogs are considered as “equals.” The comparison logic identifies situations in each master catalog that the other does not have and creates fixes to make each master catalog “look” like the other. For example, if either master catalog has an alias that the other does not have, a DEFINE ALIAS fix command is created.

```
//FIXDD DD DSN=YCJRES1.CRPLUS.DIAG ALIAS.FIXFILE,DISP=(,CATLG),
// UNIT=SYSALLDA,SPACE=(CYL,(1,1),RLSE),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)

DIAGNOSE ALIAS PEER -
   COMPARE-MASTERCATALOG(MCAT.SANDBOX.R10 MCAT.SANDBOX.Z02) -
   FIXDATASET(YCJRES1.CRPLUS.DIAG ALIAS.FIXFILE)
```

Figure 2-29 DIAGNOSE ALIAS PEER mode

With NONPEER (Figure 2-30), the first-specified master catalog is considered to be “superior” to the second-specified master. The comparison logic identifies situations where the superior master contains entries that the subordinate master does not have. For example, if the first master catalog has an alias that the second does not have, a DEFINE ALIAS fix command is created, but if the second has an alias that is not in the first, a fix command is not created.

```
//FIXDD DD DSN=YCJRES1.CRPLUS.DIAG ALIAS.FIXFILE,DISP=(,CATLG),
// UNIT=SYSALLDA,SPACE=(CYL,(1,1),RLSE),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)

DIAGNOSE ALIAS NONPEER -
   COMPARE-MASTERCATALOG(MCAT.SANDBOX.R10 MCAT.SANDBOX.Z02) -
   FIXDATASET(YCJRES1.CRPLUS.DIAG ALIAS.FIXFILE)
```

Figure 2-30 DIAGNOSE ALIAS NONPEER mode
EXCLUDE and INCLUDE keywords
The DIAGNOSE ALIAS command provides EXCLUDE and INCLUDE keywords for both alias and user catalog specification so that you can tailor the execution of the command more closely to your requirements.

The example in Figure 2-31 shows a situation where we only wanted to synchronize the master catalogs with respect to aliases that are related to the user catalogs under our control.

```
//FIXDD DD DSN=YCJRES1.CRPLUS.DIAG.ALIAS.FIXFILE,DISP=(,CATLG),
// UNIT=SYSALLDA,SPACE=(CYL,(1,1),RLSE),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)

DIAGNOSE ALIAS NONPEER -
    COMPARE-MASTERCATALOG(MCAT.SANDBOX.R10 MCAT.SANDBOX.Z02) -
    INCLUDE-USERCATALOG(UCAT.CRPLUS*) -
    FIXDATASET(YCJRES1.CRPLUS.DIAG.ALIAS.FIXFILE)
```

Figure 2-31  DIAGNOSE ALIAS with INCLUDE-USERCATALOG option

Error analysis
DIAGNOSE ALIAS looks for several types of discrepancies for which fixes are generally created, including:

- User catalogs that are connected in one master catalog, but not in the other
- Alias entries in one master catalog that are not in the other (under the rules of the PEER and NONPEER keywords)
- Aliases that exist but do not have any data sets cataloged “under” them in the related user catalog; these are considered “empty” aliases
- Inconsistent aliases, where an alias related to a user catalog is defined in one master catalog but the same alias is defined in the other master catalog and related to a different user catalog

DIAGNOSE ALIAS also looks for user catalogs that have no aliases defined. Effectively, these are unusable user catalogs, because it is not possible to direct cataloging of data sets to them without an alias (assuming that you are not using STEPCAT/JOBCAT statements, for which support has been dropped, and should not be used).

This situation is not considered an error, so a fix is not created. Instead you receive an information message that alerts you that there are user catalogs without aliases.

For more information about the fixes that are created for each of these situations, refer to 3.3, “Case study #3: Synchronizing aliases” on page 54. All the fixes in this case study were generated as comment statements, and as such, cannot be run as IDCAMS commands until the /* at the beginning of each command is removed.

Because the fix file is a physical sequential file, the /* characters can be removed using TSO ISPF edit. In many situations, this might be a laborious task, so the DIAGNOSE ALIAS command has the DEFINE-ALIAS and DELETE-ALIAS keywords available. If you know in advance which command is more appropriate for your environment, the command processor builds those commands in the fix data set as “ready-to-execute.”
2.6.2 DIAGNOSE BCS-VVDS and VOLUME-BCS commands

The DIAGNOSE command has two modes of operation for comparing entries in one or more BCSs against one or more VVDSs, or one or more VVDSs against one or more BCSs.

DIAGNOSE BCS-VVDS logic

One mode, BCS-VVDS, compares entries in the selected BCSs to the selected VVDSs. In simplest terms, the primary function of this mode is to identify BCS records that do not have matching and corresponding records in the VVDS. In other words, this command is looking for “orphaned” BCS records, and for each one found, an IDCAMS DELETE clustername NOSCRATCH command is generated in the fix data set. There are other fix commands that may be generated, but this is, by far, the most common one.

All fix commands are prefixed by a /*, making them comment statements. You must either remove the /* with TSO EDIT, or you can specify the DELETE-ENTRIES keyword in the DIAGNOSE command and all fixes are generated as “ready-to-execute” in the fix data set.

DIAGNOSE VOLUME-BCS logic

The other mode, VOLUME-BCS, compares entries in the selected VVDSs to the selected BCSs. In simplest terms, the primary function of this mode is to identify VVDS records that do not have matching and corresponding records in the appropriate BCS. In other words, this command is looking for “orphaned” VVDS records. The solution and fix to this is more difficult, however, because there are two possible interpretations:

- The data set is one that you want access to and it needs an IDCAMS DEFINE CLUSTER ... RECATALOG command.
- The data set (or component) was deleted and has now reappeared, in which case, it must be physically deleted with an IDCAMS DELETE objectname VVR/NVR command. Since the DIAGNOSE command cannot determine which fix command is appropriate, it generates both in the fix data set and the decision of which is correct is left to you. By default, both commands are prefixed by a /*, to make them into comment statements. You either have to remove the /* preceding the desired fix command with TSO EDIT or you can specify the DEFINE-RECATALOG or DELETE-ENTRIES keyword in the DIAGNOSE command, if you know in advance which fix is appropriate.

DIAGNOSE VOLUME-BCS performs extensive volume-level diagnostics (VTOC, VTOCIX and VVDS) to identify inconsistencies between those structures, building appropriate fix commands for any errors found. When an apparent “orphaned” component is identified, extensive analysis determines if it is part of a multi-volume data set.

Selecting BCSs and VVDSs for DIAGNOSE

In both modes of operation, you can explicitly specify:

- One or more BCSs or VVDSs
- Mask values
- Storage groups of volsers
- ** (double asterisk)

EXCLUDE keywords are provided, so that individual BCSs/VVDSs or masked BCSs/VVDSs can be removed from the selection list.

Possibly the most useful keyword is ALLRELATED, which causes the DIAGNOSE command processor to determine all of the “related” VVDSs automatically when the comparison is from BCS-to-VVDS, or “related” BCSs when the comparison is from VOLUME-to-BCS. To determine the valid “all related” list, this facility does not search for cataloged
SYS1.VVDS.Vvolser entries in the BCSs, nor does it use the BCS name list in the VVCR record in the VVDS. Instead, it searches through each and every record in the BCS or VVDS to identify volume pointers for VVDSs with data sets and BCS back-pointers in VVR/NVR records. Once the list is built, the diagnosis is made to every BCS or VVDS in the list.

In Figure 2-32, the DIAGNOSE BCS-VVDS command diagnoses all BCSs (as indicated by COMPAREBCS(**), against “all related” VVDSs. All errors identified by this DIAGNOSE will have fixes created and stored in the FIXDATASET specified in the command. See 3.9, “Case study #9: Preventive maintenance” on page 74, for actual output from a sample execution of this command.

In Figure 2-33, the DIAGNOSE VVDS-BCS command diagnoses all of our ITSO sandbox volume VVDSs as indicated by COMPAREVVDS(SBOX*), against “all related” BCSs (the default). Fixes are created for all errors that are identified by this DIAGNOSE and the fixes are stored in the FIXDATASET specified in the command. See 3.9, “Case study #9: Preventive maintenance” on page 74, for actual output from a sample execution of this command.
2.7 MERGECAT command

A BCS is a single point of failure for all the data sets cataloged in it. Part of managing availability is minimizing the impact of outage of any one catalog. Most z/OS systems have hundreds of thousands of data sets cataloged in a handful of catalogs. Brief examples of a few environments include:

- Installation A has 1,300,000 data sets and 25 user catalogs. The largest catalog contains 27% of the total data sets. The top 4 catalogs contain 73% of the data sets.
- Installation B has 10 user catalogs and 592,620 data sets. Sixty-eight percent of them are cataloged in a single user catalog.
- Installation C has 3,700,000 data sets in 22 user catalogs. Ninety-seven percent are cataloged in the top 5 catalogs. The largest single catalog contains 79% of the production data sets.

How are your cataloged data sets spread across your user catalogs? What application outage would you experience if you lost one of your critical user catalogs? Analyze your environment and initiate a project to spread your cataloged data sets across more user catalogs.

The CR+ MERGECAT command provides functions that are similar to the IDCAMS REPRO MERGECAT and NOMERGECAT commands. It either moves or copies records from one BCS to another, so you can use it to merge, split, or clone BCS records.

In general, you can use the MERGECAT command to move all or some BCS records to another BCS. Depending on the keywords that you code with this command, aliases can be changed from the input BCS to the output BCS.

When you are not using the COPYONLY keyword, the MERGECAT command acts as a move function, and therefore, VVDS records for all data sets whose catalog entry is moved have a BCS back-pointer update.

Figure 2-34 illustrates the command that is used for moving all records in a BCS to another BCS.

```
MERGECAT INBCS(UCAT.CRPLUS3) -
   OUTBCS(UCAT.CRPLUS5) -
   JOURNAL(YCJRES1.MERGECAT.JOURNAL)
```

*Figure 2-34  MERGECAT command to move all BCS records*

Every catalog record in UCAT.CRPLUS3 is moved to UCAT.CRPLUS5, every alias for UCAT.CRPLUS3 in the master catalog is moved to UCAT.CRPLUS5, and UCAT.CRPLUS3 becomes an empty catalog. For every data set whose catalog record is moved, all the VVDS records (VVRs and NVRs) change their BCS back-pointer values from UCAT.CRPLUS3 to UCAT.CRPLUS5.

In Figure 2-35, you see the command used to move all records with a high-level qualifier of YCJRES3 from user catalog UCAT.CRPLUS3 to UCAT.CRPLUS5.

```
MERGECAT INBCS(UCAT.CRPLUS3) -
   OUTBCS(UCAT.CRPLUS5) -
   LEVEL(YCJRES3) -
   JOURNAL(YCJRES3.MERGECAT.JOURNAL)
```

*Figure 2-35  MERGECAT command to move BCS records by LEVEL*
By using the level keyword, the alias YCJRES3 is also changed in the master catalog from its relationship to UCAT.CRPLUS3 to UCAT.CRPLUS5.

In Figure 2-36, all catalog entries in UCAT.CRPLUS3 that begin with ITSO1 (regardless of anything else in their names) are moved to UCAT.CRPLUS5. This type of process is most likely done when entries such as these are cataloged incorrectly into the source BCS and you must move them to the correct BCS (and even more likely, they actually have the alias of ITSO1). With the ENTRIES keyword, the MERGECAT command does not make any alias changes.

```
MERGECAT INBCS(UCAT.CRPLUS3) -
  OUTBCS(UCAT.CRPLUS5) -
  ENTRIES(ITSO1.**) -
  JOURNAL(YCJRES1.MERGECAT.JOURNAL)
```

Figure 2-36   MERGECAT command to move BCS records by ENTRIES

### 2.7.1 Alias handling

A unique feature of the CR+ MERGECAT command is that it reassociates user catalog aliases to the target catalog. This only happens when the move is specified with the LEVEL keyword, and therefore, only when an entire alias group of records is moved.

The significance of the emphasis is, if you specify the following LEVEL example, you successfully move all catalog records that match those two levels of data set name qualification:

```
LEVEL(YCJRES1.CRPLUS2)
```

But, if your system is running with MLA=1 (multi-level alias support of 1), only the high-level qualifier of YCJRES1 is set up as an alias, and it will not be reassociated to the target catalog. The reason is that you might well have many other data sets under this same high-level qualifier, such as YCJRES1.CRPLUS1, and you would make these data sets inaccessible if the alias of YCJRES1 was reassociated. In fact, you actually “orphan” the data sets with high-level qualifier YCJRES1.CRPLUS2, because they are undoubtedly now located in a BCS that does not (and cannot) have an alias of YCJRES1.

This type of operation should only be done when you are splitting catalogs apart, and most likely, getting ready for multi-level alias support. Or you should do it when you are performing system consolidations or workload balancing where you are moving whole categories of data sets over to user catalogs that are accessible from different systems.

When you use the ENTRIES keyword, aliases are never reassociated, even if you move all catalog records for an entire alias group.

### 2.7.2 Using the journal file

MERGECAT uses a journal file in part to provide RESTART or BACKOUT capability if the command should fail during processing. In the event of failure, you can rerun the MERGECAT command, coded exactly as before, and add either the RESTART or BACKOUT keyword (see 2.7.3, “RESTART and BACKOUT capability” on page 38 for details about the RESTART and BACKOUT keywords).
It is common for users to place the DELETE and DEFINE of the journal file in a step that immediately precedes the MERGECAT command, because MERGECAT needs a “clean” journal each time it runs. Therefore, it is important to recognize that this operation must not be done in a RESTART or BACKOUT situation, which allows the command to use the journal records from the previous, interrupted run.

2.7.3 RESTART and BACKOUT capability

A second powerful use of the journal file is to provide the ability to restart and back out of a failed MERGECAT operation, simply by running the same command again, but with the RESTART or BACKOUT keyword coded:

- If RESTART is specified, the command resumes at the point of processing that is indicated by the current status of the journal file. Assuming that the problem that caused the interruption is corrected, the MERGECAT process should continue to completion.

- If BACKOUT is specified, the command resumes at the point of processing that is indicated by the current status of the journal file, but it re-runs all of the already-completed operations to that point, in reverse processing order, and undoes each operation.

The RESTART or BACKOUT operation is possible if you receive the following messages in the SYSOUT report on the failing or interrupted job:

CAT12000I >>> JOURNAL FILE HAS BEEN POPULATED
CAT12000I >>> MERGECAT IS NOW RESTARTABLE

2.7.4 SIMULATE capability

The SIMULATE keyword provides the ability to test the command in advance, ensuring that it performs exactly what you expect it to do. The output report from the command looks exactly as it would if you ran it “for real,” but there are no I/Os to the output BCS, aliases are not moved, and VVDS back-pointer updates are not performed.

The last thing you can tolerate is a failure of the MERGECAT, particularly if it caused by a command coding failure on your part. With the SIMULATE capability, you can run the MERGECAT in advance in a simulation mode, so that you can see exactly what the command is going to accomplish.

Because the CR+ MERGECAT command provides a line-by-line process report of every data set moved (which IDCAMS REPRO MERGECAT does not), you can verify exactly what data sets are to be moved and information about any and all user catalog aliases that are to be re-associated to the target catalog. The simulations can be done as far in advance as you want. When you feel comfortable that your command is performing exactly as you want it to, you then have the confidence that it should work when you do it “for real” within your downtime window.

2.7.5 Catalog activity during MERGECAT processing

In many instances, particularly with user catalog merging and splitting of records, you want to run MERGECAT when the source catalog is quiesced. In other words, all data set records being moved or copied must be in a closed state.

Job termination problems almost certainly occur if you move catalog records from one BCS to another while the actual data set for that catalog record is open to a processing job. CAS has to write CLOSE information to the catalog and its in-storage control blocks only know where the catalog record came from, not where it is now supposed to go. This is a particular problem
for online data sets, which are open for long periods of time, requiring you to schedule
downtime when these catalog records to be moved are in a quiesced condition.

To help you control this, MERGECAT gives you the option of internally issuing an ALTER
LOCK command at the beginning of processing:

- If the LOCK keyword is specified (or left to default), MERGECAT internally issues an
  IDCAMS ALTER … LOCK command to both the source and target BCSs, ensuring
  serialized access throughout execution.

- If the NOLOCK keyword is specified, MERGECAT does not lock the source and target
  BCSs at the start of move or copy processing, allowing other catalog access within the
  BCS during the time that MERGECAT is performing its operations.

2.7.6 Creating an alternate master catalog

The CR+ MERGECAT command has a COPYONLY keyword for copying records from one
BCS to another. Although this can be used in some situations for setting up test catalogs, it is
most frequently used to create and maintain an alternate master catalog. The command is
similar to IDCAMS REPRO NOMERGECAT, with some major differences.

With IDCAMS REPRO NOMERGECAT, the BCS back-pointers in each copied VVDS record
are updated. With CR+ MERGECAT COPYONLY, the BCS back-pointers are not
updated, even if you specifically code the VVDSUPDATE keyword in the command. Other
major differences include the ability to RESTART, BACKOUT, and SIMULATE in this mode.

The implication of the VVDS back-pointer update rule is that, when the CR+ MERGECAT
COPYONLY operation is used to create an alternate master catalog, the source remains your
official production master catalog, and the target becomes your alternate master catalog.

With IDCAMS REPRO NOMERGECAT, where the VVDS back-pointer entries are
reassociated, this makes the target the new official master catalog, and the source is now the
alternate master catalog. Therefore you must do an IPL again after the command is
completed to make the change effective.

If you are doing this on a regular basis as you perform master catalog updates, you are forced
to “flip-flop” back and forth with respect to which master catalog is the official one and which is
the alternate. With the logic in CR+ MERGECAT COPYONLY, your production master catalog
remains the same at all times, and you continually create a new alternate master with this
command.

2.8 SUPERCLIP command

CR+ provides the SUPERCLIP command for changing the volser of disk volumes.

2.8.1 Changing a volser

Changing a disk volser with the ICKDSF utility is an operation that is quite regularly
performed in many installations. It is commonly referred to as a volser “clip,” which is an old,
S/360™ term that means “change label in process.” It has survived in everyday use, even
though many people do not have any idea what it stands for. It is most commonly done for
empty volumes.
Changing a volser of a volume that has data sets already allocated on it is a bit more troublesome. With standard MVS utility functions, you must first move all data sets from the volume by copying the data and then deleting the data set. You then VARY the volume offline to each system that is online to it. You run ICKDSF to clip the volser to its new value. Once you issue VARY to bring the volume online, you can allocate and reload the data sets on the volume.

With the CR+ SUPERCLIP command, you simply code a command (Figure 2-37). SUPERCLIP then performs the entire volser change for you without unloading and reloading the data sets on the volume.

```
SUPERCLIP OLDVOLSER(oldvol) NEWVOLSER(newvol) -
JOURNAL(YCJRES1.SUPERCLP.JOURNAL)
```

*Figure 2-37   SUPERCLIP example syntax*

The SUPERCLIP command uses a journal file, and this provides you with a process log of all steps that SUPERCLIP performs. It also has RESTART, BACKOUT, and SIMULATE capabilities. You can see the complete output from the SUPERCLIP command in 3.6, “Case study #6: Changing a disk volser” on page 62.

### 2.8.2 Caution when using SUPERCLIP

SUPERCLIP is an extremely powerful facility, but one that you must use with utmost caution. Consider a RACF FACILITY class profile definition that restricts who can use this command. Also, make certain that the volume is already varied offline to all other systems that share the device, because SUPERCLIP does not issue the VARY to systems other than the one on which the command is running.

### 2.9 ZAP command

The CR+ ZAP command is a multi-purpose command that you can use to:

- Print one or more BCS records.
- Delete a BCS record.
- Patch a BCS record.
- Print one or more VVDS records.
- Delete a VVDS record.
- Patch a VVDS record.
- Print one or more VTOC records.
- Delete a VTOC record.
- Patch a VTOC record.
- Rename a VTOC record.
- Print one or more data set records.
- Delete a data set record.
- Patch a data set record.

There is a wide range of variations for specifying each capability. Refer to *Mainstar Catalog RecoveryPlus User Guide*, document number 006-0202, for specific details about how all the commands can be coded and how each process works.
2.9.1 Printing BCS and VVDS records

You can print one or more BCS or VVDS records with the ZAP command. Figure 2-38 shows examples of how you might code this command. In each example shown, you can specify the output to be in FORMAT, DUMP, or CELLDUMP layout.

```
To print a single BCS record, by key specification:
  ZAP BCS(UCAT.CRPLUS1) -
    PRINT(KEY(YCJRES1.CRPLUS1.KSDS.CLUSTER))
To print 5 BCS records, in ascending key sequence, by key specification:
  ZAP BCS(UCAT.CRPLUS1) -
    PRINT(KEY(YCJRES1.CRPLUS1.KSDS.CLUSTER) COUNT(5))
To print 5 BCS records, by generic key specification:
  ZAP BCS(UCAT.CRPLUS1) -
    PRINT(KEY(YCJRES1.CRPLUS1.K**) COUNT(5))
To print a single VVDS record, by component name specification:
  ZAP VVDS(SBOX01) -
    PRINT(COMponent(YCJRES1.CRPLUS1.KSDS.DATA))
To print a single VVDS record, by RBA offset:
  ZAP VVDS(UCAT.CRPLUS1) -
    PRINT(RBA(000096DE))
To print 5 VVDS records, in physical sequence, starting at a component name location:
  ZAP VVDS(SBOX01) -
    PRINT(KEY(YCJRES1.CRPLUS1.KSDS.CLUSTER) COUNT(5))
```

Figure 2-38  ZAP PRINT of BCS and VVDS records

2.9.2 Deleting a BCS and VVDS record

You can DELETE a BCS or VVDS record with the ZAP command (Figure 2-39).

```
To delete a BCS record, by key specification:
  ZAP BCS(UCAT.CRPLUS1) -
    DELETE(KEY(YCJRES1.CRPLUS1.KSDS.CLUSTER))
To delete a BCS record, but only if it is verified to contain a certain value at a specified location within the record:
  ZAP BCS(UCAT.CRPLUS1) -
    DELETE(KEY(YCJRES1.CRPLUS1.KSDS.CLUSTER) -
    VER(04,C4))
To delete a single VVDS record, by component name specification:
  ZAP VVDS(SBOX01) -
    DELETE(COMponent(YCJRES1.CRPLUS1.KSDS.DATA))
To delete a VVDS record, by RBA offset within the VVDS, but only if it is verified to contain a certain value at a specified location within the record:
  ZAP VVDS(SBOX01) -
    DELETE(RBA(000096DE) -
    VER(04,F9))
To delete a VVDS record, at an RBA offset within the VVDS:
  ZAP VVDS(SBOX01) -
    DELETE(COMponent(YCJRES2.CRPLUS.TESTK.K000953.DATA) - RBA(0005C000))
```

Figure 2-39  ZAP DELETE of a BCS and VVDS record

You can only delete one record at a time with this command. It is also very important to understand that this command, if successful, deletes only the single record that you “point” it to and does not delete any associated records. For example, if you delete a BCS cluster sphere record, it does not delete any associated truename records for the cluster.
components (if they exist). If you delete a VVDS record (a VVR or NVR), it does not delete an associated VTOC record (by contrast, an IDCAMS DELETE VVR/NVR command does). Therefore, you should be careful that you do not cause problems greater than you are trying to solve.

### 2.9.3 Patching BCS and VVDS records

You can PATCH a BCS or VVDS record with the ZAP command. You can only patch one record at a time with this command. You should be very careful when executing this command, because it is extremely powerful, and you are essentially “on your own” whenever you “zap” a BCS or VVDS record. Figure 2-40 shows examples of how you might code this command.

![Figure 2-40](image)

To patch a BCS record, but only if both VER and REP conditions are met:

```
ZAP BCS(UCAT.CRPLUS2) -
PATCH(COMPONENT(YCJRES2.CRPLUS.TESTK.K000954) -
VER(04,X'E3',37,X'24',58,X'F1') -
REP(58,X'F2'))
```

To patch a VVDS record, but only if both VER and REP conditions are met:

```
ZAP VVDS(SBOX79) -
PATCH(COMPONENT(YCJRES2.CRPLUS.TESTK.K000954.DATA) -
VER(04,X'E9',1F,X'E3E2E3E1') -
REP(1F,X'E3E2E3E4'))
```

The ZAP command also has an ISPF interface that provides on-screen formatting of BCS, VVDS, and VTOC records, cells, and fields.

### 2.10 REORG command

A scheduled EXPORT/IMPORT or backup and recovery procedure is frequently used to reorganize or change attributes of a catalog. You can use the CR+ REORG command to do both without scheduling an outage. Keep in mind that splits are not bad. VSAM is simply doing what it was designed to do, which is to obtain additional free space when and where it is needed.

Catalogs that grow rapidly typically contain data set names that end with a time stamp. Catalogs that undergo CI and control area (CA) splits caused by the pattern of update activity continue to do so after they have been reorganized. However, when a BCS exhausts available extents, processing can come to a sudden halt. So, you should not reorganize solely because of some number of CI or CA splits but do reorganize before a catalog is in danger of running out of extents.

Figure 2-41 on page 43 shows an example of a REORG command where space allocation and CI size changes are made and it is specified that unused extents are to be released.
Figure 2-41 Sample REORG command

In 3.10, "Case study #10: Reorganizing a catalog" on page 79, you can see the output of a simple REORG.

2.11 Other CR+ Commands

Catalog availability is assured by more than backup and recovery procedures. This section provides a brief overview of the following CR+ catalog management commands:

- **MAP**
- **EXPLORE**
- **GENERATE**
- **CATSCRUB**

2.11.1 MAP command

You can run the CR+ MAP command if you want to map and analyze the internal structure of any VSAM key-sequenced data set (KSDS), including a BCS. The reports show you where CA splits occur and help determine if you have too little or too much free space. It also detects “dead” control intervals. Neither of these situations constitutes a broken catalog but both result in wasted space and are the reasons why many installations schedule reorganizations using EXPORT/IMPORT or some backup/recover scenario.

Contrary to popular belief, splits are not bad. VSAM obtains free space when and where it is needed with these splits. If records are inserted randomly throughout the BCS, specifying FREESPACE improves performance as it is designed to do. However, if inserts are concentrated in a few places, free space that is scattered throughout the catalog goes unused and splits continue to occur at the insertion point. The MAP command can show you where CA splits are occurring and help you decide whether to specify any free space for the catalog.

“Dead” control intervals result when the sequence set control interval is not large enough to hold a pointer for every data CI within the CA. A sequence set CI should have a high key and pointer for each CI in the related data CA. The key value is compressed. When selecting the index CI size, VSAM makes an assumption as to how much the key will compress. If keys do not compress as expected, dead CIs result. The MAP command alerts you to the presence of dead CIs and makes recommendations for eliminating the problem.

Six reports are available from the MAP command including: Catalog Listing, Control Area Analysis, Free CI Distribution, Recordsize Distribution, Index Structure, Statistical Summary, and Recommendations.
Figure 2-42 shows two examples of how you might code this command.

```
MAP BCS(UCAT.CRPLUS2) REPORTS(ALL)
MAP DSN(cluster.name) REPORTS(CA-MAP RECOMMENDATIONS)
```

You can view report samples in 3.12, “Case study #12: Mapping a BCS” on page 82. Refer to `Mainstar Catalog RecoveryPlus User Guide`, document number 006-0202, for specific details of interpreting report contents.

### 2.11.2 EXPLORE command

The CR+ EXPLORE command allows you to search your ICF catalog environment for data sets that match any of over 90 attributes. It is very easy to use. The output can be presented as a simple list of data set names or in a format that is similar to LISTCAT for each data set selected. You can also obtain the output in an extract file that you can post-process with the REPORT command or with your own program for generating custom reports.

The GENMAP keyword can be used to create and execute a CR+ MAP command for each data set selected. Figure 2-43 on page 45 shows two examples of the EXPLORE command that were generated by using the ISPF panels.
Figure 2-43  Sample EXPLORE commands

The first example generates a list of names of user catalogs in more than 50 extents or with more than 200 CA splits. The MAP command then runs for each of the identified catalogs.

The second example generates a list of user catalogs or KSDS data components with the obsolete attributes of IMBED or REPLICATE. Output from this example is included in 3.11, “Case study #11: Locating data sets with IMBED option” on page 81.
2.11.3 GENERATE command

The CR+ GENERATE BCS-UNLOAD command produces control statements that can be used as input to IDCAMS based on existing records in one or more specified BCSs. It is primarily designed for building a new master catalog but can serve many other purposes. It is a replacement for the IBM utility MCNVTCAT, which was designed to convert a master catalog from VSAM to ICF and is no longer distributed.

GENERATE can save a tremendous amount of time when you need to recreate DEFINE commands for GDGs or IMPORT CONNECT and DEFINE ALIAS commands for user catalogs. These were common uses for MCNVTCAT long after the master catalog had been converted to ICF.

Unlike MCNVTCAT (which did not do VSAM), GENERATE also allows you to recreate DEFINE commands for catalogs, clusters, AIXs, paths, and page data sets. Figure 2-44 illustrates the use of GENERATE in the original MCNVTCAT mode to recreate catalog connector records from a master catalog.

```
GENERATE BCS-UNLOAD
BCS(      -
  SYSS80.MASTER.CATALOG -
    )            -
NOCANDIDATE -
INCLUDE-TYPE(  -
  USERCATALOG -
  USERCATALOG-ALIAS -
    )         -
JOBCARD      -
MCNVTCAT     -
OUTFILE(GENOUT)
```

Figure 2-44 Sample GENERATE command

The keyword JOBCARD means IDCAMS JCL as well as control statements will be created in the data set pointed to by OUTFILE. The output of this command can be found in 3.13, “Case study #13: GENERATE” on page 86.

2.11.4 CATSCRUB command

Different installations have different disaster recovery plans. CR+ supports either selective recovery or “scrubbing” as the technique to ensure that catalogs reflect the actual contents of the restored volumes. The CR+ CATSCRUB command is typically used when full volumes are restored at the disaster recovery site and you must synchronize restored catalogs with the restored volumes. The scrubbing process compares each catalog record with actual data sets and deletes entries for data sets that are not found. Extensive anomaly keywords allow you to specify which entries should be kept although no data set is present (for example, empty GDG bases, migrated GDSs, and tape data sets).

CATSCRUB is very fast. A test was run where 127,242 catalog entries were scrubbed from a catalog. The elapsed time was 12 minutes. It had previously taken more than 4 hours to accomplish a similar cleanup with IDCAMS. Refer to Mainstar Catalog RecoveryPlus User Guide, document number 006-0202, for details about the CATSCRUB command.
Case studies

This chapter presents several real-life examples of how ICF catalog structures can “break.” The studies provide actual job listings that contain specific command sequences that can be used to analyze each situation and to recover from the errors. They use commands and facilities from IDCAMS, VVDSFIX (an unsupported ad hoc tool from IBM), and Mainstar Catalog RecoveryPlus (CR+), including:

- IDCAMS EXPORT and IMPORT
- IDCAMS EXAMINE and DIAGNOSE
- VVDSFIX DELBCSR and DELVVR
- CR+ BACKUP and RECOVER
- CR+ DIAGNOSE
- CR+ REORG
- CR+ EXPLORE
- CR+ MAP

The information in this chapter is not intended to be all-encompassing. There is no limit on the ways that catalogs can break. Even in two similar breakage situations, the symptoms can be different and manifest themselves in different ways. The case studies are intended to be examples that can help you develop a methodology and sequence of analysis to use when debugging catalog and catalog address space problems.

The case studies that are examined in this chapter include:

- “Case study #1: Ensuring good backups” on page 48
- “Case study #2: BCS backup and forward recovery” on page 50
- “Case study #3: Synchronizing aliases” on page 54
- “Case study #4: VVDS backup and forward recovery” on page 57
- “Case study #5: Enlarging VVDS” on page 60
- “Case study #6: Changing a disk volser” on page 62
- “Case study #7: Alternate master catalog” on page 67
- “Case study #8: Print and delete BCS/VVDS records” on page 70
- “Case study #9: Preventive maintenance” on page 74
- “Case study #10: Reorganizing a catalog” on page 79
- “Case study #11: Locating data sets with IMBED option” on page 81
- “Case study #12: Mapping a BCS” on page 82
- “Case study #13: GENERATE” on page 86
3.1 Case study #1: Ensuring good backups

This case study illustrates the importance of ensuring that the BCS you are backing up is structurally sound—and that all records from the actual BCS are contained in the backup file.

Example 3-1 begins with an IDCAMS VERIFY command that does not find anything “out of the ordinary” with the BCS. The subsequent IDCAMS EXAMINE command finds structural errors in its analysis of the data and the index component. This includes what it considers to be a minor index error and a major data error, where there is a data component CA that is apparently not indexed by any level 1 sequence set record. The IDCAMS DIAGNOSE command that follows does not find any data content errors. The fact that the EXAMINE command raised a condition code 8 should alert you that this BCS is broken in some way, either localized around a specific area of the BCS or catastrophically. Regardless of what occurs in the following steps of this case study, whatever is broken in this catalog must be fixed as soon as possible.

Example 3-1   Verifying the structure and content of the BCS

//YCJRES1X JOB MSGCLASS=X,NOTIFY=YCJRES1,RESTART=*  
//S1 EXEC PGM=IDCAMS  
//SYSPRINT DD SYSOUT=*  
//SYSIN    DD *  
VERIFY DATASET(UCAT.VSBOX09)  
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

EXAMINE NAME(UCAT.VSBOX09) DATATEST
IDC01700I INDEXTEST BEGINS  
IDC11724I DATA COMPONENT CA NOT KNOWN TO SEQUENCE SET  
IDC21700I MINOR ERRORS FOUND BY INDEXTEST  
IDC01700I DATATEST BEGINS  
IDC11724I DATA COMPONENT CA NOT KNOWN TO SEQUENCE SET  
IDC21703I MAJOR ERRORS FOUND BY DATATEST  
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 8

DIAGNOSE ICFCATALOG INDATASET(UCAT.VSBOX09)
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 8

Example 3-2 illustrates an EXPORT backup of this BCS that was run immediately after the DIAGNOSE step in Example 3-1. Note that the EXPORT output indicates that the number of BCS records backed up is 4,722. Condition code 0 indicates that no errors were encountered.

Example 3-2   Backing up the BCS with EXPORT

//YCJRES1X JOB MSGCLASS=X,NOTIFY=YCJRES1,RESTART=*  
//S1 EXEC PGM=IDCAMS  
//SYSPRINT DD SYSOUT=*  
//OUT1   DD DSN=YCJRES1.EXPORT.BACKUP2,DISP=(,CATLG,DELETE), 
//       UNIT=SYSALLDA,SPACE=(CYL,(1,1),RLSE)  
//SYSIN    DD *  
EXPORT UCAT.VSBOX09 -  
    OUTFILE(OUT1) -  
    TEMPORARY
Example 3-3 illustrates a CR+ BACKUP command for this BCS. The most significant aspect to note in the command output is that it indicates a total of 26,207 backed up records. The EXPORT output in Example 3-2 on page 48 indicates that 4,722 records were backed up on the same BCS and with a condition code of 0.

**Example 3-3  Backing up the BCS with CR+ BACKUP**

//YCJRES1Z JOB MSGCLASS=X,NOTIFY=YCJRES1,RESTART=*  
//S1 EXEC PGM=CAT00010  
//OUT1     DD DSN=YCJRES1.VSBOX09.BACKUP2,DISP=(,CATLG),  
//          UNIT=SYSALLDA,SPACE=(CYL,(1,1),RLSE)  
//SYSPRINT DD SYSOUT=*  
//SYSUDUMP DD SYSOUT=*  
//SYSIN    DD *  
// BACKUP BCS(UCAT.VSBOX09) -  
//       OUTFILE(OUT1)  

The difference is that EXPORT treats the BCS as though it were a KSDS, using the index component as the reference vehicle for the data records. If there is an error in the index that is not detected by EXPORT, the result could be an incomplete or incorrect backup. CR+ BACKUP treats the BCS as though it were an entry-sequenced data set (ESDS), bypassing all reference to the index as it backs up the data records in physical sequence. For this type of BCS error, this logic provides greater insurance that all records are backed up.
One final detail should be clarified in the output report in Example 3-3 on page 49. Note that it indicates a return code 4 from the internally-issued IDCAMS EXAMINE. Recall that the “explicitly-issued” EXAMINE, in the output of Example 3-1 on page 48, contained the keyword DATATEST, which causes EXAMINE to actually run both an INDEXTEST and a DATATEST, and it returned a condition code of 8. By default, the CR+ BACKUP command in Example 3-3 on page 49 internally issues an EXAMINE command, with only the INDEXTEST keyword. As you can see from the output in Example 3-1 on page 48, the INDEXTEST logic only considered the error to be minor. The DATATEST logic considered the error to be major. Apparently, this results in a return code of 4 from INDEXTEST and a return code of 8 from DATATEST, and only the highest return code is passed back.

Note: This case study does not imply that IDCAMS EXPORT is likely to produce a “bad” BCS backup and CR+ BACKUP always produces a good one. In other circumstances, it could happen that CR+ BACKUP has problems and EXPORT does not. Instead, this case study illustrates the importance of frequent and regular diagnostics for your entire catalog environment followed by a careful review of the output listings when there is any indication of a problem. It also points out the benefit of using two different catalog backup methods, increasing your chances that your backup is valid when the all-important time comes to restore and recover it.

3.2 Case study #2: BCS backup and forward recovery

This case study demonstrates how to use the CR+ BACKUP and RECOVER commands to forward recover a BCS.

Example 3-4 shows a CR+ BACKUP command of the specific BCS, named UCAT.CRPLUS2, taken at 14:23:11 on 21 September. In a production environment, it might be more appropriate to back up all BCSs in a single step by specifying the keyword as BCS(**).

An IDCAMS EXAMINE, with the INDEXTEST keyword, is specified (which is also the default), and the result of this structural diagnostic test of the BCS index component shows that it has no errors.

Our backup of the BCS then runs, writing a total of 7,838 catalog records to the backup file on the DD statement OUT1. This is set up as a GDG, so we identify it as our newest (+1) generation of the GDG.

Example 3-4   Backing up the BCS with CR+ BACKUP

//YJCRES1A JOB MSGCLASS=X,NOTIFY=YJCRES1,RESTART=*  
//S1 EXEC PGM=CAT00010  
//OUT1 DD DSN=YJCRES1.CRPLUS2.BKUP(+1),DISP=(,CATLG),  
//UNIT=SYSALLDA,SPACE=(CYL,(1,1),RLSE)  
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *  
//BACKUP BCS(UCAT.CRPLUS2) -  
//OUTFILE(OUT1) -  
//EXAMINE(INDEXTEST)  
/*

FAC01001I Catalog RecoveryPlus MAINSTAR SOFTWARE CORPORATION 21 SEP 2001 PAGE:1

CAT02016I THE FOLLOWING DSNS/MASKS ARE TO BE PROCESSED DURING BCS PROCESSING.  
UCAT.CRPLUS2

CAT02031I BASED ON THE MASKS/DSNS ENTERED, THE FOLLOWING CATALOGS HAVE BEEN SELECTED.
Following the BACKUP step, considerable processing continued in the system, including a significant number of DEFINEs, DELETEs, and updates to the data sets in UCAT.CRPLUS2.

On the assumption that you could have a later catastrophic error in the BCS named UCAT.CRPLUS2, you should now prepare for a forward recovery of this BCS. To do this, identify all processing that is currently running through this BCS and take whatever steps are necessary to quiesce that activity. This could mean waiting until all current jobs that use this BCS are completed or taking the more drastic action of canceling them. If you allow currently executing jobs to continue to completion, one or more might not terminate cleanly as a result of errors when data sets cataloged in UCAT.CRPLUS2 are closed. If that occurs, you must be prepared for job recovery. You must also stop any new jobs from initiating using the job scheduler. Note that, if you are allowing currently running jobs to complete, issuing an IDCAMS ALTER LOCK on the BCS definitely results in CLOSE failures when these jobs attempt to complete.

Next, you identify the most recent backup that was taken of this BCS. Because we set up a GDG for our BCS backups, we pointed to the current (0) generation in our recovery JCL (see the INFILE1 DD statement in Example 3-5 on page 52).

Then, take all steps necessary to gather up all of the SMF data that is required for the recovery. In our ITSO test environment, we ran our test jobs on three systems, identified as SC63, SC64, and SC65. When we knew that all processing was quiesced on all three systems that had jobs processing through UCAT.CRPLUS1, we switched from the current SMF log file to the other by issuing the SMF operator command for each of the systems.

The next step is to dump the “switched” SMF data collection file, often considered the most difficult task of the recovery process. In our example, we gathered all of the dumped SMF files from all three systems for the entire time period between the most recent BACKUP and the current time. The frequency of the backups of the BCS, the number of systems that are sharing it, and the quality of procedures in place to gather the dumped records affect how easy or difficult this part of the process is. In our case, when we gathered up all of the necessary SMF data, we created JCL that concatenates every dump file in the SMFINDD statement in our RECOVER job.

Example 3-5 on page 52 shows the complete RECOVER BCS job. The command indicates the BCS that is to be restored, where the backup copy is located, and that we want to perform a forward recovery using SMF data that is included in the SMFINDD JCL statement. As the processing begins, the defaults tell RECOVER that alias records from the backup file must be applied to the current system master catalog and that all SMF records, from the time of backup to the time of this forward recovery, must be applied to the backup records.
The PRINT options message indicates that we did not want a detailed listing of all restored records. If you want to change this, use the PRINT keyword in the RECOVER command. See 2.5.8, “Printing BCS records during a RECOVER” on page 29, for more details.

Example 3-5  BCS forward recovery with CR+ RECOVER

```
//YCJRES1 JOB MSGCLASS=X,NOTIFY=YCJRES1,RESTART=*  
//S1 EXEC PGM=CAT00010  
//INFILE1 DD DSN=YCJRES1.CRPLUS1.BACKUP(0),DISP=SHR  
//SMFINDD DD DSN=YCJRES1.SC63.SMFDATA,DISP=SHR  
//         DD DSN=YCJRES1.SC64.SMFDATA,DISP=SHR  
//         DD DSN=YCJRES1.SC65.SMFDATA,DISP=SHR  
...plus the DD statements for RCVDD1, RCVMRG, RCVSMF, SMFERR, and others
//SYSPRINT DD SYSOUT=*  
//SYSOUT DD SYSOUT=*  
//SYSIN DD *  
RECOVER BCS(UCAT.CRPLUS2) -  
     INFILE(INFILE1) -  
     FORWARD(SMFFILE(SMFINDD))  
/*

CAT01001I Catalog RecoveryPlus MAINSTAR SOFTWARE CORPORATION 21 SEP 2001 PAGE:1

CAT04079I DEFAULT IN EFFECT: ALIAS
CAT04079I DEFAULT IN EFFECT: FROMBACKUPDATE
CAT04079I DEFAULT IN EFFECT: TOCURRENTDATE
CAT04080I PRINT OPTION IN EFFECT: NONE

CAT04029I 15.50.22 RECOVERY STEP: CHECK BCS STATUS - STARTED
CAT04029I 15.50.22 RECOVERY STEP: CHECK BCS STATUS - COMPLETED

CAT04029I 15.50.22 RECOVERY STEP: LOCK BCS - STARTED
CAT04029I 15.50.22 RECOVERY STEP: LOCK BCS - COMPLETED

CAT04029I 15.50.22 RECOVERY STEP: PROCESSING BACKUP - STARTED
CAT04035I RECOVERY STARTED FOR BCS: UCAT.CRPLUS2
     BACKUP DATE/TIME: 2001264/14:23:11

CAT04029I 15.50.24 RECOVERY STEP: PROCESSING BACKUP - COMPLETED

CAT04029I 15.50.24 RECOVERY STEP: PROCESSING SMF - STARTED

CAT04019I PROCESSING SMF RECORDS
     FROM DATE/TIME: 2001264/14:23:11
     TO DATE/TIME: 2001264/15:50:22
CAT04020I SMF RECORDS SELECTED: 1689
CAT04021I SMF RECORDS FOR SYSSID: SC63
     EARLIEST: 2001263/16:21:47
     LATEST: 2001264/15:19:41
CAT04021I SMF RECORDS FOR SYSSID: SC64
     EARLIEST: 2001263/16:21:47
     LATEST: 2001264/15:19:30
CAT04021I SMF RECORDS FOR SYSSID: SC65
     EARLIEST: 2001263/23:30:00
     LATEST: 2001264/14:49:58
CAT04026W ******** GAP IN TIME FOR SYSSID: SC63 LATEST TIME GAP
CAT04026W ******** GAP IN TIME FOR SYSSID: SC64 LATEST TIME GAP
CAT04026W ******** GAP IN TIME FOR SYSSID: SC65 LATEST TIME GAP

CAT04027I RECORD COUNTS FROM APPLYING SMF DATA
     7580 UNCHANGED; 4 DELETED; 250 INSERTED; 4 UPDATED

CAT04029I 15.50.26 RECOVERY STEP: PROCESSING SMF - COMPLETED

CAT04029I 15.50.26 RECOVERY STEP: REWRITE TO BCS - STARTED
CAT04029I 15.50.35 RECOVERY STEP: REWRITE TO BCS - COMPLETED

CAT04029I 15.50.35 RECOVERY STEP: UNLOCK OF BCS - STARTED
```
The RECOVER output is essentially a process log of all steps that the command performs:

- **CHECK BCS STATUS**
  
  In this process, RECOVER determines if a BCS that is named UCAT.CRPLUS2 is connected to the master catalog and physically exists in the system. If one is found, a DELETE (with RECOVERY) and DEFINE USERCATALOG command is submitted “under the covers” to IDCAMS, so that the BCS into which the backup records is written is a brand new, empty BCS (see 2.5.3, “Explicit/implicit DEFINE of the BCS or VVDS to restore” on page 24, if you want to perform your own DELETE/DEFINE command in preparation for RECOVER).

- **LOCK BCS**
  
  An IDCAMS ALTER command is issued for the BCS that is named UCAT.CRPLUS2, with the keyword LOCK specified. This makes the BCS inaccessible to all users, including users sharing the catalog on other systems. The LOCK is removed as one of the last steps in RECOVER command processing.

- **PROCESSING BACKUP ... RECOVERY STARTED FOR BCS: UCAT.CRPLUS2**
  
  This process reads the backup file. If there were multiple BCS backups in this file, it locates the backup records for UCAT.CRPLUS2. When they are located, it identifies the date and time of the backup, for use in the SMF record extract, and the job is positioned and ready to restore and forward recover.

- **PROCESSING SMF**
  
  This step performs the forward recovery with the SMF data, including the extraction of all SMF records between the backup and restore times, sorting the records into ascending time sequence, and then applying all updates to the backup records. If it detects gaps of more than 10 minutes between SMF update records for the BCS, it prints an informational message to alert you to the possibility that dumped SMF records from one or more systems might not be included in the process.

- **REWRITE TO BCS**
  
  This step writes the updated (forward recovered) BCS records to the output BCS.

- **UNLOCK OF BCS**
  
  An IDCAMS ALTER for the BCS, with UNLOCK, is now issued.

- **APPLYING ALIASES**
  
  All aliases for the BCS are then restored to the master catalog for the system on which the RECOVER command is run.

- **RECOVERY COMPLETED FOR BCS: UCAT.CRPLUS2 ... RETURN CODE: 4**
  
  Indicates the successful completion of the forward recovery. The return code value of 4 in this instance is due to the gap in SMF records.
After the forward recovery of the BCS that is named UCAT.CRPLUS2 completes, it is a good idea to run the following clean-up steps:

1. Run IDCAMS EXAMINE INDEXTEST DATATEST to ensure there are not any structural problems with either the index or data component of the newly recovered BCS.

2. Run IDCAMS DIAGNOSE ICFCATALOG without the COMPAREDD operand to ensure that the BCS does not contain any data content errors.

3. Depending on how quickly this BCS needs to be back in production, it might also be prudent to run the two CR+ DIAGNOSE commands (Example 3-6) to ensure there are no errors between the BCS and its related VVDSs (see 2.6.2, “DIAGNOSE BCS-VVDS and VOLUME-BCS commands” on page 34, for more details of how to code these commands).

Example 3-6 CR+ DIAGNOSE commands

```
DIAGNOSE BCS-VVDS -
  COMPAREBCS(UCAT.CRPLUS2) -
  ALLRELATED -
  JOURNAL(YCJRES1.CRPLUS.DIAGNOSE.JOURNAL) -
  FIXDATASET(YCJRES1.CRPLUS.DIAGNOSE.FIXFILE)

DIAGNOSE VVDS-BCS -
  COMPAREVVDS(SBOX*) -
  TOBCS(UCAT.CRPLUS2) -
  FIXDATASET(YCJRES1.CRPLUS.DIAGNOSE.FIXFILE)
```

4. Run a CR+ DIAGNOSE ALIAS command to ensure that all aliases related to this newly restored user catalog are consistent across all system master catalogs. The next section shows you how to code this command and how to interpret the output and fix file created with this command. If we had run the forward recovery on system SC63, we would have had to run the DIAGNOSE ALIAS command twice, the first time specifying SC63 and SC64 as our two master catalogs, and the second time specifying SC63 and SG65 as our two master catalogs. Presumably, you would run this in NONPEER mode.

If you fully test and practice the steps in this case study regularly, the whole process should not take more than a few minutes. If you do not test them and you do not have good procedures for tasks such as collecting the required SMF data, it could be a long and painful process. To accomplish this testing, the RECOVER command that is illustrated in this case study could be run at any time with the SIMULATE keyword.

### 3.3 Case study #3: Synchronizing aliases

This case study demonstrates how to use the Catalog RecoveryPlus DIAGNOSE ALIAS command to synchronize the user catalog aliases in two master catalogs.

Example 3-7 on page 55 shows the CR+ DIAGNOSE ALIAS control statement, identifying the two master catalogs for comparison in the COMPARE-MASTERCATALOG keyword. We specified the NONPEER keyword, so the first master catalog specified was considered “superior” to the master catalog that was specified second. To state that another way, the first-specified master catalog was considered to be the “control catalog,” so alias entries in it were compared against the other to help us synchronize the second master catalog to the first. The FIXDATASET keyword indicates that IDCAMS “fix” commands are to be created for each identified discrepancy and stored in the data set that is specified by the FIXDD statement.
Example 3-7  DIAGNOSE ALIAS master catalog job stream

```
//YCJRESIN JOB MSGCLASS=X,NOTIFY=YCJRES1,RESTART=* 
//S2 EXEC PGM=CAT00010
//FIXDD    DD DSN=YCJRES1.CRPLUS.DIAG ALIAS.FIXFILE,DISP=(,CATLG), 
//         UNIT=SYSALLDA,SPACE=(CYL,(1,1),RLSE), 
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//SYSPRINT DD SYSOUT=* 
//SYSOUT DD SYSOUT=* 
//SYSIN DD *
```

```
DIAGNOSE ALIAS - 
    COMPARE-MASTERCATALOG(MCAT.SANDBOX.R10 MCAT.SANDBOX.Z02) - 
    FIXDATASET(YCJRES1.CRPLUS.DIAG.ALIAS.FIXFILE) NONPEER 
```

Refer to 2.6.1, “DIAGNOSE ALIAS command” on page 32, for specific details of what this command is used for and how it works.

Example 3-8 shows the output report from DIAGNOSE ALIAS. The first part of the output shows statistics from each master catalog, including the total number of records, the number of aliases, and the number of connected user catalogs.

```
Example 3-8  DIAGNOSE ALIAS master catalog compare report
```

```
CAT01001I Catalog RecoveryPlus MAINSTAR SOFTWARE CORPORATION 24 SEP 2001 PAGE:1 
CAT11270I CATALOG MCAT.SANDBOX.R10 WILL BE COMARED TO MCAT.SANDBOX.Z02 
CAT11272I CATALOG MCAT.SANDBOX.R10 WILL BE TREATED AS SUPERIOR TO CATALOG 
    MCAT.SANDBOX.Z02 
    THAT IS, DEFINE/DELETE ALIAS COMMANDS WILL BE CREATED FOR ENTRIES 
    MISSING FROM MCAT.SANDBOX.Z02 BUT PRESENT IN MCAT.SANDBOX.R10 
CAT11221I EXTRACTING USERCATALOG AND ALIAS INFORMATION FROM MCAT.SANDBOX.R10 
CAT11221I MCAT MAXIMUM RECORD LENGTH: 32,400 
CAT11222I 5,811 RECORDS IN CATALOG 
CAT11222I 328 ALIAS RECORDS FOR USERCATALOGS 
CAT11222I 31 USERCATALOG RECORDS 
CAT11223I THE FOLLOWING CATALOGS IN MCAT.SANDBOX.R10 HAVE NO ALIAS NAMES 
CAT11223I  MCAT.OS3R6V01.VTOTCAT 
CAT11223I  MCAT.SANDBOX.R9.VSBOX11 
CAT11223I  MCAT.SANDBOX.VSBOX11 
CAT11223I  MCAT.SBOXALT.SBOX02 
CAT11223I  SYS1.VOLCAT.VGENERAL 
CAT11223I  UCAT.CRPLUS5 
CAT11221I EXTRACTING USERCATALOG AND ALIAS INFORMATION FROM MCAT.SANDBOX.Z02 
CAT11221I MCAT MAXIMUM RECORD LENGTH: 32,400 
CAT11222I 4,860 RECORDS IN CATALOG 
CAT11222I 324 ALIAS RECORDS FOR USERCATALOGS 
CAT11222I 30 USERCATALOG RECORDS 
CAT11223I THE FOLLOWING CATALOGS IN MCAT.SANDBOX.Z02 HAVE NO ALIAS NAMES 
CAT11223I  MCAT.OS3R6V01.VTOTCAT 
CAT11223I  MCAT.SANDBOX.R10.VSBOX11 
CAT11223I  MCAT.SANDBOX.R9.VSBOX11 
CAT11223I  MCAT.SANDBOX.VSBOX11 
CAT11223I  MCAT.SBOXALT.SBOX02 
CAT11223I  SYS1.VOLCAT.VGENERAL 
CAT11221I BEGINNING USERCATALOG COMPARISON 
CAT11231I COMPARE-MASTERCATALOG MCAT.SANDBOX.R10 FOUND IN MCAT.SANDBOX.Z02 
CAT11231I COMPARE-MASTERCATALOG MCAT.SANDBOX.Z02 FOUND IN MCAT.SANDBOX.R10 
CAT11232W UCAT UCAT.CRPLUS5 IN MCAT.SANDBOX.R10 BUT NOT IN MCAT.SANDBOX.Z02 
```
It also lists any user catalogs that do not have any aliases related to them. Effectively, these would be unusable user catalogs, since it is not possible to direct cataloging of data sets to them without an alias. This is not considered to be an error, so a fix is not created for this situation. Instead an informational message alerts you to the situation.

Next, the command processor begins master catalog comparison. For each error situation identified, a fix is created and stored in the fix data set. There is a corresponding reference number for the correlation of each error message to its fix. Example 3-9 shows the fixes that were created when we ran this command for the case study. Immediately after the “BEGINNING...” message, note that the output report comments that each master catalog is found to be connected as a user catalog in the other. This cross-connection of master catalogs is a common practice. Using this information, you can perform maintenance on a master catalog from the other system, where you then treat the master catalog as though it were a user catalog.

Example 3-9  DIAGNOSE ALIAS master catalog generated fixes

```cpp
/* CATALOG RECOVERYPLUS FIX LIST FROM DIAGNOSE ALIAS OF */ /* CATALOG MCAT.SANDBOX.R10.VSBOX11 AND MCAT.SANDBOX.Z02 */
/* CATALOG RECOVERYPLUS FIX REFERENCE 1 */ /* IMPORT CONNECT OBJECTS - (( UCAT.CRPLUS5 - DEVICETYPE ( 3390 ) - VOLUMES ( SBOX02 ) )) - CATALOG ( MCAT.SANDBOX.Z02 ) */
/* CATALOG RECOVERYPLUS FIX REFERENCE 2 */ /* DEFINE ALIAS ( NAME ( ALA110 ) - RELATE ( UCAT.VSBOX09 ) ) - CATALOG ( MCAT.SANDBOX.Z02 ) */
/* CATALOG RECOVERYPLUS FIX REFERENCE 3 */ /* DEFINE ALIAS ( NAME ( AUO110 ) - RELATE ( UCAT.VSBOX09 ) ) - CATALOG ( MCAT.SANDBOX.Z02 ) */
/* CATALOG RECOVERYPLUS FIX REFERENCE 4*/ /* DELETE ICSF - ALIAS - CATALOG ( MCAT.SANDBOX.R10 ) */
/* DEFINE ALIAS ( NAME ( ICSF ) - RELATE ( UCAT.VSBOX09 ) ) */
```
Fix #1 indicates a user catalog, UCAT.CRPLUS5, that was found to be connected in MCAT.SANDBOX.R10, but not in MCAT.SANDBOX.Z02. The generated fix is an IMPORT CONNECT command for that user catalog in MCAT.SANDBOX.Z02.

Fix #2 and #3 indicate aliases that were in MCAT.SANDBOX.R10, but were not found in MCAT.SANDBOX.Z02. The generated fix is a DEFINE ALIAS command.

Fix #4 is more complicated. In that instance, the alias ICSF was found in each master catalog. In one master, it was related to the user catalog UCAT.VSBOX09. In the other master, it was related to the user catalog UCAT.VSBOX01. This is considered to be an error situation, but because there is no way for the DEFINE ALIAS command processor to determine which is correct, it builds two sets of commands that you can choose from, after you determine which fix is correct for your situation.

All the fixes in Example 3-9 on page 56 were generated as comment statements, and as such, were not directly executable as IDCAMS commands until the /* at the beginning of each command was removed. Because the fix file is a physical sequential file, you can remove the /* with TSO ISPF edit. In many situations, this might be a laborious task, so the DIAGNOSE ALIAS command has the keywords DEFINE-ALIAS and DELETE-ALIAS. If you know in advance which command is more appropriate for your environment, the command processor will build those commands as “ready to execute.”

3.4 Case study #4: VVDS backup and forward recovery

This case study demonstrates how to use the CR+ BACKUP and RECOVER VVDS command for two tasks:

- Producing backup copies for VVDSs on critical volumes
- Forward recovering a VVDS if it “breaks.”

The following examples do not show an actual problem with the VVDS being recovered, but rather, the job streams that perform VVDS backup and recovery.

It is frequently asked, “How often do VVDSs break?” The answer is “more often than most people realize.” The DIAGNOSE VVDS command is rarely used and as a result, any errors encountered are probably credited to other causes.
Interestingly, during system testing for this paper, we discovered a broken VVDS on one of our ITSO volumes. Because other IBM Redbook projects were using this volume heavily and the broken VVDS was only causing intermittent failures during data set allocation, its recovery was left for a later time. In this case study, we demonstrate the steps that would have been used in our recovery in the ITSO system, assuming it was a VVDS failure on volume SBOX79.

In Example 3-10, the VVDSs on all of the SBOXxx volumes were backed up with a single CR+ BACKUP command and the backup records for all VVDSs were written to a single output file identified by the OUT1 DD statement. In a production environment, this would presumably be a regularly scheduled job, possibly run at the same time as all BCSs are backed up.

Example 3-10  BACKUP VVDS job stream and output listing

```
20.11.15 JOB10237  IEF403I YCJRES1Z - STARTED - ASID=0019
20.13.12 JOB10237  IEF404I YCJRES1Z - ENDED - ASID=0019
20.13.12 JOB10237  ---TIMINGS (MINS.)---
20.13.12 JOB10237  -JOBNAME  STEPNAME RC   EXCP    CPU    SRB  CLOCK
20.13.12 JOB10237  -YCJRES1Z S1  04 33667 .11 .01 1.9

//YCJRES1Z JOB MSGCLASS=x,NOTIFY=YCJRES1,RESTART=*  
//S2 EXEC PGM=CAT00010  
//OUT1 DD DSN=YCJRES1.BACKUP(+1),DISP=(,CATLG),  
//       UNIT=SYSALLDA,SPACE=(CYL,(10,10),RLSE)  
//SYSPRINT DD SYSOUT=*  
//SYSOUT DD SYSOUT=*  
//SYSIN DD *  
BACKUP VVDS(SBOX*) -  
    OUTFILE(OUT1)  
/*

Output Listing:  
CAT01001I Catalog RecoveryPlus MAINSTAR SOFTWARE CORPORATION 24 SEP 2001 PAGE:1

CAT02061W VVDS SYS1.VVDS.VSBOX07 IS NOT ON THE VOLUME  
CAT02061W VVDS SYS1.VVDS.VSBOX34 IS NOT ON THE VOLUME  
... and 29 more volumes where VVDSs were not found on the volume,  
... followed by a list of 66 volumes that were selected for the BACKUP

CAT02045I START OF DATA UNLOAD FOR SYS1.VVDS.VSBOX01 ON 24 SEP 2001 AT 20.11.16  
CAT02092I CALLING IDCAMS TO PERFORM DIAGNOSE VVDS  
IDC11367I THE FOLLOWING VVDS REFERENCED CATALOGS WERE NOT ENCOUNTERED:  
CATALOG.TOTICF1.VTOTCAT  
CAT02050W IDCAMS (8) ISSUED RETURN CODE 4  
CAT02046I RECORD SUMMARY FOR SYS1.VVDS.VSBOX01  
    PRIMARY SECONDARY NONVSAM VVCR/VVCN OTHER TOTAL TOTAL BYTES  
    295 39 0 1 0 335 179,560  
CAT02047I END OF DATA UNLOAD FOR SYS1.VVDS.VSBOX01 ON 24 SEP 2001 AT 20.11.48  

... additional volume unloads removed from listing

CAT02045I START OF DATA UNLOAD FOR SYS1.VVDS.VSBOX79 ON 24 SEP 2001 AT 20.11.51  
CAT02092I CALLING IDCAMS TO PERFORM DIAGNOSE VVDS  
CAT02046I RECORD SUMMARY FOR SYS1.VVDS.VSBOX79  
    PRIMARY SECONDARY NONVSAM VVCR/VVCN OTHER TOTAL TOTAL BYTES  
    1,032 0 0 1 0 1,033 378,834  
CAT02047I END OF DATA UNLOAD FOR SYS1.VVDS.VSBOX79 ON 24 SEP 2001 AT 20.11.57  

CAT02009I BACKUP FUNCTION COMPLETE. RETURN CODE 4  
CAT01009I Catalog RecoveryPlus EXECUTION COMPLETE. HIGHEST RETURN CODE WAS 4
```
There were a total of 95 SBOXxx volumes, as identified by the mask SBOX*, of which 29 were found to be non-system-managed, without any VSAM on them. Therefore, they did not contain a VVDS. A total of 66 volumes had VVDSs and were backed up. As part of BACKUP command processing, an IDCAMS DIAGNOSE VVDS command was first issued for each volume, followed by a data unload of all records from the VVDS.

As you can see from the job timings message, the BACKUP of all 66 VVDS required 33,667 EXCPs and 1.9 minutes of elapsed time to run. In a subsequent test to determine what the cost increase would be if the backup were duplexed, the EXCP cost increased by only 259, and the elapsed time remained at 1.9 seconds.

Subsequent to the backup, our case study assumes that SBOX79 has an internal error, necessitating a recovery from the most recent backup. The RECOVER command job stream in Example 3-11 illustrates forward recovery for the VVDS on this volume, specifying that the SMF data from all systems that share this volume are concatenated on the DD statement, SMFINDD.

Example 3-11  RECOVER VVDS job stream

```
//YCJRES1N JOB MSGCLASS=x,NOTIFY=YCJRES1,RESTART=*
//S2 EXEC PGM=CAT00010
//INFILE1 DD DSN=YCJRES1.VVDS.BACKUP,DISP=SHR
//SMFINDD DD DSN=YCJRES1.SC63.SMFDATA,DISP=SHR
// DD DSN=YCJRES1.SC64.SMFDATA,DISP=SHR
// DD DSN=YCJRES1.SC65.SMFDATA,DISP=SHR
//SYSPRINT DD SYSOUT=*  
//SYSUDUMP DD SYSOUT=*  
//SYSOUT DD SYSOUT=*  
//SYSIN DD *
RECOVER VVDS(SBOX79) -
   INFILE(INFILE1) -
   FORWARD(SMFFILE(SMFINDD))
```

The output from the VVDS forward recovery can be found in the listing report in Example 3-12. After first sorting the SMF records, RECOVER deletes and defines the original VVDS, then writes the new, updated records into the new VVDS. Notice that it also identified one or more VVR/NVR records that “back-pointed” to three BCSs whose names were not in the VVCR record, so these BCS names were added.

Example 3-12  RECOVER VVDS job stream and output listing

```
CAT01001I Catalog RecoveryPlus MAINSTAR SOFTWARE CORPORATION 24 SEP 2001 PAGE:1
CAT04079I DEFAULT IN EFFECT: FROMBACKUPDATE
CAT04079I DEFAULT IN EFFECT: TOCURRENTDATE
CAT04080I PRINT OPTION IN EFFECT: NONE
CAT04029I 20.32.26 RECOVERY STEP: READ BACKUP - STARTED
CAT04058I RECOVERY STARTED FOR VVDS: SYS1.VVDS.VSBOX79
BACKUP DATE/TIME: 2001267/20:11:16
CAT04030I RECORD SUMMARY FOR SYS1.VVDS.VSBOX79 RECORDS FROM BACKUP

<table>
<thead>
<tr>
<th>PRIMARY</th>
<th>SECONDARY</th>
<th>NONVSAM</th>
<th>VVCR/VVCN</th>
<th>OTHER</th>
<th>TOTAL</th>
<th>TOTAL BYTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,012</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1,013</td>
<td>371,856</td>
</tr>
</tbody>
</table>

CAT04029I 20.32.26 RECOVERY STEP: READ BACKUP - COMPLETED

CAT04029I 20.32.26 RECOVERY STEP: READ VTOC - STARTED
CAT04029I 20.32.29 RECOVERY STEP: READ VTOC - COMPLETED

CAT04029I 20.32.29 RECOVERY STEP: SORT INPUT RECORDS - STARTED
CAT04029I 20.32.30 RECOVERY STEP: SORT INPUT RECORDS - COMPLETED

CAT04029I 20.32.30 RECOVERY STEP: PROCESSING SMF - STARTED
CAT04019I PROCESSING SMF RECORDS FROM DATE/TIME: 2001267/20:11:16
```
3.5 Case study #5: Enlarging VVDS

This case study demonstrates how to use the CR+ BACKUP VVDS and RECOVER VVDS commands to enlarge a VVDS.

In this example, the VVDS on volume VSBOX36 was allocated using the default allocation size of TRK(10 10). For today's volumes, where thousands of small data sets could be allocated, you might fill up the VVDS and put it into secondary extents (which are not necessarily a problem, but it is one more complicating factor for your volumes that can potentially lead to problems).

Example 3-13 illustrates the BACKUP job stream and output that creates the backup copy of the VVDS, in preparation for recovery.

Example 3-13  BACKUP VVDS job stream and output listing

```bash
//YCJRESIZ JOB MSGCLASS=X,NOTIFY=YCJRES1,RESTART=*
//S1 EXEC PGM=CAT00010
//OUT1 DD DSN=YCJRES1.CRPLUS.VVDS.SBOX36.BKUP4,DISP=(,CATLG),
//     UNIT=SYSALLDA,SPACE=(CYL,(1,1),RLSE)
//SYSPRINT DD SYSOUT=* 
//SYSUDUMP DD SYSOUT=* 
//SYSIN DD *
// BACKUP VVDS(SBOX36) -
//   OUTFILE(OUT1)
CAT01009I Catalog RecoveryPlus MAINSTAR SOFTWARE CORPORATION 28 SEP 2001 PAGE:1
CAT02045I START OF DATA UNLOAD FOR SYS1.VVDS.VSBOX36 ON 28 SEP 2001 AT 12.44.32
```
Example 3-14 illustrates the IDCAMS DELETE and DEFINE CLUSTER commands that you then run. The RECOVERY keyword tells IDCAMS to physically delete the VVDS from the volume, without deleting any of the actual data sets on the volume. The FILE keyword is necessary for this operation, because the MVS allocation routines use it for the “locate” operation of the VVDS on the volume. The DEFINE CLUSTER command that follows reallocates a new VVDS on the volume, at the size of CYL(2 2).

Example 3-14  IDCAMS DELETE/DEFINE job stream and output listing

```
YCJRES1B JOB MSGCLASS=X,NOTIFY=UCJRES1,RESTART=*  
//S EXEC PGM=IDCAMS  
//SYSPRINT DD SYSOUT=*  
//SBOX36 DD VOL=SER=SBOX36,UNIT=3390,DISP=SHR  
//SYSIN DD *  
  DELETE SYS1.VVDS.VSBOX36 -  
  FILE(SBOX36) -  
  RECOVERY  
IDC0550I ENTRY (C) SYS1.VVDS.VSBOX36 DELETED  
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0  
DEFINE CLUSTER(NAME(SYS1.VVDS.VSBOX36) -  
  VOL(SBOX36) -  
  FILE(SBOX36) -  
  NONINDEXED -  
  CYL(2 2))  
IDC0508I DATA ALLOCATION STATUS FOR VOLUME SBOX36 IS 0  
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0  
IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

Example 3-15 shows the next step, which runs a CR+ RECOVER VVDS command with the INTOEMPTY keyword to alert the command processor that a DELETE and DEFINE for the VVDS is already accomplished, and that it is to look for an empty VVDS on the volume for restore processing. This RECOVER command only does a restore of the backup records and does not perform a forward recovery because the FORWARD keyword is not specified. This is a valid situation, because the time between the BACKUP and RECOVER commands was minimal, without any volume data set accesses or updates. As part of its normal processing, RECOVER sorted the VVR/NVR records prior to restoring them, so that any duplicate VVR/NVR records could be detected. The “read VTOC” messages indicate that the command is checking to ensure that all VVDS records have associated VTOC records.

Example 3-15  RECOVER VVDS job stream and output listing

```
//YCJRESIK JOB MSGCLASS=X,NOTIFY=UCJRES1,RESTART=*  
//S EXEC PGM=CAT00010  
//INFILE1 DD DSN=YCJRES1.CRPLUS.VVDS.SBOX36.BKUP3,DISP=SHR  
//SYSPRINT DD SYOUT=*  
//SYOUT DD SYOUT=*  
//SYSIN DD *  
CAT0001I Catalog RecoveryPlus MAINSTAR SOFTWARE CORPORATION 28 SEP 2001 PAGE:1  
RECOVER VVDS(SBOX36) -  
  INFILE(INFILE1) -  
  INTOEMPTY
```
CASE STUDY #6: CHANGING A DISK VOLSER

This case study demonstrates how to use the CR+ SUPERCLIP command.

Without CR+ SUPERCLIP, standard system utilities require you to:

1. Unload all of the data sets from the volume.
2. Delete (for recovery) any catalogs that reside on the volume.
3. VARY the volume offline.
4. Run the utility ICKDSF to "clip" the volser.
5. VARY the volume online.
6. Restore any catalogs that previously resided on the volume.
7. Reload all of the data sets for the volume.

Depending on how many data sets are on the volume, this could take a long time and is the primary reason the process is not performed very often.

With CR+ SUPERCLIP, you simply specify the old volser and the new volser, and the command processor internally performs all of the functions necessary to change the volser.

Example 3-16 on page 63 illustrates how we used the SUPERCLIP command to change volser CRPL99 to CRPL00. The volume currently had data sets physically allocated on it. There was also a BCS on the volume, UCAT.CRPLUS1. Since its connector (U-type) record in the master catalog "pointed" to it by volser, this volser value was also changed.

The JCL and commands in this example were generated by the CR+ ISPF dialogs. The IDCAMS step that allocated the journal file is included because the Generate DEFINE option was selected in the SUPERCLIP panel.
The output in the example is step-by-step process log of all functions that the SUPERCLIP command provides, including the MVS MODIFY commands to “unallocate” the catalog, and VARY OFFLINE/ONLINE commands for running the ICKDSF program.

Example 3-16 SUPERCLIP job stream and output listing

```
12.04.20 JOB26049 IEF403I MHLRES1T - STARTED - TIME=12.04.20 - ASID=0038
12.04.20 JOB26049 - --TIMINGS (MINS)--
12.04.20 JOB26049 -JOBNAME STEPNAME RC EXCP CPU SRB CLOCK
12.04.20 JOB26049 -MHLRES1T DEFINE 00 116 .00 .00 .00
12.04.22 JOB26049 IEA630I OPERATOR SCLIP001 NOW ACTIVE, SYSTEM=SC64
12.04.22 JOB26049 F CATALOG,UNALLOCATE(UCAT.CRPLUS1)
12.04.38 JOB26049 IEE302I D417 ONLINE BY CAT00140
12.04.38 JOB26049 IEA631I OPERATOR SCLIP001 NOW INACTIVE, SYSTEM=SC64
12.04.39 JOB26049 -MHLRES1T CRPLUS 00 1675 .00 .00 .30
12.04.39 JOB26049 IEF404I MHLRES1T - ENDED - TIME=12.04.39 - ASID=0038

//MHLRES1T JOB CLASS=A,MSGCLASS=X,NOTIFY=MHLRES1
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *
DELETE MHLRES1.SCLIP.JOURNAL PURGE  
SET MAXCC = 0
DEFINE CLUSTER(NAME(MHLRES1.SCLIP.JOURNAL) -  
CYLINDERS(10 10) KEYS(133 0)  
RECORDSIZE(242 1442)) -  
DATA(CONTROLINTERVALSIZE(4096)) -  
INDEX(CONTROLINTERVALSIZE(2048))  
//CRPLUS EXEC PGM=CAT00010,REGION=0M  
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *
SUPERCLIP NEW-VOLSER(CRPL00) -  
OLD-VOLSER(CRPL99) -  
JOURNAL(MHLRES1.SCLIP.JOURNAL) -  
UNCATALOGED-DSN(RC(4),CONTINUE) */

... IDCAMS SYSPRINT deleted

CAT01001I Catalog RecoveryPlus MAINSTAR SOFTWARE CORPORATION 07 SEP 2006
CAT14021I SUPERCLIP CLIP started
CAT14021I DD=SYS00001 allocated OLD for DSN=MHLRES1.SCLIP.JOURNAL

Discover Volume Phase

CAT14082I Storage manager is HSM
CAT14081I Volume CRPL99 is not SMS managed
CAT14081I Volume CRPL00 is not SMS managed

CAT14360I Target VOLSER=CRPL99 ADDR=D417 device status follows:
  Current system name ............ SC64
  UCB device address ............... D417
  Mount status .................... online
  SYSZVVSDE reserve status ...... not owned *
  SYSVTOC reserve status ........ owned Excl
  Device type ..................... 3030200F
  Unit name ...................... 3390
  Cached controller .............. yes
  Cylinder capacity .............. 10017
  Tracks per cylinder .......... 15
  VTOC-DSCB per track .......... 50
  VTOCIX-C/I per track .......... 21
  VVDS-C/I per track ............ 12
  VTOC location ................ 0005.0000.01

CAT14361I Dump statistics follow:
```
Dump start time ................... 09/07/2006 12:04:21.26
end time ......................... 09/07/2006 12:04:22.03
elapsed ..................                   0.76 seconds
Volume serial number .......... CRPL99
UCB device address .......... D417
Cached controller .......... yes
Current system name .......... SC64
Dataspase allocation .......... 2097148-K
Dataspase used .......... 397-K
Total number of records ...... 10380
Starting CCHHR for VTOC ...... 0005.0000.01
Total DSCB count .......... 9000
Number of DSCBs in use .......... 98
Number of free DSCBs .......... 8902
Format-1 DSCB count .......... 96
Format-2 DSCB count .......... 0
Format-3 DSCB count .......... 0
Format-4 DSCB count .......... 1
Format-5 DSCB count .......... 1
VTOC-Index DSN .......... SYS1.VTOCIX.CRPL99
Starting CCHHR for VTOCIX ... 0001.0000.01
VTOCIX extents ................ 1
Total VIR count .......... 1260
Number of active VIRs .......... 15
Number of free VIRs .......... 1245
VVDS dataset name .......... SYS1.VVDS.VCRPL99
Starting CCHHR for VVDS ...... 0000.0001.01
VVDS extents .................. 1
Total VVR count .......... 120
Number of active VVRs .......... 7
Number of free VVRs .......... 113
CAT14335I 10380 Records dumped

Discover Volume Phase
Catalogs on volume
UCAT.CRPLUS1

Create Journal Phase
CAT14020I 09/07 12:04:22.03 SUPERCLIP step: Create journal records
CAT14020I 09/07 12:04:22.06 SUPERCLIP step: Save UCAT/ALIAS in Journal - started
CAT14020I 09/07 12:04:22.17 SUPERCLIP step: Save UCAT/ALIAS in Journal - ended

Catalog connectors/aliases
Master catalog/User catalog/Alias
MCAT.SANDBOX.Z17.SBOX00
UCAT.CRPLUS1
CRPB
CRPI
CRP6

Catalogs to be altered
MCAT.SANDBOX.Z17.SBOX00
UCAT.CRPLUS1 ** on OLDVOLSER **
UCAT.CRPLUS2
UCAT.CRPLUS3
UCAT.CRPLUS4
UCAT.CRPLUS5
Datasets by Catalog

<table>
<thead>
<tr>
<th>Catalog</th>
<th>F1 DSCB name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Not cataloged</strong></td>
<td></td>
</tr>
<tr>
<td>SYS1.VTOCIX.CRPL99</td>
<td>SYS1.VTOCIX.CRPL99</td>
</tr>
<tr>
<td>CRPL99</td>
<td></td>
</tr>
<tr>
<td>SYS1.VVDS.VCRPL99</td>
<td>SYS1.VVDS.VCRPL99</td>
</tr>
<tr>
<td>CRPL99</td>
<td></td>
</tr>
<tr>
<td>MCAT.SANDBOX.Z17.SBOX00</td>
<td>UCAT.CRPLUS1</td>
</tr>
<tr>
<td>CRPL99</td>
<td></td>
</tr>
<tr>
<td>UCAT.CRPLUS1</td>
<td>UCAT.CRPLUS1.CATINDEX</td>
</tr>
<tr>
<td>CRPL99</td>
<td></td>
</tr>
<tr>
<td>UCAT.CRPLUS1</td>
<td>CRP1.DSNCO.DATA</td>
</tr>
<tr>
<td>CRPL99</td>
<td></td>
</tr>
<tr>
<td>UCAT.CRPLUS1</td>
<td>CRP1.DSNC1.DATA</td>
</tr>
<tr>
<td>CRPL99</td>
<td></td>
</tr>
<tr>
<td>UCAT.CRPLUS1</td>
<td>CRP1.DSNF0.DATA</td>
</tr>
<tr>
<td>CRPL99</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-----------lines deleted---------------</td>
</tr>
<tr>
<td>UCAT.CRPLUS2</td>
<td>CRP2.DSNA0.DATA</td>
</tr>
<tr>
<td>CRPL99</td>
<td></td>
</tr>
<tr>
<td>UCAT.CRPLUS2</td>
<td>CRP2.DSNA1.DATA</td>
</tr>
<tr>
<td>CRPL99</td>
<td></td>
</tr>
<tr>
<td>UCAT.CRPLUS2</td>
<td>CRP2.DSND0 DATA</td>
</tr>
<tr>
<td>CRPL99</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-----------lines deleted---------------</td>
</tr>
</tbody>
</table>

Volumes to be altered

CRPL99

Create Journal Phase

CAT14043I Number of datasets on OLDVOLSER: 96
CAT14020I 09/07 12:04:22.74 SUPERCLIP step: Create journal records
CAT14020I 09/07 12:04:22.74 SUPERCLIP step: RESTART

Modification Phase

CAT14092I 09/07 12:04:22.88 Console SCLIP001 acquired for UNALLOCATE of
CAT14091I Issue operator command : F CATALOG,UNALLOCATE(UCAT.CRPLUS1)
CAT14003I Ignoring console response: F CATALOG,UNALLOCATE(UCAT.CRPLUS1)
CAT14003I Ignoring console response: IEC351I CATALOG ADDRESS SPACE MODIFY COMMAND ACTIVE
CAT14007I CAS Command successful   : IEC352I CATALOG ADDRESS SPACE MODIFY COMMAND COMPLETED
CAT14091I Operation : VARY OFFLINE
CAT14001I 09/07 12:04:37.91 Device is now OFFLINE

CAT14020I 09/07 12:04:38.16 SUPERCLIP step: BACKOUT - enabled
CAT14020I 09/07 12:04:38.16 SUPERCLIP step: Update external F1 DSCBS - not required
CAT14020I 09/07 12:04:38.16 SUPERCLIP step: Update external catalogs - started

CAT14024I DD=SYS00008 allocated SHR for DSN=MCAT.SANDBOX.Z17.SBOX00
CAT14046I DD=SYS00008 freed DSN=MCAT.SANDBOX.Z17.SBOX00
CAT14024I DD=SYS00009 allocated SHR for DSN=UCAT.CRPLUS2
CAT14046I DD=SYS00009 freed DSN=UCAT.CRPLUS2
CAT14024I DD=SYS00010 allocated SHR for DSN=UCAT.CRPLUS3
CAT14046I DD=SYS00010 freed DSN=UCAT.CRPLUS3
CAT14024I DD=SYS00011 allocated SHR for DSN=UCAT.CRPLUS4
CAT14046I DD=SYS00011 freed DSN=UCAT.CRPLUS4
CAT14024I DD=SYS00012 allocated SHR for DSN=UCAT.CRPLUS5
CAT14046I DD=SYS00012 freed DSN=UCAT.CRPLUS5

CAT14020I 09/07 12:04:38.16 SUPERCLIP step: Update external catalogs - completed
As the example shows, the job began at 12:04:20 and ended at 12:04:39, including 1,675 I/Os. Most of the time and I/O was used to update the volume cell in the associated catalog record for each of the data sets. SUPERCLIP, as part of its processing, does a Catalog Search Interface (CSI) LOCATE for every data set on the volume to identify the BCSs where they are cataloged. It then updates the volser pointer in those catalog records to CRPL00. Also, for the BCS that is housed on the volume (UCAT.CRPLUS1), its “connector” record in the master catalog must be updated to CRPL00.

Prudence dictates that you should consider running the SUPERCLIP command in advance, in SIMULATE mode, so that you can determine if the command is going to do exactly what you expect. The output from this execution is exactly the same as without SIMULATE, but does not perform any of the actual change operation that a real SUPERCLIP command does.
If your SUPERCLIP window is very short, you can also use the SIMULATE facility to complete the long-running tasks of SUPERCLIP ahead of time and then perform the actual “clip” process in a very short time. To do this, you first run SUPERCLIP with SIMULATE, as shown in Example 3-17.

Example 3-17   SUPERCLIP SIMULATE feature

SUPERCLIP OLDVOLSER(SBOX36) -
    NEWVOLSER(CRPL36) -
    JOURNAL(YCJRES1.SUPERCLP.JOURNAL) -
    SIMULATE

This process does the LOCATE to the BCS for all data sets on the volume, storing the information in the journal file. You then replace the word SIMULATE in the command with the word RESTART. At the beginning of your “clip” window, you run the SUPERCLIP command again. SUPERCLIP uses the previously created journal data set records to “drive” the processing so that it performs the volser change, completing this in a very short time. For this to work correctly, though, it is imperative that there are no allocations.

3.7 Case study #7: Alternate master catalog

This case study demonstrates two techniques for setting up and maintaining an alternate master catalog.

One technique, illustrated in Example 3-18, uses IDCAMS REPRO NOMERGECAT to create a clone of your current master catalog. In this example, INFILE represents the “source” master catalog and is identified as MCAT.SANDBOX.VSBOX01 in the INDD1 DD statement. OUTFILE represents the “target” master catalog for the NOMERGECAT operation and is identified as MCAT.CRPLUS.VSBOX03 in the OUTDD1 DD statement.

Example 3-18   REPRO NOMERGECAT to create an alternate master catalog

//YCJRES1Z JOB MSGCLASS=X,NOTIFY=YCJRES1,RESTART=*  
//S1 EXEC PGM=IDCAMS  
//SYSPRINT DD SYSOUT=*  
//SYSUDUMP DD SYSOUT=*  
//INDD1 DD DSN=MCAT.SANDBOX.VSBOX01,DISP=SHR  
//OUTDD1 DD DSN=MCAT.CRPLUS.VSBOX03,DISP=OLD  
//SYSIN DD *  
    REPRO INFILE(INDD1) OUTFILE(OUTDD1) NOMERGECAT  
/*
IDCAMS SYSTEM SERVICES  TIME: 20:08:36  10/02/01  PAGE 1

IDC11468I NVR/VVR NOW POINTS TO TARGET CATALOG.
IDC00005I NUMBER OF RECORDS PROCESSED WAS 4431
IDC00011I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
IDC00021 IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0

Note the message IDC11468I in the output report in this example, which indicates that NVR/VVR records have been updated to “point” to the new target catalog. The effect of this is to create the requirement that the target is to be the official master catalog (for IPL), and the source catalog is to be your alternate master catalog.
The other technique uses the CR+ MERGECAT command with the COPYONLY keyword. This command copies all records in the “source” master catalog named in INBCS to the “target” master catalog named in OUTBCS (MERGECAT normally moves the records of a catalog, but with COPYONLY, it becomes a copy command.) A fundamental difference in CR+ MERGECAT COPYONLY from that of IDCAMS REPRO NOMERGECAT is that it does not update the BCS back-pointer in the VVDS records, and REPRO does. The result is that, with the CR+ logic, your source master catalog remains your official master catalog, and the target copy master catalog is the alternate master catalog.

Example 3-19 shows how to use the second technique. Note that the output listing from MERGECAT automatically provides a list of every data set record that is copied, so that you can verify the specific data set records that were copied if you want. We generated the JCL and commands in this example using the CR+ ISPF dialogs. The IDCAMS steps allocating the journal file and the target catalog are included because the Generate DEFINE and Generate OUTBCS DEFINE options were selected in the MERGECAT panel.

Example 3-19  CR+ MERGECAT to create an alternate master catalog

```
14.55.40  JOB26005  IEF403I MHLRES1K - STARTED - TIME=14.55.40 - ASID=0038
14.55.41  JOB26005  - --TIMINGS (MINS.)--
14.55.41  JOB26005  - JOBNAME  STEPNAME RC EXCP    CPU    SRB    CLOCK
14.55.41  JOB26005  - MHLRES1K  DEFBCS      00    127    .00    .00    .00
14.55.41  JOB26005  - MHLRES1K  DEFJRNL     00    135    .00    .00    .00
14.55.43  JOB26005  CAT12039I * MERGECAT IS NOW RESTARTABLE *
14.55.57  JOB26005  - MHLRES1K  CRPLUS      00   9287    .02    .00    .27
14.55.57  JOB26005  IEF404I MHLRES1K - ENDED - TIME=14.55.57 - ASID=0038
//MHLRES1K JOB (ACCT),'CAT BATCH',CLASS=A,MSGCLASS=X,NOTIFY=MHLRES1
//DEFBCS    EXEC PGM=IDCAMS
//SYSIN    DD *
DELETE   UCAT.CRPLUS.ALT.MCAT -
UCAT -
RECOVERY
SET      MAXCC = 0
DEFINE   USERCATALOG(ICFCATALOG -
NAME(UCAT.CRPLUS.ALT.MCAT) -
MODEL(MCAT.SANDBOX.Z17.SBOX00) -
CYLINDERS(20 10) -
VOLUMES(CRPL01))
//SYSPRINT DD SYSOUT=* //*
//DEFJRNL   EXEC PGM=IDCAMS
//SYSIN    DD *
DELETE   MHLRES1.MERGECAT.JRNL PURGE
SET      MAXCC = 0
DEFINE   CLUSTER( -
NAME(MHLRES1.MERGECAT.JRNL) -
CYLINDERS(20 20) -
KEYS(65 0) -
RECORDSIZE(32744 32744) -
SPANNED) -
DATA(CONTROLINTERVALSIZE(4096)) -
INDEX(CONTROLINTERVALSIZE(2048))
//SYSPRINT DD SYSOUT=* //*
//CRPLUS    EXEC PGM=CAT00010,REGION=0M
//SYSPRINT DD SYSOUT=* //SYSIN    DD *
MERGECAT INBCS(MCAT.SANDBOX.Z17.SBOX00) -
OUTBCS(UCAT.CRPLUS.ALT.MCAT) -
COPY-ONLY -
```
Chapter 3. Case studies

JOURNAL(MHLRES1.MERGECAT.JRNL) - UNLOCK

CAT10001 Catalog RecoveryPlus MAINSTAR SOFTWARE CORPORATION 06 SEP 2006

CAT12001 >>> MERGECAT STARTING

CAT12001 ***********************
CAT12001 * ALLOCATE FILES *
CAT12001 ***********************
CAT12001 >>> ALL FILES ALLOCATED

CAT12001

CAT12073I 14:55:41 JOURNAL BUILD STARTED

CAT12001 ***********************
CAT12001 * BUILD PROCESS JOURNAL *
CAT12001 ***********************
CAT12001 >>> JOURNAL FILE HAS BEEN POPULATED
CAT12001 ***********************
CAT12039I * MERGECAT IS NOW RESTARTABLE *
CAT120001 ***********************
CAT12042I >>> THE NUMBER OF ELIGIBLE CATALOG ENTRIES IS:
AIX -------------------0
ALIAS ---------------728
CLUSTER --------------62
DATA -----------------86
GDG -------------------1
INDEX ----------------30
NONVSAM -------------2,897
PAGESPACE ------------24
PATH ------------------0
USERCATALOG ----------84
TAPELIBRARY -----------0
TAPEVOLUME ------------0
TOTAL -------------3,912

CAT12001

CAT12073I 14:55:43 JOURNAL BUILD COMPLETED

CAT12001

CAT12073I 14:55:43 CATALOG UPDATE STARTED

CAT12001 ***********************
CAT12001 * COPY CATALOG ENTRIES *
CAT12001 ***********************
CAT12078I NOVDSUPD OR COPYONLY SPECIFIED - VVCR/VVCC AND VVDS BACKPOINTERS WILL NOT BE UPDATED
CAT12035I SYS1.VVDS.VTIVO02 ADDED TO TARGET BCS
CAT12035I SYS1.VVDS.VTIVO03 ADDED TO TARGET BCS
-------------- lines deleted showing 24 more VVDSs--------------
CAT12009I PROCESS STARTED FOR (X) $DSNB10
CAT12011I PROCESS ENDED FOR (X) $DSNB10
CAT12009I PROCESS STARTED FOR (X) $DSN910
CAT12011I PROCESS ENDED FOR (X) $DSN910
CAT12009I PROCESS STARTED FOR (A) ABJ.AABJCLST
CAT12011I PROCESS ENDED FOR (A) ABJ.AABJCLST
-------------- lines deleted ------------
CAT120001 >>> SELECTED ENTRIES HAVE BEEN COPIED
CAT12073I 14:55:57 CATALOG UPDATE COMPLETED

CAT12001

CAT120001 >>> MERGECAT ENDED SUCCESSFULLY

CAT01008I Commands processed: 1
CAT01009I Catalog RecoveryPlus execution complete. Highest return code was 00
3.8 Case study #8: Print and delete BCS/VVDS records

This case study demonstrates several techniques for printing and deleting records from the BCS and VVDS. The facilities that were used in this case study include the IBM ad hoc tool, VVDSFIX, and the Catalog RecoveryPlus ZAP command.

The scenario in this case study is contrived, that is, we intentionally created the situation and then used VVDSFIX and CR+ ZAP to correct the error. Nevertheless, the error is one that occurs in real life and illustrates the type of corrective action that might be necessary.

Example 3-20 illustrates the IDCAMS DIAGNOSE VVDS command. It checks all records in the named VVDS for validity of data content and with the COMPAREDS operand, names a BCS to which all relevant VVDS records will be compared.

Example 3-20  DIAGNOSE VVDS to identify duplicate VVR in VVDS

```
//YCJRES1V JOB MSGCLASS=X,NOTIFY=YCJRES1,RESTART=* 
//S1 EXEC PGM=IDCAMS 
//SYSPRINT DD SYSOUT=*
//DIAGDD DD DSN=SYS1.VVDS.VSBOX36,DISP=SHR,
//         VOL=SER=SBOX36,UNIT=SYSALLDA,AMP='AMORG'
//SYSIN DD *
DIAGNOSE VVDS INFILE(DIAGDD)
/*
IDCAMS  SYSTEM SERVICES               TIME: 18:20:54        10/02/01     PAGE 1
IDC21364I ERROR DETECTED BY DIAGNOSE:
VVDS ENTRY: YCJRES2.CRPLUS4.VVDSTST1.DAT (Z)
RECORD: X'0000B160'
OFFSET: X'0002'
REASON: 41 - DUPLICATE VVR/NVRS IN VVDS
IDC21365I VVDS RECORD DISPLAY:
RECORD: X'0000B160'
000 015F0068 E9000000 00001EE8 C3D1D9C5 E2F24BC3 D9D7D3E4 E2F44BE5 E5C4E2E3 *.¬..Z......YCJRES2.CRPLUS4.VVDSTST1
020 E2E3F14B C4C1E3C1 0019E8C3 D1D9C5E2 F24BC3D9 D3E4E2F2 19E8C3D1 D9C5E4F2 4BC3D9D7 D3E4E2F4 *1..UCAT.CRPLUS2.YCJRES2.CRPLUS4
040 48E3E5C4 E2E3E5E2 21402000 00003000 00000000 00001000 00018000 00000000 *.VVDSTST1.... .................
060 00C00000 0000F000 00000000 FFFFFFFF FF000000 00000000 00000000 00000000 *.{-}.00000000 .................
080 00000000 00000000 00000000 00000000 00000000 00000000 01800000 00000000 00000000 *...........................
```

Note: You might be wondering why it is important to determine which master catalog is the "official" one and which is the alternate master catalog and why these BCS back pointers are important. The simple answer is, in normal day-to-day system processing (whether it is for ordinary data sets or critical system data sets), the BCS to which the back-pointer "points" is largely irrelevant, because it is not used or checked during data set LOCATE or OPEN/CLOSE processing. However, if you attempt to delete the data set, the BCS back-pointer in the VVR/NVR for the data set is checked to ensure that the BCS involved is the same one that is named in the BCS back-pointer. If it is not, the DELETE operation fails.

For that reason, when you use IDCAMS REPRO NOMERGECAT to clone a master catalog, and it updates the BCS back-pointers, that makes the target master catalog your official master catalog for future IPLs. When you use CR+ MERGECAT COPYONLY, it does not update the BCS back-pointers. That leaves your original source master catalog as your official master catalog, and the target is your alternate master. Either way, it works. You just need to know the differences so that you do not accidentally fall into a trap of using the wrong catalog as your master catalog.
The output from this command shows that there was a duplicate VVR/NVR in the VVDS for the VSAM component named YCJRES2.CRPLUS4.VVDSTST1.DATA (the Z following the name indicates that this is a primary VVR for a VSAM data set component). This particular VVR record is identified as the **incorrect** VVR. Simply, it is the one dumped; the next error tells us why.

The next error that was identified by DIAGNOSE specifically indicated that extent information for the data sets in this VVR does not match the corresponding VTOC record. For further analysis, the Format-1 DSCB record from the VTOC record was then dumped. Presumably, the other VVR of this same component name had extent information that matched the corresponding VTOC record.

The conclusion is that we had a duplicate VVR in the VVDS for the data set named YCJRES2.CRPLUS4.VVDSTST1.DATA. Note that we do not know anything about the “other” VVR—presumably the correct one—particularly where it is located in the VVDS. The DIAGNOSE output only indicates that the VVR **in error** is located at RBA offset **x'0000B160’** in the VVDS. Understanding the implication of this is important if you want to fix this problem correctly. Otherwise, you might easily delete the wrong record. The rest of this case study shows the various ways to correct this using the VVDSFIX utility and CR+.

Example 3-21 illustrates how this problem of the duplicate VVR can be solved with VVDSFIX. The command (CMD) being used is DELVVR, which means delete VVR.

**Example 3-21 VVDSFIX DELVVR command to delete a VVDS record**

```
//YCJRES1F JOB MSGCLASS=X,NOTIFY=YCJRES1,RESTART=*  
//VVDSGO EXEC PGM=VVDSFIX  
//SYSPRINT DD SYSUT=*
//VVDS DD DSN=SYS1.VVDS.VSBOX36,DISP=SHR,UNIT=3390
//CATDD DD DSN=UCAT.CRPLUS1,DISP=SHR
//SYSIN DD *  
CMD=DELVVR;YCJRES2.CRPLUS4.VVDSTST1.DATA;Y;CO;UCAT.CRPLUS2;
```
Enter command for VVDS fixer
Enter VVR component or Cluster name.
The name entered = "YCJRES2.CRPLUS4.VVDSTST1.DATA ". Is this correct?(Y/N)
Is this a Cluster or a Component (data or index) name ? (Enter CL or CO)
Enter CATALOG name that goes with the VVR (optional)
The specific VVR is identified by component name YCJRES2.CRPLUS4.VVDSTST1.DATA, and it is the same one identified previously in the DIAGNOSE output. The operand value CO that is specified in the command indicates that we want to delete a component. The last operand, UCAT.CRPLUS2, is crucial to the success of the command for this particular problem (see the description for Example 3-23 on page 73 for more information). When the BCS name is not specified, the DELVVR command deletes every VVR in the VVDS that has the specified name, which is not what we wanted to do here. By specifying the BCS name, the DELVVR command can match up the BCS cluster record with the correct VVR and then delete only the invalid VVR.

An alternative method to delete the duplicate VVDS record is the CR+ ZAP VVDS DELETE command (Example 3-22). The command specifies the explicit component name, the volser where the VVDS is housed, and importantly, the RBA offset of the record, at x'00008160', within the VVDS. With the RBA specification, we deleted the correct duplicate VVR, because we told the ZAP DELETE command exactly where to find the record. If we had not specified the RBA offset, and let the ZAP DELETE command determine the location of the record by reading sequentially through the VVDS, it would have deleted the first record of that component name that it came to, and that might not have been the one we wanted to delete. The next example shows just how important this detail was to us.

Example 3-22   CR+ ZAP VVDS DELETE to delete a VVDS record

Important: The ZAP VVDS DELETE command is not intended to be a functional replacement for the IDCAMS DELETE VVR command because it only deletes the specific record that it is asked to delete. In other words, if the VVDS record to be deleted has a corresponding VTOC record, the VTOC record is not deleted with this command. With IDCAMS DELETE VVR, any corresponding VTOC record is deleted.

```
/*

Enter VVR component or Cluster name.
The name entered = "YCJRES2.CRPLUS4.VVDSTST1.DATA ". Is this correct?(Y/N)
Is this a Cluster or a Component (data or index) name ? (Enter CL or CO)
Enter CATALOG name that goes with the VVR (optional)
```

```
IEF142I YCJRES1F VVDSGO - STEP WAS EXECUTED - COND CODE 0000
```

```
IEF142I YCJRES1F VVDSGO - STEP WAS EXECUTED - COND CODE 0000
```

```
The specific VVR is identified by component name YCJRES2.CRPLUS4.VVDSTST1.DATA, and it is the same one identified previously in the DIAGNOSE output. The operand value CO that is specified in the command indicates that we want to delete a component. The last operand, UCAT.CRPLUS2, is crucial to the success of the command for this particular problem (see the description for Example 3-23 on page 73 for more information). When the BCS name is not specified, the DELVVR command deletes every VVR in the VVDS that has the specified name, which is not what we wanted to do here. By specifying the BCS name, the DELVVR command can match up the BCS cluster record with the correct VVR and then delete only the invalid VVR.

An alternative method to delete the duplicate VVDS record is the CR+ ZAP VVDS DELETE command (Example 3-22). The command specifies the explicit component name, the volser where the VVDS is housed, and importantly, the RBA offset of the record, at x'00008160', within the VVDS. With the RBA specification, we deleted the correct duplicate VVR, because we told the ZAP DELETE command exactly where to find the record. If we had not specified the RBA offset, and let the ZAP DELETE command determine the location of the record by reading sequentially through the VVDS, it would have deleted the first record of that component name that it came to, and that might not have been the one we wanted to delete. The next example shows just how important this detail was to us.

Example 3-22   CR+ ZAP VVDS DELETE to delete a VVDS record

Important: The ZAP VVDS DELETE command is not intended to be a functional replacement for the IDCAMS DELETE VVR command because it only deletes the specific record that it is asked to delete. In other words, if the VVDS record to be deleted has a corresponding VTOC record, the VTOC record is not deleted with this command. With IDCAMS DELETE VVR, any corresponding VTOC record is deleted.

```
/*

Enter VVR component or Cluster name.
The name entered = "YCJRES2.CRPLUS4.VVDSTST1.DATA ". Is this correct?(Y/N)
Is this a Cluster or a Component (data or index) name ? (Enter CL or CO)
Enter CATALOG name that goes with the VVR (optional)
```

```
IEF142I YCJRES1F VVDSGO - STEP WAS EXECUTED - COND CODE 0000
```

```
The specific VVR is identified by component name YCJRES2.CRPLUS4.VVDSTST1.DATA, and it is the same one identified previously in the DIAGNOSE output. The operand value CO that is specified in the command indicates that we want to delete a component. The last operand, UCAT.CRPLUS2, is crucial to the success of the command for this particular problem (see the description for Example 3-23 on page 73 for more information). When the BCS name is not specified, the DELVVR command deletes every VVR in the VVDS that has the specified name, which is not what we wanted to do here. By specifying the BCS name, the DELVVR command can match up the BCS cluster record with the correct VVR and then delete only the invalid VVR.

An alternative method to delete the duplicate VVDS record is the CR+ ZAP VVDS DELETE command (Example 3-22). The command specifies the explicit component name, the volser where the VVDS is housed, and importantly, the RBA offset of the record, at x'00008160', within the VVDS. With the RBA specification, we deleted the correct duplicate VVR, because we told the ZAP DELETE command exactly where to find the record. If we had not specified the RBA offset, and let the ZAP DELETE command determine the location of the record by reading sequentially through the VVDS, it would have deleted the first record of that component name that it came to, and that might not have been the one we wanted to delete. The next example shows just how important this detail was to us.

Example 3-22   CR+ ZAP VVDS DELETE to delete a VVDS record

Important: The ZAP VVDS DELETE command is not intended to be a functional replacement for the IDCAMS DELETE VVR command because it only deletes the specific record that it is asked to delete. In other words, if the VVDS record to be deleted has a corresponding VTOC record, the VTOC record is not deleted with this command. With IDCAMS DELETE VVR, any corresponding VTOC record is deleted.

```
/*

Enter VVR component or Cluster name.
The name entered = "YCJRES2.CRPLUS4.VVDSTST1.DATA ". Is this correct?(Y/N)
Is this a Cluster or a Component (data or index) name ? (Enter CL or CO)
Enter CATALOG name that goes with the VVR (optional)
```

```
IEF142I YCJRES1F VVDSGO - STEP WAS EXECUTED - COND CODE 0000
```

```
The specific VVR is identified by component name YCJRES2.CRPLUS4.VVDSTST1.DATA, and it is the same one identified previously in the DIAGNOSE output. The operand value CO that is specified in the command indicates that we want to delete a component. The last operand, UCAT.CRPLUS2, is crucial to the success of the command for this particular problem (see the description for Example 3-23 on page 73 for more information). When the BCS name is not specified, the DELVVR command deletes every VVR in the VVDS that has the specified name, which is not what we wanted to do here. By specifying the BCS name, the DELVVR command can match up the BCS cluster record with the correct VVR and then delete only the invalid VVR.

An alternative method to delete the duplicate VVDS record is the CR+ ZAP VVDS DELETE command (Example 3-22). The command specifies the explicit component name, the volser where the VVDS is housed, and importantly, the RBA offset of the record, at x'00008160', within the VVDS. With the RBA specification, we deleted the correct duplicate VVR, because we told the ZAP DELETE command exactly where to find the record. If we had not specified the RBA offset, and let the ZAP DELETE command determine the location of the record by reading sequentially through the VVDS, it would have deleted the first record of that component name that it came to, and that might not have been the one we wanted to delete. The next example shows just how important this detail was to us.

Example 3-22   CR+ ZAP VVDS DELETE to delete a VVDS record

Important: The ZAP VVDS DELETE command is not intended to be a functional replacement for the IDCAMS DELETE VVR command because it only deletes the specific record that it is asked to delete. In other words, if the VVDS record to be deleted has a corresponding VTOC record, the VTOC record is not deleted with this command. With IDCAMS DELETE VVR, any corresponding VTOC record is deleted.

```
/*

Enter VVR component or Cluster name.
The name entered = "YCJRES2.CRPLUS4.VVDSTST1.DATA ". Is this correct?(Y/N)
Is this a Cluster or a Component (data or index) name ? (Enter CL or CO)
Enter CATALOG name that goes with the VVR (optional)
```

```
IEF142I YCJRES1F VVDSGO - STEP WAS EXECUTED - COND CODE 0000
```

```
The specific VVR is identified by component name YCJRES2.CRPLUS4.VVDSTST1.DATA, and it is the same one identified previously in the DIAGNOSE output. The operand value CO that is specified in the command indicates that we want to delete a component. The last operand, UCAT.CRPLUS2, is crucial to the success of the command for this particular problem (see the description for Example 3-23 on page 73 for more information). When the BCS name is not specified, the DELVVR command deletes every VVR in the VVDS that has the specified name, which is not what we wanted to do here. By specifying the BCS name, the DELVVR command can match up the BCS cluster record with the correct VVR and then delete only the invalid VVR.

An alternative method to delete the duplicate VVDS record is the CR+ ZAP VVDS DELETE command (Example 3-22). The command specifies the explicit component name, the volser where the VVDS is housed, and importantly, the RBA offset of the record, at x'00008160', within the VVDS. With the RBA specification, we deleted the correct duplicate VVR, because we told the ZAP DELETE command exactly where to find the record. If we had not specified the RBA offset, and let the ZAP DELETE command determine the location of the record by reading sequentially through the VVDS, it would have deleted the first record of that component name that it came to, and that might not have been the one we wanted to delete. The next example shows just how important this detail was to us.

Example 3-22   CR+ ZAP VVDS DELETE to delete a VVDS record

Important: The ZAP VVDS DELETE command is not intended to be a functional replacement for the IDCAMS DELETE VVR command because it only deletes the specific record that it is asked to delete. In other words, if the VVDS record to be deleted has a corresponding VTOC record, the VTOC record is not deleted with this command. With IDCAMS DELETE VVR, any corresponding VTOC record is deleted.
In Example 3-23, we ran a CR+ ZAP VVD PRINT command, specifying YCJRES2.CRPLUS4.VVDSTST1.DATA, which again was the component name of the duplicate VVR. In the messages preceding the dump, note the RBA offset of the record found, at x‘000096DE’ in the VVDS. Remember that the invalid, duplicate record is at RBA offset x‘0000B160’, at a location after the valid record. As you can see, the correct record of the duplicate pair is at a lower offset location, where the invalid record is at a higher offset. In this instance, if we can delete the invalid record, it is imperative that we explicitly point to it, whether with the user catalog name that is specified in the VVDSFIX DELVVR command or with the CR+ ZAP DELETE command.

Example 3-23  CR+ ZAP PRINT of a VVDS record

//YCJRESIX JOB MSGCLASS=X,NOTIFY=YCJRES1,RESTART=*  /*S1 EXEC PGM=CAT00010  //SYSPRINT DD SYSOUT=*  //SYSIN DD */  ZAP VVDS(SBOX36) -  PRINT(COMPONENT(YCJRES2.CRPLUS4.VVDSTST1.DATA) -  GENERICKEY DUMP)  /*
  CAT01001I Catalog RecoveryPlus MAINSTAR SOFTWARE CORPORATION 02 OCT 2001 PAGE:1
 *******************************************************************************
  VVDS DATA FOR DSN: YCJRES2.CRPLUS4.VVDSTST1.DATA
  RBA FOUND: 000096DE
  TYPE FOUND: Z
 *******************************************************************************
  00000000 015F0068 E9000000 00001EE8 C3D1D9C5 *.?..Z......YCJRE*
  00000010 E2F24BC3 D9D7D3E4 E2F44BE5 E5C4E2E3 *S2.CRPLUS4.VVDST*
  00000020 E2E3F14B C4C1E3C1 0019E8C3 D1D9C5E2 *ST1..UCAT.CRPLUS2*
  00000030 F24BC3D9 D7D3E4E2 F4BE5E5 C4E2E3E2 *.YCJRES2.CRPLUS4*
  00000040 59E83C31 00010000 F0000000 00000000 *.VVDSTST1.... ..*
  00000050 00000050 00000000 00000000 00000000 *.................*
  00000060 FF000000 00000000 00000000 00000000 *.................*
  00000070 00000000 00000000 00000000 00000000 *.................*
  00000080 00000000 00000000 00000000 00000000 *.................*
  00000090 00000000 00000000 00000000 00000000 *.................*
  000000A0 00000000 00000000 00000000 00000000 *.................*
  000000B0 00010000 01880000 00000000 00000000 *.................*
  000000C0 62608000 60000000 00000000 00000000 *.................*
  000000D0 00000000 00000000 00000000 00000000 *.................*
  000000E0 00000000 00000000 00000000 00000000 *.................*
  000000F0 00866A27 273689E A6000000 01000000 *.FS.L, W.....*
  00000100 64000000 00000000 00000000 00000000 *.................*
  00000110 00000000 00000000 00000000 00000000 *.................*
  00000120 00000000 00000000 00000000 00000000 *.................*
  00000130 00000000 00000000 00000000 00000000 *.................*
  00000140 00000000 00000000 00000000 00000000 *.................*
  00000150 01000000 03001000 00000000 00BFF *.................*  CAT080001 ZAP FUNCTION COMPLETE. RETURN CODE 0  CAT01009I Catalog RecoveryPlus EXECUTION COMPLETE. HIGHEST RETURN CODE WAS 0

Example 3-24 illustrates that it is also possible to delete a specific BCS record with VVDSFIX DELBCSR. The key of the record to be deleted is specified (in this case, a cluster sphere name), and so is the BCS name from which the record is to be deleted. In the output report, we were told that a C-type (Cluster) record is deleted, and a T-type (Truename) record for the accompanying data component is deleted.

Example 3-24  VVDSFIX DELBCSR command to delete a BCS record

//YCJRESIX JOB MSGCLASS=X,NOTIFY=YCJRES1,RESTART=*  //VVDGSO EXEC PGM=VVDGSO  //SYSPRINT DD SYSOUT=*  //VVDSD DD DSN=SYS1.VVDS.VSBOX94,DISP=SHR,VOL=SER=SBOX94,UNIT=3390
You can also delete a BCS record with the CR+ ZAP BCS DELETE command by specifying the key of the record to be deleted (a cluster sphere name) and the BCS name from which the record is to be deleted. Only the record with that key is deleted (Example 3-25).

Example 3-25  CR+ ZAP BCS DELETE to delete a BCS record

```
//YCJRES1Z JOB MSGCLASS=X,NOTIFY=&SYSUID,RESTART=*  // S1 EXEC PGM=CAT00010
//SYSPRINT DD SYSOUT=*  // SYsin DD *
//SYsin DD *
ZAP BCS(UCAT.CRPLUS1) -
   DELETE(KEY(YCJRES1.CRPLUS.TESTE.E000025))
/*
CAT01001I Catalog RecoveryPlus MAINSTAR SOFTWARE CORPORATION 02 OCT 2001 PAGE:1
CAT08213I RECORD DELETED
CAT08000I ZAP FUNCTION COMPLETE. RETURN CODE 0
CAT01009I Catalog RecoveryPlus EXECUTION COMPLETE. HIGHEST RETURN CODE WAS 0
```

### 3.9 Case study #9: Preventive maintenance

This case study demonstrates how to use the CR+ DIAGNOSE command so that you can establish true catalog preventive maintenance. It uses two commands that can run on a regularly-scheduled basis with fixes that are automatically created for problems encountered.

The commands are DIAGNOSE BCS-VVDS and DIAGNOSE VVDS-BCS, both specified with masking values for the BCSs and VVDSs and the ALLRELATED keyword. These two features permit these commands to be run over and over again, regardless of changes made to the catalog and disk environment.

Example 3-26 on page 75 illustrates the DIAGNOSE BCS-VVDS command. This command compares all records in the selected BCSs to determine if they each have corresponding VVR/NVR records in the appropriate VVDS. In this case, the COMPAREBCS mask selected all BCSs that matched UCAT.CRPLUS*.

The command requires only two small changes to convert it into one that diagnoses the entire environment:

- COMPAREBCS(**)
- ALLRELATED (instead of the current TOVVDS keyword)

With these changes, this DIAGNOSE command selects every BCS in your environment and compares outward to every “related” VVDS.
How much time does this take? For Example 3-26, the total run time was less than two elapsed minutes. This is low because it was the ITSO test environment, with fairly small BCSs and not a high number of data sets in each volume.

Example 3-26  CR+ DIAGNOSE BCS-VVDS to search for errors

17.56.28 JOB11471 IEF403I YCJRES1B - STARTED - ASID=001B.
17.58.17 JOB11471 IEF404I YCJRES1B - ENDED - ASID=001B.
//YCJRES1B JOB MSGCLASS=X,NOTIFY=YCJRES1,RESTART=*
//S1 EXEC PGM=CAT00010
//SYSPRINT DO SYSOUT=*
//SYSOUT DO SYSOUT=*
//SYSIN DO *
  DIAGNOSE BCS-VVDS -
  COMPAREBCS(UCAT.CRPLUS*) -
  TOVVDS(SBOX**) -
  JOURNAL(YCJRES1.CRPLUS.DIAGNOSE.JOURNAL) -
  FIXDATASET(YCJRES1.CRPLUS.DIAGNOSE.FIXFILE)
/*
CAT01001I Catalog RecoveryPlus MAINSTAR SOFTWARE CORPORATION 04 OCT 2001 PAGE:1

CAT11030I DSNS/MASKS FOR KEYWORD: COMPAREBCS
  UCAT.CRPLUS1
  UCAT.CRPLUS2
  UCAT.CRPLUS3
  UCAT.CRPLUS4

CAT11033I VOLSERS OR MASKS FOR KEYWORD: TOVVDS
  SBOX**

CAT11033I THE FOLLOWING BCS(S) WILL BE DIAGNOSED:
  UCAT.CRPLUS1
  UCAT.CRPLUS2
  UCAT.CRPLUS3
  UCAT.CRPLUS4

CAT11032I THE FOLLOWING VOLSERS ARE ELIGIBLE:
  SBOX01 SBOX02 SBOX03 SBOX04 SBOX05 SBOX06 SBOX07 SBOX08 SBOX09 SBOX10 SBOX11
  SBOX12 SBOX13 SBOX14 SBOX15 SBOX16 SBOX17 SBOX18 SBOX19 SBOX20 SBOX21 SBOX22 SBOX23 SBOX24 SBOX25 SBOX26 SBOX27 SBOX28
  SBOX29 SBOX30 SBOX31 SBOX32 SBOX33 SBOX34 SBOX35 SBOX36 SBOX37 SBOX38 SBOX39 SBOX40 SBOX41 SBOX42 SBOX43 SBOX44 SBOX45 SBOX46 SBOX47 SBOX48 SBOX49 SBOX50 SBOX51 SBOX52 SBOX53 SBOX54 SBOX55 SBOX56 SBOX57 SBOX58 SBOX59 SBOX60 SBOX61 SBOX62 SBOX63 SBOX64 SBOX65 SBOX66 SBOX67 SBOX68 SBOX69 SBOX70 SBOX71 SBOX72 SBOX73 SBOX74 SBOX75 SBOX76 SBOX77 SBOX78 SBOX79 SBOX80 SBOX81 SBOX82 SBOX83 SBOX84 SBOX85 SBOX86 SBOX87 SBOX88 SBOX89 SBOX90 SBOX91 SBOX92 SBOX93 SBOX94 SBOX95 SBOX96 SBOX97

CAT11040I 17.56.32 DIAG BCS-VVDS STEP: BCS READ - STARTED
CAT11040I 17.57.20 DIAG BCS-VVDS STEP: BCS READ - COMPLETED

CAT11040I 17.57.23 DIAG BCS-VVDS STEP: VVDS EXTRACT - STARTED
CAT11040I 17.57.30 DIAG BCS-VVDS STEP: VVDS EXTRACT - COMPLETED

CAT11040I 17.57.30 DIAG BCS-VVDS STEP: VTOC EXTRACT - STARTED
CAT11040I 17.58.13 DIAG BCS-VVDS STEP: VTOC EXTRACT - COMPLETED

CAT11040I 17.58.13 DIAG BCS-VVDS STEP: SORT - STARTED
CAT11040I 17.58.17 DIAG BCS-VVDS STEP: SORT - COMPLETED

<table>
<thead>
<tr>
<th>BCS-VVDS MAPPING</th>
<th>VVDS IN BCS IN DATASETS FIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCS NAME</td>
<td>VOLSER</td>
</tr>
<tr>
<td>UCAT.CRPLUS1</td>
<td>SBOX01</td>
</tr>
<tr>
<td>SBOX02</td>
<td>YES</td>
</tr>
</tbody>
</table>

Chapter 3. Case studies  75
Example 3-27 on page 77 illustrates a CR+ DIAGNOSE VVDS-BCS command. This command compares all records in the selected VVDSs to determine if they each have corresponding catalog records in the appropriate BCS. In this case, the COMPAREVVDS mask selected all volumes that match SBOX*, which was all of the volumes available to us in our ITSO test environment. The ALLRELATED keyword indicated that all BCSs that were “related” to these volumes were to be diagnosed. For all errors identified, an IDCAMS fix command was created and stored in the “flat file” identified by the FIXDATASET keyword.

For this run, a total of 415 errors were identified, which seems very large. It is a safe to say that almost every installation that runs a command such as this is likely to encounter a high number of such errors when the command is run for the first time. For years, too little attention was given to keeping errors like this under control; given the current situation, there are likely to be large numbers of “orphaned” data sets, wasted disk space, and errors lying around that can cause problems when you least expect them. Once the environment is cleaned up, subsequent and regularly-scheduled runs should not find so many errors.

A small change converts this command into one that diagnoses the entire environment:

```
COMPAREVVDS(**)
```
This DIAGNOSE command selects every VVDS in your environment and compares outward to every "related" BCS. How much time does this take? For Example 3-27, the total run time was approximately 6.5 elapsed minutes. This is because it was the ITSO test environment, with a fairly small number of volumes and not a high number of data sets on each volume.

**Example 3-27**  
**CR+ DIAGNOSE VVDS-BCS to search for errors**

```
17.53.22 J0811469 IEF403I YCJRES1C - STARTED - ASID=0019.
17.59.56 J0811469 IEF404I YCJRES1C - ENDED - ASID=0019.
//YCJRES1C JOB MSGCLASS=X,NOTIFY=YCJRES1C,RESTART=S0
//SYSOUT DD SYSOUT=*  
//SYSIN   DD SYSOUT=*  
  DIAGNOSE VVDS-BCS -  
    COMPAREVVDS(SBOX*) -  
    ALLRELATED -  
    FIXDATASET(YCJRES1.CRP+DIAGNOSE.FIXFILE2)
/*

CAT11001I Catalog RecoveryPlus MAINSTAR SOFTWARE CORPORATION 04 OCT 2001 PAGE:1
CAT11116I THE FOLLOWING ENTRIES/MASKS ARE TO BE PROCESSED DURING COMPAREVVDS
    PROCESSING.
    SBOX*

CAT11103I THE FOLLOWING VOLUMES HAVE MET ALL SELECTION REQUIREMENTS AND WILL BE
    PROCESSED:
    SBOX01 SBOX02 SBOX03 SBOX04 SBOX05 SBOX06 SBOX07 SBOX08 SBOX09
    SBOX10 SBOX11 SBOX12 SBOX13 SBOX14 SBOX15 SBOX16
    SBOX17 SBOX18 SBOX19 SBOX20 SBOX21 SBOX22 SBOX23 SBOX24 SBOX25
    SBOX26 SBOX27 SBOX28 SBOX29 SBOX30 SBOX31 SBOX32
    SBOX33 SBOX34 SBOX35 SBOX36 SBOX37 SBOX38 SBOX39
    SBOX40 SBOX41 SBOX42 SBOX43 SBOX44 SBOX45 SBOX46 SBOX47 SBOX48
    SBOX49 SBOX50 SBOX51 SBOX52 SBOX53 SBOX54 SBOX55 SBOX56 SBOX57
    SBOX58 SBOX59 SBOX60 SBOX61 SBOX62 SBOX63 SBOX64
    SBOX65 SBOX66 SBOX67 SBOX68 SBOX69 SBOX70 SBOX71 SBOX72 SBOX73
    SBOX74 SBOX75 SBOX76 SBOX77 SBOX78 SBOX79 SBOX80
    SBOX81 SBOX82 SBOX83 SBOX84 SBOX85 SBOX86 SBOX87 SBOX88 SBOX89
    SBOX90 SBOX91 SBOX92 SBOX93 SBOX94 SBOX95 SBOX96 SBOX97

CAT11129I PROCESSING STARTING FOR VVDS SYS1.VVDS.VSBOX01
CAT11144I VTOC DATA: F1=300     FX=4,201   CVAF CALLS=91

CAT11122I PRELIMINARY CHARACTERISTICS OF VVDS SYS1.VVDS.VSBOX01
    SMS MANAGED NO
    CONTROL INTERVALS ALLOCATED 120
    CATALOGS REFERRING 15

CAT11119W VTOC ENTRY ANGIEB.DDIR.I IS NOT CATALOGED
CAT11184I CORRESPONDING FIX: 1
CAT11119W VTOC ENTRY DB2QM.UNLOAD.MODEL IS NOT CATALOGED
CAT11184I CORRESPONDING FIX: 2
CAT11119W VTOC ENTRY HERING.GDG.TESTDATA IS NOT CATALOGED
CAT11184I CORRESPONDING FIX: 3
CAT11119W VTOC ENTRY PAOLOR3.ALLRECS.GDGMODEL IS NOT CATALOGED
CAT11184I CORRESPONDING FIX: 4
CAT11119W VTOC ENTRY TATERS3.GDG1 IS NOT CATALOGED
CAT11184I CORRESPONDING FIX: 41
CAT11190I ONE OR MORE DELETE...VVR OR DELETE...NVR COMMANDS HAVE BEEN GENERATED
    FOR THIS VVDS

CAT11123I FINAL CHARACTERISTICS OF VVDS SYS1.VVDS.VSBOX01
    PRIMARY VVRS (Z) ENCOUNTERED 296
    SECONDARY VVRS (Q) ENCOUNTERED 38
    NON VSAMS (N) ENCOUNTERED 0
    VVCNS ENCOUNTERED 0
    VVCMS ENCOUNTERED 0

** DIAGNOSE VVDS-BCS SYS1.VVDS.VSBOX01 **
** UNKNOWN TYPES ENCOUNTERED 0**
### Catalog Summary for VVDS SYS1.VVDS.VSBOX01

- **Catalog Name**
- **VV/NVR Count**

<table>
<thead>
<tr>
<th>Catalog Name</th>
<th>VV/NVR Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATALOG.TOTICF1.VTOTCAT</td>
<td>0</td>
</tr>
<tr>
<td>MCAT.CRPLUS.VSBOX01</td>
<td>2</td>
</tr>
<tr>
<td>MCAT.CRPLUS.VSBOX03</td>
<td>0</td>
</tr>
<tr>
<td>MCAT.SANDBOX.R10.VSBOX11</td>
<td>2</td>
</tr>
<tr>
<td>MCAT.SANDBOX.R9.VSBOX11</td>
<td>4</td>
</tr>
<tr>
<td>MCAT.SANDBOX.VSBOX01</td>
<td>20</td>
</tr>
<tr>
<td>MCAT.SANDBOX.VSBOX05</td>
<td>2</td>
</tr>
<tr>
<td>UCAT.CRPLUS1</td>
<td>51</td>
</tr>
<tr>
<td>UCAT.CRPLUS2</td>
<td>6</td>
</tr>
<tr>
<td>UCAT.CRPLUS3</td>
<td>2</td>
</tr>
<tr>
<td>UCAT.CRPLUS4</td>
<td>2</td>
</tr>
<tr>
<td>UCAT.TATERS2</td>
<td>2</td>
</tr>
<tr>
<td>UCAT.VSBOX05</td>
<td>36</td>
</tr>
<tr>
<td>UCAT.VSBOX09</td>
<td>203</td>
</tr>
<tr>
<td>UCAT.VSBOX23</td>
<td>2</td>
</tr>
</tbody>
</table>

**CAT11129I** PROCESSING COMPLETE FOR VVDS SYS1.VVDS.VSBOX01

CAT11129I PROCESSING STARTING FOR VVDS SYS1.VVDS.VSBOX02

**CAT11144I** VTOC DATA: F1=2,543 FX=1,955 CVAF CALLS=91

**CAT11122I** PRELIMINARY CHARACTERISTICS OF VVDS SYS1.VVDS.VSBOX02
- **SMS Managed No**
- **Control Intervals Allocated 120**
- **Catalogs Referencing 15**

**CAT11132W** UNABLE TO LOCATE COMPONENT (Z) IXGLOGR.CICSTT11.DFHLOG.SC63.DATA IN BCS MCAT.SANDBOX.VSBOX01

**CAT11137W** UNABLE TO LOCATE BASE CLUSTER IXGLOGR.CICSTT11.DFHLOG.SC63 IN BCS MCAT.SANDBOX.VSBOX01 FOR COMPONENT IXGLOGR.CICSTT11.D

**CAT11137W** LOG.SC63.DATA

**CAT11184I** CORRESPONDING FIX: 42

**CAT11132W** UNABLE TO LOCATE COMPONENT (Z) IXGLOGR.CICSTT21.DFHLOG.SC63.DATA IN BCS MCAT.SANDBOX.VSBOX01

**CAT11137W** UNABLE TO LOCATE BASE CLUSTER IXGLOGR.CICSTT21.DFHLOG.SC63 IN BCS MCAT.SANDBOX.VSBOX01 FOR COMPONENT IXGLOGR.CICSTT21.D

**CAT11137W** LOG.SC63.DATA

**CAT11184I** CORRESPONDING FIX: 43

**CAT11119W** VTOC ENTRY ANDREVN.BRODCAST IS NOT CATALOGUED

**CAT11184I** CORRESPONDING FIX: 101

**CAT11150I** ONE OR MORE DELETE...VVR OR DELETE...NVR COMMANDS HAVE BEEN GENERATED FOR THIS VVDS

**CAT11123I** FINAL CHARACTERISTICS OF VVDS SYS1.VVDS.VSBOX02
- **Primary VVRS (Z) Encountered 816**
- **Secondary VVRS (Q) Encountered 6**
- **Non VSAMS (N) Encountered 0**
- **VVCMS Encountered 0**
- **VVCNS Encountered 0**
- **Unknown Types Encountered 0**

**CAT11125I** CATALOG SUMMARY FOR VVDS SYS1.VVDS.VSBOX02

- **Catalog Name**
- **VV/NVR Count**

<table>
<thead>
<tr>
<th>Catalog Name</th>
<th>VV/NVR Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATALOG.SHRICF1.VIODFPK</td>
<td>1</td>
</tr>
<tr>
<td>CATALOG.TOTICF2.VTOTCAT</td>
<td>0</td>
</tr>
<tr>
<td>MCAT.CRPLUS.VSBOX01</td>
<td>3</td>
</tr>
<tr>
<td>MCAT.CRPLUS.VSBOX03</td>
<td>0</td>
</tr>
<tr>
<td>MCAT.SANDBOX.R10.VSBOX11</td>
<td>74</td>
</tr>
<tr>
<td>MCAT.SANDBOX.VSBOX01</td>
<td>3</td>
</tr>
<tr>
<td>MCAT.SANDBOX.VSBOX11</td>
<td>1</td>
</tr>
<tr>
<td>MCAT.SBOXALT.VSBOX2</td>
<td>2</td>
</tr>
<tr>
<td>UCAT.CRPLUS1</td>
<td>562</td>
</tr>
<tr>
<td>UCAT.CRPLUS2</td>
<td>142</td>
</tr>
</tbody>
</table>
3.10 Case study #10: Reorganizing a catalog

This case study shows how to use the CR+ REORG command to reorganize a BCS while it is open. Note the WTO in the job log (Example 3-28). Because we specified NO-REPLY, no operator response was required. We made a backup to the data set identified in the OUTDD statement before the REORG. No explicit attribute changes were requested, but IMBED and REPLICATE were removed.

Example 3-28  CR+ REORG

17.53.05  JOB26031  IEF403I MHLRES1Q - STARTED - TIME=17.53.05 - ASID=0038
17.53.05  JOB26031  *CAT13030I Re-org for catalog requested:  088
088             CATALOG.TOTICF2.VTOTCAT on TOTCAT
088             Catalog accesses temporarily suspended by MHLRES1Q X'0038' SC64
17.53.21  JOB26031  *CAT13031I Catalog Re-org in progress - DO NOT CANCEL 089
089             MHLRES1Q SC64
089             Catalog.TOTICF2.VTOTCAT on TOTCAT
17.53.21  JOB26031  *CAT13032I Catalog Re-org Completed Successfully RC=X'00'  090
090             MHLRES1Q 0038  SC64
090             Catalog.TOTICF2.VTOTCAT on TOTCAT
17.53.21  JOB26031  -                                 --TIMINGS (MINS.)--
17.53.21  JOB26031  -JOBNAME  STEPNAME     RC   EXCP    CPU    SRB  CLOCK
17.53.21  JOB26031  MHLRES1Q  CRPLUS      00  2855    .03    .00    .27
17.53.21  IEF404I MHLRES1Q - ENDED - TIME=17.53.21 - ASID=0038
//MHLRES1Q JOB (ACCT),'CAT BATCH',CLASS=A,MSGCLASS=X,NOTIFY=MHLRES1
//CRPLUS    EXEC PGM=CAT00010,REGION=0M
//SYSPRINT DD SYSOUT=* //OUTDD  DD DSN=MHLRES1.RWO.BKUP4,  // DISP=(,CATLG,DELETE),UNIT=SYSALLDA,  // SPACE=(CYL,(15,15),RLSE) //SYSIN  DD * REORG     BCS(CATALOG.TOTICF2.VTOTCAT) -
CAT01001I Catalog RecoveryPlus MAINSTAR SOFTWARE CORPORATION 06 SEP 2006

CAT13042I IMBED Catalog will be converted to NOIMBED
CAT13042I REPLICATE Catalog will be converted to NOREPLICATE

CAT13081I Attribute Change Summary

<table>
<thead>
<tr>
<th>OLD</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALUE</td>
<td>VALUE</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Imbed/NoImbed</td>
<td>IMBD</td>
</tr>
<tr>
<td>Replicate/NoReplicate</td>
<td>REPL</td>
</tr>
</tbody>
</table>

CAT13045I Start unload for CATALOG.TOTICF2.VTOTCAT 06 Sep 2006 (2006249)

CAT13046I Unload Record Summary for CATALOG.TOTICF2.VTOTCAT

Cluster | GDG | E | J | nonVSAM | Truename | Path
---|---|---|---|-------|---------|---
3,049 | 35 | 0 | 0 | 29,907 | 3,730 | 15

Spanned Record Count: 0
Record Length Errors: 0

CAT13047I End unload for CATALOG.TOTICF2.VTOTCAT 06 Sep 2006 (2006249)
CAT13058I Catalog RecoveryPlus BACKUP WRITTEN to OUTDD

CAT13011I Internal Backup Summary:

<table>
<thead>
<tr>
<th>Index CI</th>
<th>Index CI</th>
<th>Data CI</th>
<th>Data CI</th>
<th>Fre</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reads</td>
<td>Writes</td>
<td>Reads</td>
<td>Writes</td>
<td>Data CI</td>
</tr>
<tr>
<td>123</td>
<td>119</td>
<td>19,825</td>
<td>1,672</td>
<td>18,04</td>
</tr>
</tbody>
</table>

CAT13040I Start Preload Verify 06 Sep 2006 17.53.20

CAT13046I Verify Record Summary for CATALOG.TOTICF2.VTOTCAT

Cluster | GDG | E | J | nonVSAM | Truename | Path
---|---|---|---|-------|---------|---
3,049 | 35 | 0 | 0 | 29,907 | 3,730 | 15

Spanned Record Count: 0
Record Length Errors: 0

CAT13041I End Preload Verify 06 Sep 2006 17.53.21

CAT13048I Start reload for CATALOG.TOTICF2.VTOTCAT 06 Sep 2006 (2006249)

CAT13046I Reload Record Summary for CATALOG.TOTICF2.VTOTCAT

Cluster | GDG | E | J | nonVSAM | Truename | Path
---|---|---|---|-------|---------|---
3,049 | 35 | 0 | 0 | 29,907 | 3,730 | 15

Spanned Record Count: 0
Record Length Errors: 0

CAT13049I End reload for CATALOG.TOTICF2.VTOTCAT 06 Sep 2006 (2006249)

CAT13032I Catalog Re-org Completed Successfully RC='X'00'

CAT01008I Commands processed: 1
CAT01009I Catalog RecoveryPlus execution complete. Highest return code was 0
3.11 Case study #11: Locating data sets with IMBED option

This case study demonstrates how to use the Catalog RecoveryPlus EXPLORE command (Example 3-29). The entire system is searched for data sets that have the obsolete IMBED attribute. It might run a long time in your environment but it is likely to find a few surprises for you.

Example 3-29 CR+ EXPLORE to search for data sets with IMBED option

```
13.47.33 JOB25999 IEF403I CATEXPL - STARTED - TIME=13.47.33 - ASID=0038
13.48.23 JOB25999 - --TIMINGS (MINS.)--
13.48.23 JOB25999 -JOBNAME STEPNAME RC EXCP CPU SRB CLOCK
13.48.23 JOB25999 -CATEXPL CRPLUS 00 3966 .06 .00 .83
13.48.23 JOB25999 IEF404I CATEXPL - ENDED - TIME=13.48.23 - ASID=0038
//CATEXPL JOB (ACCT),'CAT BATCH',CLASS=A,MSGCLASS=X,NOTIFY=MHLRES1
//CRPLUS EXEC PGM=CAT00010,REGION=0M
//SYSPRINT DD SYSOUT=* //EXOUT DD SYSOUT=* //SYSPRINT DD SYSOUT=*
//SYSPRINT DD SYSOUT=* //SYSPRINT DD SYSOUT=* //SYSPRINT DD SYSOUT=* //SYSPRINT DD SYSOUT=* EXPLORE - 
DSN-LISTFILE(EXOUT) - CRITERIA( - INCLUDE( - (DSN EQ **.**) AND - (IMBED EQ Y)))}
```

CAT01001I Catalog RecoveryPlus MAINSTAR SOFTWARE CORPORATION 06 SEP

CAT06111I DSNs/mask **.** has been selected
CAT06109I EXPLORE parsing of the "CRITERIA" keyword was successful

CAT06014I Number of catalog entries returned: 30,180
CAT06015I Number of catalog entries selected: 12

DSNLIST datasets 12
COLLIST datasets 0
ICFLIST items 0
EXTRACT items 0
EXTRACT rows 0
MAP items 0

CAT06009I EXPLORE processing completed; return code: 0

CAT01008I Commands processed: 1
CAT01009I Catalog RecoveryPlus execution complete. Highest return code was 0

***** Following is the list of data sets with the IMBED option. (EXOUT DD) *****

SERVRPAC.MASTER.SCPPEENU
SERVRPAC.MASTER.SCPPEENU.DATA
SERVRPAC.MASTER.SCPPEENU.INDEX
SERVRPAC.MASTER.SCPPHENU
SERVRPAC.MASTER.SCPPHENU.DATA
SERVRPAC.MASTER.SCPPHENU.INDEX
SERVRPAC.MASTER.SCPPVENU
SERVRPAC.MASTER.SCPPVENU.DATA
SERVRPAC.MASTER.SCPPVENU.INDEX
3.12 Case study #12: Mapping a BCS

Example 3-30 shows the CR+ MAP command when it is run against a BCS. There are six reports that are available from the MAP command: Catalog Listing, Control Area Analysis, CA Split Distribution, Recordsize Distribution, Index Structure, Statistical Summary, and Recommendations. Refer to Mainstar Catalog RecoveryPlus User Guide, document number 006-0202, for details of interpreting report contents. Three of the reports are shown in the example.

Briefly, from the Catalog Listing, we see that this is a user catalog with a data CISIZE of 4,096 and an index CISIZE of 2,048. There are 180 data CIs per CA. Both CI and CA FREESPACE are zero. The space allocation was specified as CYL (4,1) and the catalog is in 17 extents. Because the secondary allocation quantity is equivalent to one CA, each CA split results in another extent.

The Control Area Analysis Report is the “heart and soul” of the MAP command. It is a point-in-time mapping of each CA in the BCS. The first two columns show the logical and physical CA number. A “jump” in the physical number is indication of a CA split. If there have been multiple splits from the original CA, you will see consecutive CA-SPLITs and a numeric value at the end of the split sequence. In Example 3-30, 8 CA splits have occurred from the original CA 0. The right hand column tells you the range of DSNs where this activity happened. Information about space utilization is also presented, including actual versus defined free space, dead CIs, and low use CIs and CAs.
## ALLOCATION

<table>
<thead>
<tr>
<th>SPACE TYPE</th>
<th>HI ALLO RBA</th>
<th>SPACE PRI</th>
<th>HI USED RBA</th>
<th>SPACE SEC</th>
<th>APPROX FREE CA'S</th>
</tr>
</thead>
<tbody>
<tr>
<td>CYLINDER</td>
<td>14,745,600</td>
<td>4</td>
<td>14,745,600</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## VOLUME 1

<table>
<thead>
<tr>
<th>VOLSER</th>
<th>PHY BLOCK SIZE</th>
<th>HI ALLO RBA</th>
<th>EXTS THIS VOL</th>
<th>DEVTYPE</th>
<th>PHY BLOCKS/TRK</th>
<th>HI USED RBA</th>
<th>EXTENT TYPE</th>
<th>VOLFLAG</th>
<th>PHY CA SIZE</th>
<th>CYLS ALLOC</th>
<th>SMALLEST EXTENT</th>
<th>LARGEST EXTENT</th>
<th>% UTIL PER TRACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONSMS</td>
<td>4,096</td>
<td>43,008</td>
<td>43,008</td>
<td>3390-3</td>
<td>21</td>
<td>43,008</td>
<td>DATA</td>
<td>PRIME</td>
<td>15</td>
<td>0.07</td>
<td>1</td>
<td>4</td>
<td>73</td>
</tr>
</tbody>
</table>

### Attributes

<table>
<thead>
<tr>
<th>KEYLEN</th>
<th>RECORD SIZE</th>
<th>CA SIZE</th>
<th>CISIZE</th>
<th>RKP</th>
<th>SHR(3,4)</th>
<th>NOIMBED</th>
<th>NOREPLICATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>2,041</td>
<td>43,008</td>
<td>2,048</td>
<td>9</td>
<td>NOIMBED</td>
<td>NOREPLICATE</td>
<td></td>
</tr>
</tbody>
</table>

### Statistics

- REC TOTAL: 0
- SEQ SET SPLITS: 0
- EXCPs: 1
- INDEX:
- REC DELETED: N/A
- IND SET SPLITS: N/A
- EXTENTS: N/A
- LEVELS: N/A
- REC INSERTED: N/A
- APPROX FREE CI'S: N/A
- LAST UPDATED: N/A
- REC UPDATED: N/A
- APPROX FREE CI'S: N/A
- HI LEVEL RBA: 4,096

### Control Area Analysis Report <2>

<table>
<thead>
<tr>
<th>CA Number</th>
<th>Notes</th>
<th>Free</th>
<th>CA</th>
<th>Dead</th>
<th>CI</th>
<th>Cmpresd</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0</td>
<td>86</td>
<td>&gt;</td>
<td>69</td>
<td></td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>18</td>
<td>90</td>
<td>&gt;</td>
<td>52</td>
<td></td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>4</td>
<td>4</td>
<td>139</td>
<td>59</td>
<td>&gt;</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>7</td>
<td>7</td>
<td>144</td>
<td>79</td>
<td>&gt;</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>5</td>
<td>5</td>
<td>151</td>
<td>71</td>
<td>&gt;</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>8</td>
<td>8</td>
<td>89</td>
<td>61</td>
<td>&gt;</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>19</td>
<td>19</td>
<td>87</td>
<td>63</td>
<td>&gt;</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>9</td>
<td>9</td>
<td>142</td>
<td>31</td>
<td>&gt;</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>10</td>
<td>10</td>
<td>53</td>
<td>26</td>
<td>&gt;</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>1</td>
<td>1</td>
<td>54</td>
<td>10</td>
<td>62</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>3</td>
<td>3</td>
<td>61</td>
<td>47</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>12</td>
<td>12</td>
<td>89</td>
<td>98</td>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>17</td>
<td>17</td>
<td>25</td>
<td>61</td>
<td>11</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>11</td>
<td>11</td>
<td>78</td>
<td>34</td>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>2</td>
<td>2</td>
<td>90</td>
<td>38</td>
<td>7</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>6</td>
<td>6</td>
<td>172</td>
<td>78</td>
<td>15</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>14</td>
<td>14</td>
<td>178</td>
<td>58</td>
<td>0</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>15</td>
<td>15</td>
<td>179</td>
<td>79</td>
<td>0</td>
<td>15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Control Area High Key Value

<table>
<thead>
<tr>
<th>CA Number</th>
<th>Notes</th>
<th>Free</th>
<th>CA</th>
<th>Dead</th>
<th>CI</th>
<th>Cmpresd</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>Control Area High Key Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0</td>
<td>86</td>
<td>&gt;</td>
<td>69</td>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td>*BACKUP.WKLY.FDRNON</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>18</td>
<td>90</td>
<td>&gt;</td>
<td>52</td>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td>*BRMI.TODISK.TOL.IVPINCR.O.C01V0056...</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>4</td>
<td>4</td>
<td>139</td>
<td>59</td>
<td>&gt;</td>
<td>7</td>
<td></td>
<td></td>
<td>*BRMI.TODISK.TOL.LRECL18.I.C01V0018</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>7</td>
<td>7</td>
<td>144</td>
<td>79</td>
<td>&gt;</td>
<td>6</td>
<td></td>
<td></td>
<td>*BRMI.TODISK.TOL.LRECL45.I.C01V0017</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>5</td>
<td>5</td>
<td>151</td>
<td>71</td>
<td>&gt;</td>
<td>7</td>
<td></td>
<td></td>
<td>*BRMI.TODISK.TOL.LRECL69.I.C01V0017</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>8</td>
<td>8</td>
<td>89</td>
<td>61</td>
<td>&gt;</td>
<td>9</td>
<td></td>
<td></td>
<td>*BRM4101.P18892T.LOAD</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>19</td>
<td>19</td>
<td>87</td>
<td>63</td>
<td>&gt;</td>
<td>11</td>
<td></td>
<td></td>
<td>*CATX.CAT6101.PF163...</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>9</td>
<td>9</td>
<td>142</td>
<td>31</td>
<td>&gt;</td>
<td>8</td>
<td></td>
<td></td>
<td>*CATX.CAT6202.PJ00089.L...</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>10</td>
<td>10</td>
<td>53</td>
<td>26</td>
<td>&gt;</td>
<td>11</td>
<td></td>
<td></td>
<td>*DEMOAGG.DEMOSDSL.TODISK.D.C01V000...</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>1</td>
<td>1</td>
<td>54</td>
<td>10</td>
<td>62</td>
<td>12</td>
<td></td>
<td></td>
<td>*HSMACT.H1.ABACKUP.DS1CC.D02213.T1B...</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>3</td>
<td>3</td>
<td>61</td>
<td>47</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td>*HSMACT.H1.MIGLOG.D06014.T065041</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>12</td>
<td>12</td>
<td>89</td>
<td>98</td>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
<td>*HSMACT.H1.MIGLOG.D06247.T220008</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>17</td>
<td>17</td>
<td>25</td>
<td>61</td>
<td>11</td>
<td>11</td>
<td></td>
<td></td>
<td>*INFRAX.MSC.PF16675.INSTJCL</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>11</td>
<td>11</td>
<td>78</td>
<td>34</td>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
<td>*MFAX.V</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>2</td>
<td>2</td>
<td>90</td>
<td>38</td>
<td>7</td>
<td>7</td>
<td></td>
<td></td>
<td>*PROD.PRODSTS.TODISK.C.C01V0033...</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>6</td>
<td>6</td>
<td>172</td>
<td>78</td>
<td>15</td>
<td>15</td>
<td></td>
<td></td>
<td>*SIS.DVLP.A000132...</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>14</td>
<td>14</td>
<td>178</td>
<td>58</td>
<td>0</td>
<td>15</td>
<td></td>
<td></td>
<td>*SIS.DVLP.A0002805...</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>15</td>
<td>15</td>
<td>179</td>
<td>79</td>
<td>0</td>
<td>15</td>
<td></td>
<td></td>
<td>*SIS.DVLP.A0004152...</td>
</tr>
</tbody>
</table>
CA Split Distribution Report <3>

Cumulative percent of CA splits..............24%....53%...100%

Number of CA splits 4 5 8
at one location

Occurred this many times 1 1 1

Resulting in this many CA splits 4 5 8

Which is this % of......24%....29%....47%
all CA splits

Index Structure Report <4>

Key size distribution:

<table>
<thead>
<tr>
<th>Minimum key size</th>
<th>Average key size</th>
<th>Maximum key size</th>
<th>DEF CL keylen</th>
<th>Standard deviation</th>
<th>Count of keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>9.32</td>
<td>42</td>
<td>45</td>
<td>6.08</td>
<td>1,473</td>
</tr>
</tbody>
</table>

Sequence Set record - key compression statistics:
Average bytes of SS record used 41%
Average front compression bytes 16
Average rear compression bytes 23
Average resulting compressed key 9
Maximum 'average' compressed key 12

*** For the INDEX CISZ analysis below, the 2 largest key(s) have been ignored due to CA's that have a used CI count which is not sufficient for an accurate sample ***

INDEX CISZ Analysis:
INDEX CISZ Ok? YES (See Report <6>: 'RECOMMENDATIONS')

INDEX Structure Summary Report

| INDEX CISZ Total Levels Total Records Index Set Records Sequence Set Read horizontally | Sequence Set Read vertically |
|---------------------------------|-----------------------------|-----------------------------|
| 2048                            | 2                           | 21                          | 1                           | 20            | 20             |

INDEX Record distribution:

<table>
<thead>
<tr>
<th>Level</th>
<th>Count</th>
<th>Resulting fan-out</th>
<th>Space (% of CI) available</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1:20</td>
<td>78%</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td></td>
<td>Min: 18% Max: 88% Typical: 40-49%</td>
</tr>
</tbody>
</table>

84 ICF Catalog Backup and Recovery: Catalog RecoveryPlus Update
Statistical Summary Report <5>

CREATION DATE-----26 MAY 2000     LAST UPDATE TIME--------N/A     LAST UPDATE DATE--------N/A
DAYS BETWEEN CRE/UPD--------N/A

DATA COMPONENT: RECSZ----------(4086 32400)     CISZ-------------------4,096
KEYS-------------------------45 9     FREESPACE-----------(0 0)
ALLOCATION---------------CYL(4 1)     SPANNED----------NOREUSE
CURRENT ALLOC SIZE-----20 CYL     EXTENTS-------------------17     VOLUME CYL--20 EXT--17
DEVICE TYPE--------3390-3     SHROPTIONS(3,4)--------SPEED

DATA COMPONENT CI FREESPACE ANALYSIS (BASED ON A RANDOM SAMPLING OF THE DATA RECORDS):
RECORDS SAMPLED---------762     COUNT OF SIZES-----------39     STD DEVIATION-----------262     MINIMUM REC SIZE---------78     AVERAGE REC SIZE--------193     MAXIMUM REC SIZE------1,641
MAX RECORDS PER CI-------20     REQUESTED CI FSPC %-------0     CI FSPC COST IN CYL-----0.2
% UTILIZATION OF CI------94     ACHIEVED CI FSPC %--------0
RECORDS LOADED W/FSPC----20     WASTED CI FSPC %----------6

DATA COMPONENT CA FREESPACE ANALYSIS:
CI/CA-----------------------180     CA SIZE (TRK)-------------15     CA FSPC COST IN CYL-----0.0
INITIALLY LOADED CI/CA-----180     INITIALLY FREE CI/CA------0     TOTAL FSPC COST IN CYL--0.2

DATA COMPONENT CI/CA SPLIT ANALYSIS:
LISTCAT-CI SPLITS------8,119     MAP-DETECTED CA SPLITS----17     RATIO CI:CA SPLITS-----427:1
LISTCAT-CA SPLITS---------19     CA SPLITS - % OF FILE----567

LOGICAL RECORD ANALYSIS:
REC-TOTAL-----------------0     INITIAL RECORD------222,628     REC-TOTAL-----------------0
REC-DELETED---------222,628     PERCENT DELETED---------100     REC-DELETED---------------0
REC-INSERTED--------------0     PERCENT INSERTED----------0     REC-INSERTED--------------0
REC-UPDATED---------------0     NET GAIN (LOSS)---(222,628)     REC-UPDATED---------------0
REC-RETRIEVED-------------8     LOGICAL I/O REQ-----445,264     LOGICAL I/O REQ-----------0
EXCP'S-----------------------1     EXCP'S----------------------1     RATIO: LOGICAL:EXCP-----N/A

DATA COMPONENT SPACE ALLOCATION ANALYSIS:
TOTAL ALLOCATED CA'S IN THE FILE---20   TOTAL ALLOCATED CYL---20   PERCENT OF 4 GB-----0.3
TOTAL USED CA'S IN THE FILE--------20   PERCENT USED------100     EQUAL TO HURBA/CASZ?--YES
TOTAL FREE CA'S IN THE FILE--------0   PERCENT FREE----------0     OVERALL FSPC IN CYL--15.9
TOTAL DEAD CA'S (MASSIVE DELETIONS)-1   PERCENT OF USED CA'S---5     EXPRESSED IN CYL----1.0
TOTAL CI'S IN THE FILE----------3,600   (INCLUDING FREE & DEAD CA'S)
TOTAL CI'S TO 'HURBA'----------3,600   (IN USED CA'S ONLY)
TOTAL CI'S TO 'HURBA'--------1,473   PERCENT USED--------41     EXPRESSED IN CYL----8.2
TOTAL FREE CI'S TO 'HURBA'-----2,127     PERCENT FREE-------59     EXPRESSED IN CYL----11.8
TOTAL FREE CI'S TO HARBA-------2,127     PERCENT OF TOTAL FILE--59     EXPRESSED IN CYL--11.8
ORIGINAL FREE CI'S PER CA--------0     AVERAGE FREE CI'S PER CA--103     PERCENT OF ORIGINAL---0
TOTAL DEAD CI'S (BAD INDEX CISZ)--187     PERCENT OF TOTAL FILE--5     EXPRESSED IN CYL--1.0

*** DEAD CI'S FROM 1 CA WERE NOT INCLUDED IN THE 'AVERAGE FREE CI'S PER CA' ***
BECAUSE THEY OCCURRED IN CA'S THAT ARE DEAD DUE TO MASSIVE DELETIONS

CATALOG: USERCAT.UCAT2

RECOMMENDATIONS <6>
There are several instances of CAs with dead CIs. They do not occur often enough to warrant a new INDEX CISZ, which is global across the entire file. However, for 100% utilization of the DATA component, a larger INDEX CI could be selected.

Suggested non shared resources (NSR) buffering values:
Sequential processing (read access): BUFND=27 BUFNI=1 (total buffer storage req 110.0K)
(load mode) : BUFND=27 BUFNI=2 (total buffer storage req 112.0K)
Direct key processing: BUFND=2 BUFNI=2 (total buffer storage req 12.0K)
Mixed (SEQ/DIR) processing: BUFND=27 BUFNI=2 (total buffer storage req 112.0K)

FUNCTION complete. Return code 0

3.13 Case study #13: GENERATE

Example 3-31 shows how to use the CR+ GENERATE command in the original MCNVTCAT mode to recreate catalog connector records from a master catalog. The report indicates that 33 catalog connector records with 255 aliases were found. The output file has 288 entries in it: 33 IMPORT CONNECT commands and 255 DEFINE ALIAS commands. Because JOBCARD was specified, JCL for IDCAMS is included and so are the commands to be used as SYSIN.

Example 3-31 CR+ GENERATE command

```
1  GENERATE  BCS-UNLOAD  -
2          BCS( -
3             SYSS80.MASTER.CATALOG -
4          ) -
5          NOCANDIDATE -
6          INCLUDE-TYPE( -
7             USERCATALOG -
8             USERCATALOG-ALIAS -
9          ) -
10         JOBCARD -
11         MCNVTCAT -
12         OUTFILE(GENOUT)
```

------------------------------

FUNCTION complete. Return code 0

THE FOLLOWING CATALOG ENTRY TYPES WILL BE HANDLED AS FOLLOWS:

- NVSM - NONVSAM - Excluded
- NALI - NONVSAM-ALIAS - Excluded
- GDG - GDG-BASE - Excluded
- GDS - GENERATION-DATASET - Excluded
- GALI - GENERATION-DATASET-ALIAS - Excluded
- UCAT - USERCATALOG - Included
- UALI - USERCATALOG-ALIAS - Included
- SYM - SYMBOLIC-RELATE - Excluded
- PAGE - PAGE-DATASET - Excluded
CL - CLUSTER - Excluded
CPTH - CLUSTER-PATH - Excluded
AIX - ALTERNATEINDEX - Excluded
APTH - AIX-PATH - Excluded

CAT22124I Default in effect: EXCLUDE-TYPE(NONE)
CAT22163I OUTPUT FILE IS: EM02.GENOUT2
CAT22164I INPUT BCS IS: SYSS80.MASTER.CATALOG
CAT22110I 06:12:50 GENERATE BCS-UNLOAD EXECUTION STARTING
CAT22110I 06:12:50 GENERATE BCS-UNLOAD EXECUTION ENDING

CAT22111I BCS RECORD TYPE SUMMARY:

<table>
<thead>
<tr>
<th>Type</th>
<th>READ</th>
<th>EXCL</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVSM</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>NALI</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GDG</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GDS</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GALI</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>UCAT</td>
<td>33</td>
<td>0</td>
<td>33</td>
</tr>
<tr>
<td>UALI</td>
<td>255</td>
<td>0</td>
<td>255</td>
</tr>
<tr>
<td>SYM</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PAGE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CL</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CPTH</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AIX</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APTH</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

CAT22113I TOTAL OUTPUT ENTRIES GENERATED: 288
CAT22114I TOTAL OUTPUT LINES GENERATED: 2385
CAT22009I GENERATE FUNCTION COMPLETE. RETURN CODE 0
CAT01008I Commands processed: 1
CAT01009I Catalog RecoveryPlus execution complete. Highest return code was 0

***** The following is the beginning of the output file EM02.GENOUT2 *****

//jobname JOB (acct),name,CLASS=A,MSGCLASS=X 00000100
//* 00000200
//* AMS CONTROL STATEMENTS FOR CATALOG: SYSS80.MASTER.CATALOG 00000300
//* 00000400
//IDC00001 EXEC PGM=IDCAMS 00000500
//SYSPRINT DD SYSOUT=* 00000600
//SYSIN DD * 00000700
/*/ ---- ID=1 ---------------------------------------------------- */ 00000800
IMPORT CONNECT - 00000900
OBJECTS( - 00001000
  ( CRSUCAT.ISPF.MERGCAT - 00001100
    DEVICETYPE( 3390 ) - 00001200
    VOLUMES( SESMS2 ) - 00001300
  ) - 00001400
) - 00001500
CATALOG( SYSS80.MASTER.CATALOG ) ; 00001600
/*/ ---- ID=2 ---------------------------------------------------- */ 00001700
IMPORT CONNECT - 00001800
OBJECTS( - 00001900
  ( CRSUCAT.ISPF.TEST1 - 00002000
    DEVICETYPE( 3390 ) - 00002100
    VOLUMES( SESMS2 ) - 00002200
  ) - 00002300
) - 00002400
ICF Catalog Backup and Recovery: Catalog RecoveryPlus Update

) -
CATALOG( SYSS80.MASTER.CATALOG ) ;
/* ---- ID=3 ------------------------------------------ */
DEFINE -
ALIAS( -
  NAME( CRSSI1 ) -
  RELATE( CRSUCAT.ISPF.TEST1 ) -
) -
CATALOG( SYSS80.MASTER.CATALOG ) ;
/* ---- ID=4 ------------------------------------------ */
DEFINE -
ALIAS( -
  NAME( CRSSI2 ) -
  RELATE( CRSUCAT.ISPF.TEST1 ) -
) -
CATALOG( SYSS80.MASTER.CATALOG ) ;
/* ---- ID=5 ------------------------------------------ */
DEFINE -
ALIAS( -
  NAME( CRSSI3 ) -
  RELATE( CRSUCAT.ISPF.TEST1 ) -
) -
CATALOG( SYSS80.MASTER.CATALOG ) ;
Related publications

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 90.

- DFSMS/MVS Recovery Management, GG24-3922
- Enhanced Catalog Sharing and Management, SG24-5594
- IBM TotalStorage Enterprise Tape: A Practical Guide, SG24-4632
- ICF Catalog Backup and Recovery: A Practical Guide, SG24-5644

Other resources

These publications are also relevant as further information sources:

- MVS System Commands, SA22-7629
- MVS System Management Facilities (SMF), SA22-7630
- DFSMS Access Method Services for Catalogs, SC26-7394
- DFSMSrmm Implementation and Customization Guide, SC26-7405
- DFSMS Managing Catalogs, SC26-7409
- DFSMS Using Data Sets, SC26-7410
- DFSMSshm Storage Administration Guide, SC35-0421
- DFSMSshm Storage Administration Reference, SC35-0422
- DFSMSdss Storage Administration Guide, SC35-0423
- DFSMS OAM Planning, Installation and Storage Administration Guide for Tape Libraries, SC35-0427
- Mainstar Catalog RecoveryPlus User Guide, document number 006-0202
  This publication is available from Mainstar Software Corporation, which you can find on the Web at: http://www.mainstar.com/

Referenced Web sites

You can refer to the following sources for relevant information:

- Catalog and VSAM Knowledge Base site:
  http://www.ibm.com/support/troubleshooting/us
  - Select “Storage”
  - Select “Storage Software”
  - Under “Other Software Products” select DFSMS
  - Select “Technical Notes”
- Mainstar Software Corporation: http://www.mainstar.com
How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services
This glossary contains a list of terms and abbreviations used within this redbook.

**access method control block (ACB)**  A control block that links an application program to VSAM or VTAM.

**access method services (AMS)**  A multifunction service program that manages both VSAM and non-VSAM data sets and ICF catalogs. It defines data sets and allocates space for VSAM data sets and ICF catalogs. It converts indexed-sequential (ISAM) data sets to key-sequenced data sets; modifies data set attributes in the catalog; reorganizes data sets; facilitates data portability among operating systems; creates backup copies of data sets and indexes; helps make inaccessible data sets accessible; lists the records of data sets; and catalogs, defines, and builds alternate indexes.

**ACS routine**  See *automatic class selection routine*.

**alias**  An alternative name for an entry or for a member of a partitioned data set (PDS).

**alias entry**  An entry that relates an alias to the real entry name of a user catalog or non-VSAM data set.

**automated tape library (ATL)**  A device consisting of robotic components, cartridge storage areas, tape subsystems, and controlling hardware and software together with the set of tape volumes that are in the library and can be mounted on the library tape drives.

**automatic class selection (ACS)**  A mechanism for assigning SMS classes and storage groups.

**automatic class selection routine**  A procedural set of ACS language statements. Based on a set of input variables, the ACS language statements generate the name of a predefined SMS class, or a list of names of predefined storage groups, for a data set.

**availability**  For a storage subsystem, the degree to which a data set or object can be accessed when requested by a user.

**backup data set**  A copy that can be used to replace or reconstruct a damaged data set.

**base cluster**  A key-sequenced data set or entry-sequenced data set over which one or more alternate indexes are built.

**basic catalog structure (BCS)**  The name of the catalog structure in the integrated catalog facility environment. See also *ICF catalog*.

**catalog**  A data set that contains the extensive information that is required to locate other data sets, to allocate and deallocate storage space, to verify the access authority of a program or operator, and to accumulate data set usage statistics. See also *master catalog* and *user catalog*.

**catalog address space (CAS)**  A separate address space in virtual storage that contains catalog management modules and control blocks.

**catalog connector**  A catalog entry, called either a user catalog entry or a catalog connector entry, in the master catalog that points to a user catalog. Contains information about aliases for the user catalog.

**cell**  An occurrence of information such as passwords, volume information, or associations.

**cluster**  In VSAM, a named structure consisting of a group of related components. For example, when the data is key-sequenced, the cluster contains both the data and the index components; for data that is entry-sequenced, the cluster contains only data components.
cluster entry  A catalog entry that contains information about a key-sequenced or entry-sequenced VSAM cluster: ownership, cluster attributes, and the passwords and protection attributes for the cluster. A key-sequenced cluster entry points to both a data and an index entry. An entry-sequenced cluster entry points to a data entry only.

component  The components of an object are usually referred to as the data component and index component. Also, the cluster, data, or index fields of a subrecord.

control area (CA)  A group of control intervals that is used as a unit for formatting a data set before adding records to it. Also, in a key-sequenced data set, the set of control intervals pointed to by a sequence-set index record adjacent to its data.

control interval (CI)  A fixed-length area of auxiliary storage space where VSAM stores records. It is the unit of information (an integer multiple of block size) transmitted to or from auxiliary storage by VSAM.

CR+  Catalog RecoveryPlus.

D

DASD volume  A DASD space identified by a common label and accessed by a set of related addresses.

data component  The part of a VSAM data set, alternate index, or catalog that contains the data records for the object.

data entry  A catalog entry that describes the data component of a cluster, alternative index, page spaces, or catalog. A data entry contains the data component attributes, allocation and extent information, and statistics. A data entry for the data component of a cluster or catalog can also contain the passwords and protection attributes for the data component.

Data Facility Sort (DFSORT™)  An IBM licensed program that is a high-speed data processing utility. DFSORT is an efficient and flexible way to handle sorting, merging, and copying operations and provides versatile data manipulation at the record, field, and bit level.

data facility storage management subsystem (DFSMS)  An operating environment that helps automate and centralize the management of storage. To manage storage, SMS provides the storage administrator with control over data class, storage class, management class, storage group, and automatic class selection routine definitions.

data set  In DFSMS, the major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access. In non-z/OS UNIX® System Services/MVS environments, the terms data set and file are generally equivalent and sometimes are used interchangeably. See also file.

DFSORT  See Data Facility Sort.

DFSMS  See data facility storage management subsystem.

DFSMSdftp™  A DFSMS functional component or base element of z/OS that provides functions for storage management, data management, program management, device management, and distributed data access.

DFSMSdss™  A DFSMS functional component or base element of z/OS, used to copy, move, dump, and restore data sets and volumes.

DFSMSshm™  A DFSMS functional component or base element of z/OS that is used for backing up and recovering data and managing space on volumes in the storage hierarchy.

DFSMSrmm™  A DFSMS functional component or base element of z/OS that manages removable media.

E

enhanced catalog sharing (ECS)  Sharing protocol that stores the information that describes changes to a shared catalog in the coupling facility. The I/O required for the VVDS mode protocol is eliminated, resulting in better sysplex-wide performance.

entry  A collection of information about a cataloged object in a master or user catalog.
entry name  A unique name for each component or object as it is defined in a catalog. The entry name is the same as the dsname in a DD statement that describes the object.

entry sequence  The order in which data records are physically arranged (according to ascending RBA) in auxiliary storage without respect to their contents.

entry-sequenced data set (ESDS)  In VSAM, a data set whose records are loaded without respect to their contents, and whose RBAs cannot change. Records are retrieved and stored by addressed access, and new records are added at the end of the data set.

export  To create a backup or portable copy of a VSAM cluster, alternative index, or ICF catalog.

extended addressability (EA)  The ability to create and access a VSAM data set that is greater than 4 GB in size. Extended addressability data sets must be allocated with DSNTYPE=EXT and EXTENDED ADDRESSABILITY=Y.

extended format (EF)  The format of a data set that has a data set name type (DSNTYPE) of EXTENDED. The data set is structured logically the same as a data set that is not in extended format but the physical format is different.

extent  A continuous space allocated on a DASD volume occupied by a data set or portion of a data set. An extent of a data set contains a whole number of control areas.

F

field  In a record or a control block, a specified area used for a particular category of data or control information.

file  A collection of information treated as a unit. In z/OS non-UNIX environments, the terms data set and file are generally equivalent and are sometimes used interchangeably.

filtering  The process of selecting data sets based on specified criteria. These criteria consist of fully or partially-qualified data set names or of certain data set characteristics.

G

generation data group (GDG)  A collection of historically related non-VSAM data sets that are arranged in chronological order; each data set is called a generation data set.

generation data set (GDS)  One of the data sets in a generation data group; it is historically related to the others in the group.

global resource serialization (GRS)  A component of z/OS used for serializing use of system resources and for converting hardware reserve on DASD volumes to data set enqueues.

I

ICF catalog  A catalog that is composed of a basic catalog structure (BCS) and its related volume tables of contents (VTOCs) and VSAM volume data sets (VVDSs).

import  To restore a VSAM cluster, alternative index, or integrated catalog facility catalog from a portable data set that was created by the EXPORT command.

index entry  A catalog entry that describes the index component of a key-sequenced cluster, alternative index, or catalog. An index entry contains the index component attributes, passwords, allocation and extent information, and statistics.

indexed VTOC  A volume table of contents with an index that contains a list of data set names and free space information, which allows data sets to be located more efficiently.

Integrated Catalog Forward Recovery Utility (ICFRU)  Program offering that provides a mechanism to forward recover an ICF catalog from a backup copy.

K

key  One or more characters within an item of data that are used to identify it or control its use. As used in this publication, one or more consecutive characters taken from a data record that are used to identify the record and establish its order with respect to other records.
key sequence  The collating sequence of data records determined by the value of the key field in each of the data records. It can be the same as or different from the entry sequence of the records.

key-sequenced data set (KSDS)  A VSAM data set whose records are loaded in key sequence and controlled by an index. Records are retrieved and stored by keyed access or by addressed access, and new records are inserted in the data set in key sequence because of free space allocated in the data set. Relative byte addresses of records can change because of control interval or control area splits.

linear data set (LDS)  A VSAM data set that contains data but no control information. A linear data set can be accessed as a byte-addressable string in virtual storage.

LRECL  Logical record length.

master catalog  A catalog that contains extensive data set and volume information that VSAM requires to locate data sets, to allocate and deallocate storage space, to verify the authorization of a program or operator to gain access to a data set, and to accumulate usage statistics for data sets.

multilevel alias facility (MLA)  A function in catalog address space that allows ICF catalog selection based on one to four data set name qualifiers.

non-SMS volume  A volume that is not controlled by SMS.

non-VSAM entry  A catalog entry that describes a non-VSAM data set. A non-VSAM entry contains the volume serial number and device type of the data set. If the data set is in a magnetic tape volume, the entry can also identify the data set file number.

non-VSAM volume record (NVR)  A VVDS record that contains SMS-related information about a non-VSAM, system-managed data set.

object  A named style stream having no specific format or record orientation.

object access method (OAM)  An access method that provides storage, retrieval, and storage hierarchy management for objects and provides storage and retrieval management for tape volumes contained in system-managed libraries.

partitioned data set (PDS)  A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

partitioned data set extended (PDSE)  A system-managed data set that contains an indexed directory and members that are similar to the directory and members of partitioned data sets. A PDSE can be used instead of a partitioned data set.

portability  The ability to use VSAM data sets with different operating systems.

portable data set  A data set that can be transported between systems with access method services.

record  A set of data treated as a unit.

record level sharing (RLS)  An extension to VSAM that provides direct record-level sharing of VSAM data sets from multiple address spaces across multiple systems. Record-level sharing uses the S/390® Coupling Facility to provide cross-system locking, local buffer invalidation, and cross-system data caching.

recovery  The process of rebuilding data after it is damaged or destroyed, often by using a backup copy of the data or by reapplying transactions recorded in a log.

relative byte address (RBA)  The displacement of a data record or a control interval from the beginning of the data set it belongs to; independent of how the data set is stored.
storage management subsystem (SMS)  A DFSMS facility that automates and centralizes the management of storage. Using SMS, a storage administrator describes data allocation characteristics, performance and availability goals, backup and retention requirements, and storage requirements to the system through data class, storage class, management class, storage group, and ACS routines definitions.

system-managed storage  Storage managed by SMS. SMS attempts to deliver required services for availability, performance, and space to applications.

system-managed tape library  A collection of tape volumes and tape devices, defined in the tape configuration database. A system-managed tape library can be automated or manual.

system management facilities (SMF)  A component of z/OS that collects input/output (I/O) statistics, provided at the data set and storage class levels, which helps you monitor the performance of the direct access storage subsystem.

tape configuration database (TCDB)  One or more volume catalogs used to maintain records of system-managed tape libraries and tape volumes.

tape library  A set of equipment and facilities that support the tape environment of an installation. This can include tape storage racks, a set of tape drives, and a set of related tape volumes mounted on those drives.

user catalog  An optional catalog used in the same way as the master catalog and pointed to by the master catalog. It lessens the contention for the master catalog and facilitates volume portability.

user catalog connector  See catalog connector.

volume table of contents (VTOC)  A table on a direct access volume that describes each data set on the volume.

virtual storage access method (VSAM)  An access method for indexed or sequential processing of fixed and variable-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry-sequence), or by relative-record number.

VSAM record level sharing  See record level sharing.

VSAM volume control record (VVCR)  The first logical record in the VVDS that contains information to manage DASD space and the BCS back pointers.

VSAM volume data set (VVDS)  A data set that describes the characteristics of VSAM and system-managed data sets on a given DASD volume; part of an ICF catalog.

VSAM volume record (VVR)  A VSAM logical record within a VVDS.
Index

A
alias 25
alternate master catalog 39, 67

B
BACKUP command 9
mask 17
Catalog RecoveryPlus BACKUP 49
EXPORT handling 49
forward recovery 50
BCS backup and forward recovery 50
buffers 8

C
Catalog 19, 42–43
allocation parameters 23
lock 3
maintenance 7
physical allocated size 23
renaming 23
reorganize 23
repair 23
Catalog RecoveryPlus
BACKUP BCS command 16
BACKUP command 15, 49
BACKUP VVDS command 18
Backup VVDS panel 9
DIAGNOSE ALIAS command 25, 54
DIAGNOSE command 31
introduction 8
ISPF interface 9
Main Menu panel 9
MERGECAT command 36
RACF FACILITY class profiles 14
RECOVER command 23
recovery simulation 30
Submit Backup panel 9
SUPERCLIP command 39
ZAP command 40
Catalog RecoveryPlus BACKUP command 58
catalogs vii
CATSCRUB command 46
changing a disk volser 62
Control Area Analysis Report 82
CR+ BACKUP BCS command
alias processing 22
backing up all catalogs 17
different frequencies 17
EXCLUDEBCS keyword 16
explicitly-specified BCSs 16
OUTFILE keyword 16
processing 22
CR+ BACKUP command
ACCEPTDIAGNOSE keyword 19
ACCEPTEXAMINE keyword 19
DIAGNOSEBCS keyword 19
DIAGNOSEVVDS keyword 19
duplex BCS 22
duplex VVDS 22
EXAMINE keyword 19
multiple backup copies 20
NORESERVE keyword 20
OUTDATASET keyword 15
OUTFILE keyword 15
RESERVE keyword 20
specifying BCSs 16
CR+ BACKUP VVDS command
EXCLUDEVVDS keyword 18
OUTFILE keyword 18
specifying VVDSs 18
CR+ DIAGNOSE ALIAS command
DEFINE-ALIAS keyword 33
DELETE-ALIAS keyword 33
error analysis 33
EXCLUDE keyword 33
INCLUDE keyword 33
NONPEER mode 32
PEER mode 32
CR+ DIAGNOSE command
ALLRELATED keyword 34
DEFINE-RECATALOG keyword 34
DELETE-ENTRIES keyword 34
DIAGNOSE ALIAS 32
DIAGNOSE BCS-VVDS 34
DIAGNOSE VVDS-BCS 34
fix commands 31
selecting BCSs and VVDSs 34
CR+ EXPLORE command 81
CR+ GENERATE command 86
CR+ MAP command 82
CR+ MERGECAT command
alias handling 37
alternate master catalog 39
BACKOUT keyword 37
COPYONLY keyword 36, 39
ENTRIES keyword 37
journal file 37
LOCK keyword 39
NOLOCK keyword 39
RESTART keyword 37
SIMULATE keyword 38
CR+ MERGECAT COPYONLY 70
CR+ RECOVER command
alias handling 25
BCS rename 25
explicit/implicit DEFINE 24
FORWARD keyword 27
INDATASET keyword 23
INFILE keyword 23
LOCK keyword 25
MASTERCATALOG keyword 25
NEWNAME keyword 25
NOLOCK keyword 25
PRINT keyword 29
PRINTing BCS records 29
re-sizing the VVDS 31
SIMULATE keyword 30
specifying BCS or VVDS 24
CR+ REORG command 79
CR+ SUPERCLIP command, changing a volser 39
CR+ ZAP command 70
deleting a BCS and VVDS record 41
patching a BCS and VVDS records 42
printing BCS and VVDS records 41
C-type (Cluster) record 73

D
data 33
data set vii, 5, 8, 15, 17–18, 20–26, 31, 33–37, 39,
42–43, 46, 49, 51, 54, 56, 58, 60–62, 66–68, 71, 75–77,
79, 81, 91–95
DIAGNOSE ALIAS
output 55
DIAGNOSE VVDS command 57
disaster recovery plan 8
duplicate key 17

E
enlarging VVDS 60
ensuring good backups 48
error-free backup 2
exclusive enqueue 20

F
FIXDATASET keyword 54
forward recovery 2, 24

G
GENERATE command 86

I
ICF catalog vii, 1–3, 6–8, 15, 22, 31, 44, 47, 91, 93–95
ICFRRAP program 2, 5
ICFRRSSV program 2, 5
ICKDSF utility 39
IDCAMS commands
ALTER LOCK 25
ALTER UNLOCK 6
DEFINE CLUSTER 24
DEFINE USERCATALOG 24
DELETE RECOVERY 24

S
SIMULATE keyword 54
SMF data
forward recovery 51
SMF records 2, 26
structural error 48
structural integrity 19
SUPERCLIP command 39, 62
SIMULATE mode 66
synchronizing aliases 54
T
T-type (Truename)  73

U
user  17
user catalog  17, 21, 32–33, 36–38, 45–46, 54–57, 73, 82, 91–92, 95
user catalog connections  32

V
volser change  39
volser clip  39
volume reserve  20
VVDS  7–9, 15–16, 18–24, 26–31, 34–36, 38–42, 54, 58–61, 70–74, 76–77, 92–93, 95
VVDS backup and forward recovery  57
VVDS backup copy  60
VVDS backups  8
VVDS forward recovery  59
ICF catalogs are essential system data sets. Even with the high availability storage subsystems and processors that are available today, there are still situations where you need to recover your catalogs. You should keep your catalogs healthy and be prepared for a recovery situation. Also, you must be sure that your recovery procedures do not have a big impact on your production environment. To minimize the recovery process, you should have a clear backup and recovery strategy and the proper utilities.

IBM Integrated Catalog Forward Recovery Utility (ICFRU) and Mainstar Catalog RecoveryPlus are two of the available tools you can use to recover your catalogs. ICFRU is a basic tool to help you in a forward recovery situation. It does not offer a wide range of features, but it is useful for catalog recovery. Mainstar Catalog RecoveryPlus offers a set of features to help you with your ICF catalog environment maintenance.

This IBM Redpaper explains how to use each of these products in a catalog recovery situation. It also offers storage administrators with useful recommendations for implementing a catalog backup and recovery plan. It does not compare the two products but instead shows how each of them work. This paper also provides practical tests and examples that can help you with the different error scenarios that you might find in your daily production activities.