

An ICF Catalog

White Paper

From Mainstar Software Corporation



A Tricky ICF Catalog Problem

Solved With Mainstar:® Catalog RecoveryPlus™ – and Some Fast Thinking!

By Geoff Littlewood - with special thanks to Elliot Hamilton for his technical support assistance.

Preface: Quick thinking by Mainstar's Level 3 Technical Support saved the day for a prospective customer. It was a major bank in Europe, and by chance, the storage administrator was conducting a trial evaluation of **Catalog Recovery Plus** at the time he encountered an ICF catalog failure. He had recently installed **Catalog Recovery Plus** to compare its functions and performance against another vendor's product - and as it turned out, he also got a chance for a real trial-by-fire of Mainstar's highly acclaimed technical support - he was extremely impressed with the speed of response and the technical solution Mainstar provided.

The date was November 28th, 2002, Thanksgiving Day in the U.S., but even though this was a national holiday, Mainstar's 24x7x365 support is always available for emergencies – and this was an emergency!

This White Paper highlights how the user's "event" evolved, and how Mainstar's European field SE Support and U.S. headquarters Technical Support staff assisted this user in a time of crisis. This paper also presents technical aspects that could very well be of value for anyone maintaining ICF catalogs.

The User Discovers a Catalog Problem

The scenario: This user had a major problem with a production ICF catalog (in this White Paper the user catalog will be named UCAT.A) that was prohibiting access to any of the catalog's data sets. The cause of the catalog problem was found to be a totally corrupted index component.

Worse, though, this user found that his existing backups for this catalog, created by IDCAMS EXPORT, were actually invalid and couldn't be used to recover the catalog, even though the backup job had given a zero return code.

At this stage, the user wasn't too worried, because he had just installed **Mainstar: Catalog RecoveryPlus**, and was aware that its BACKUP BCS command does not use the index of an ICF

catalog to retrieve the catalog's data records. In fact, during the backup, **Catalog RecoveryPlus** avoids any of the problems within the index that plague EXPORT, retrieving the records from the catalog's data component as if it was an ESDS, in physical record sequence. Later, when executing the **Catalog RecoveryPlus** RECOVER command, the data records are sorted into ascending key sequence before reloading them into a newly-defined BCS.

So What Happened Next?

Knowing that **Catalog RecoveryPlus** does not read the index for backing up a catalog, the user decided to delete the corrupted index component of the ICF catalog, thinking that it wouldn't be used anyway, and that **Catalog RecoveryPlus** would just use the data component that remained. To accomplish this, he ran the following job:

```
//S1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//DISK DD
UNIT=SYSALLDA,DISP=OLD,VOL=SER=CAT00A
//SYSIN DD *
DELETE 'UCAT.A.CATINDEX' VVR
FILE(DISK)
/*
```

Big mistake - our user soon learned this was not what he should have done! The IDCAMS DELETE job worked perfectly - actually too perfectly - as it did exactly what it was told to do - physically deleting the index component of the BCS from the volume's VVDS, VTOC, and VTOCIX, but leaving the data component intact and on the volume. This left the BCS' data component physically residing on the volume, but orphaned without an associated index component.

The user next tried a **Catalog RecoveryPlus** BACKUP command, to attempt a backup of the catalog without using the index. He tried several iterations of this, using different BACKUP control statement parameters.

The first attempt was:

```
//S1      EXEC PGM=CAT00010
//SYSPRINT DD  SYSOUT=*
//DUMPDD DD  DISP=OLD,DSN=CAT.BACKUP
//SYSIN DD *
  BACKUP BCS(UCAT.A) -
    OUTFILE(DUMPDD) -
    ACCEPTEXAMINE(I) -
```

```
EXPLICIT-VERIFY(YES) -
DIAGNOSEBCS -
ACCEPTDIAGNOSE(W)
```

//

In looking at the output of this job (see *Figure 1*), the user began to have that sinking feeling that this is not going to be a good day.

```
CAT02145I START OF DATA UNLOAD FOR UCAT.A ON 28 NOV 2002 (2002332) AT 10.20.55
CAT02192I CALLING IDCAMS TO PERFORM VERIFY
IDC3300I ERROR OPENING UCAT.A
IDC3351I ** VSAM OPEN RETURN CODE IS 148
IDC3003I FUNCTION TERMINATED. CONDITION CODE IS 12
CAT02150W IDCAMS (2) ISSUED RETURN CODE 12
CAT02192I CALLING IDCAMS TO PERFORM EXAMINE
IDC3300I ERROR OPENING UCAT.A
IDC3351I ** VSAM OPEN RETURN CODE IS 145
IDC31700I VSAM OPEN ERROR
IDC3003I FUNCTION TERMINATED. CONDITION CODE IS 12
CAT02150W IDCAMS (4) ISSUED RETURN CODE 12
CAT02167W BCS BACKUP BYPASSED DUE TO IDCAMS EXAMINE RETURN CODE

CAT02147I END OF DATA UNLOAD FOR UCAT.A ON 28 NOV 2002 AT 10.20.56

CAT02009I BACKUP FUNCTION COMPLETE. RETURN CODE 4
CAT01008I COMMANDS PROCESSED: 1
CAT01009I Catalog RecoveryPlus EXECUTION COMPLETE. HIGHEST RETURN CODE WAS 4
```

Figure 1 - Output from first attempt to run a backup of the catalog without using the index.

Undeterred, the user took a careful look at the **Catalog RecoveryPlus** manual to determine if there was a different set of parameters he could use to get **Catalog RecoveryPlus** to read just the data component of the BCS. On his second attempt, he tried:

```
BACKUP BCS (UCAT.A) -
  OUTFILE(DUMPDD) -
  EXCP-MODE -
  NOEXAMINE -
  NODIAGNOSEBCS
```

Reasoning that **Catalog RecoveryPlus** couldn't use a standard OPEN on a BCS with a missing index, the user selected EXCP-MODE,

NOEXAMINE, and NODIAGNOSEBCS, on the theory that these parameters might tell BACKUP to entirely avoid using the index. The reasoning was good, but still not correct. Normal VSAM OPEN processing is not issued when EXCP-MODE is coded, but instead, uses proprietary EXCP-level I/O logic to dump all tracks on which the BCS is allocated, and locates the EOF within the object being backed up.

Unfortunately, even with these parameters, the **Catalog RecoveryPlus** BACKUP still did not work. As shown in *Figure 2*, zero records were dumped, and the user's situation was getting more desperate.

```
CAT02145I START OF DATA UNLOAD FOR FACUCAT.EH.MRGC2.CAT ON 28 NOV 2002 (2002332) AT 10.31.58
CAT02146I RECORD SUMMARY FOR FACUCAT.EH.MRGC2.CAT
  CLUSTER  GDG      E      J      NONVSAM  TRUENAME  PATH  UCAT  ALIAS  OTHER  TOTAL
         0      0      0      0          0          0      0      0      0      0
MASTER CATALOG ALIASES BACKED UP                2
CAT02151I NUMBER OF SPANNED RECORDS ENCOUNTERED: 0
CAT02147I END OF DATA UNLOAD FOR FACUCAT.EH.MRGC2.CAT ON 28 NOV 2002 AT 10.31.58
CAT02139I CLOSE COMPLETE FOR OUTPUT DDNAME=DUMPFIL
CAT02009I BACKUP FUNCTION COMPLETE. RETURN CODE 8
CAT01008I COMMANDS PROCESSED: 1
CAT01009I Catalog RecoveryPlus EXECUTION COMPLETE. HIGHEST RETURN CODE WAS 8
```

Figure 2 - Output from second attempt to run a backup of the catalog without using the index. This time using EXCP-MODE, NOEXAMINE, and NODIAGNOSEBCS.

Next Step - Contact Mainstar

As Mainstar's Senior Technical SE for Europe, the initial e-mail from this user was directed to me. I replied that the **Catalog RecoveryPlus** BACKUP command has situations where an index record might be required for BACKUP processing, even though it does not *normally* use the index, and therefore, the physical presence of an index is mandatory for BACKUP to properly function (because BACKUP doesn't use the index for normal record access, it does not suffer from the problems that EXPORT has with index structural problems). The user then sent a reply e-mail with several attachments showing the jobs and commands he had used so far in his attempts to recover this catalog.

When I opened the first attachment, showing the output from the job which deleted the index, the penny immediately dropped, and I began to understand that he had a real problem on his hands. By explicitly deleting the index component of the BCS, the **Catalog RecoveryPlus** BACKUP command could not process this catalog even in EXCP-MODE, as having an index component physically allocated and available is absolutely necessary to unload the data component - no matter what parameters are coded.

My first response was to send a reply e-mail, suggesting that he should recover from his most recent EXPORT backup of the catalog (as **Catalog RecoveryPlus** RECOVER can accept either its own BACKUP format, or that of EXPORT). He could then use RECOVER, with SMF records to forward recover from this previous backup of the catalog, bringing it up-to-date and in synchronization with its cataloged data sets. Imagine my horror when he replied that he had now discovered none of his EXPORT backups for this catalog were valid, due to EXPORT backing up with a return code of zero, when in fact, the structural problems in the BCSs index were precluding valid backups.

At this point, I knew I had to involve Mainstar's Technical Support staff in the U.S. I first sent all the documentation to techsupport@mainstar.com, and then telephoned the Emergency Hotline Number (remember, this was Thanksgiving Day) to get support for this trial customer. This emergency number is manned 24 hours a day, even on holidays, and within two minutes I was re-directed to talk to one of Mainstar's Level 3 Tech Support staff members for **Catalog RecoveryPlus**. He was already looking at the problem documentation I had sent, and the phone call discussion that followed quickly convinced him how critical the situation was for my potential customer - that we needed to help him recover this catalog as quickly as possible, using

what was left of his current catalog, as it was all he had.

What Could Mainstar Tech Support Do?

Level 3 now understood the situation about this catalog as:

- It was corrupted and totally unusable.
- It was critical to their production systems.
- A valid backup copy did not exist.
- The index had been deleted.

By happy coincidence, the Level 3 person was intimately familiar with the BACKUP logic, and he began exploring off-the-wall and out-of-the-box ways to get around the fact that BACKUP couldn't process the catalog in its present state.

His analysis explored how a valid index could be re-created, and came up with an idea that proved to be quite simple in principle, and reasonably easy to execute using **Catalog RecoveryPlus** - but highly unusual

His idea was - create a dummy catalog, then modify the dummy catalog's index entry in the VVDS so that it becomes the surrogate index for the damaged catalog. He then set out to test this procedure, and e-mailed the results and commands back to me - *within the space of two hours*.

Here are the procedure steps he came up with:

Step 1: Create a new dummy catalog, on the same volume as the corrupted catalog, with a name very similar to the name of the corrupted catalog. He suggested choosing a name for this dummy catalog in which just a single character of its name is changed (such as UCAT.B).

Step 2: Modify the VVR in the VVDS for the index component of the dummy catalog, such that all references to the catalog name are changed to the name of the corrupted catalog. This is done to trick catalog management into believing that the structure of the original catalog is physically present, intact, and correct. Remember, **Catalog RecoveryPlus** does not normally read the information in the index component, however, it does require the presence of a valid VVR record for the index component, in case spanned records are present (in this case, we were hoping there were no spanned records, and in fact, none were present in the catalog).

This second step can be done fairly simply and safely using the ZAP PRINT command of **Catalog RecoveryPlus**, to print the index VVR of the dummy catalog:

```
ZAP VVDS(CAT00A) PRINT
      (COMPONENT(UCAT.B.CATINDEX))
```

Note: The 2nd qualifier of this catalog name is defined as B, identifying it as the dummy catalog.

Executing this command produced the output shown in *Figure 3*, and you can readily see, at the right hand side of the listing, that the catalog name occurred four times.

```
*****
VVDS DATA FOR DSN: UCAT.B.CATINDEX
  RBA FOUND: 00005744
  TYPE FOUND: Z
*****
00000000 01680030 E9280000 000010E4 C3C1E34B *...Z.....UCAT.*
00000010 C24BC3C1 E3C9D5C4 C5E70007 E4C3C1E3 *B.CATINDEX..UCAT*
00000020 4BC20006 E4C3C1E3 4BC207E4 C3C1E34B *.B..UCAT.B.UCAT.*
00000030 C2000096 2140A000 00000000 00000100 *B..O. ....*
00000040 00008000 00080000 00A80000 00000000 *.....Y.....*
00000050 00FFFFFF FFFFFFFF FF000000 00080000 *.....*
```

Figure 3 - Output from ZAP PRINT command to print the index VVR of the dummy catalog.

Step3: Determine the displacements of the CHANGED character in the catalog's name in each of these four places within this VVR record. In this

case (see *Figure 4*), it was the second character (Z) of each occurrence of the catalog name.

```
00000000 01680030 E9280000 000010E4 C3C1E34B *...Z.....UCAT.*
00000010 C24BC3C1 E3C9D5C4 C5E70007 E4C3C1E3 *B.CATINDEX..UCAT*
++-displacement to ZAP is X'10', the byte value 'B' to an 'A'
00000020 4BC20006 E4C3C1E3 4BC207E4 C3C1E34B *.B..UCAT.B.UCAT.*
++-displacement to ZAP is X'21', the byte value 'B' to an 'A'
++-displacement to ZAP is X'29', the byte value 'B' to an 'A'
00000030 C2000096 2140A000 00000000 00000100 *B..O. ....*
++-displacement to ZAP is X'30', the byte value 'B' to an 'A'
```

Figure 4 - These displacements describe the character that will be changed, resulting in the index of the newly defined catalog becoming the surrogate index of the original catalog, and are only valid for this particular catalog name.

Step4: Once the displacements are identified, create the appropriate ZAP control statements, as follows. The VER compare value X'C2' (or character B) is the value shown in the ZAP PRINT output. The REP change-to value, X'C1' (or character A) will convert the name to match that of the original catalog's deleted index component.

```
ZAP VVDS(TSTSM2)PATCH(RBA(X'5744') -
                        VER(X'0029',X'C2') -
                        REP(X'0029',X'C1'))
```

```
CAT08200I COMP: UCAT.A.CATINDEX
CAT08206I RECORD MODIFIED: UCAT.A.CATINDEX
CAT08000I ZAP FUNCTION COMPLETE.
RETURN CODE 0
```

```
ZAP VVDS(TSTSM2)PATCH(RBA(X'5744') -
                        VER(X'0010',X'C2') -
                        REP(X'0010',X'C1'))
```

```
ZAP VVDS(TSTSM2)PATCH(RBA(X'5744') -
                        VER(X'0030',X'C2') -
                        REP(X'0030',X'C1'))
```

```
CAT08200I COMP: UCAT.B.CATINDEX
CAT08206I RECORD MODIFIED: UCAT.B.CATINDEX
CAT08000I ZAP FUNCTION COMPLETE.
RETURN CODE 0
```

```
CAT08200I COMP: UCAT.A.CATINDEX
CAT08206I RECORD MODIFIED: UCAT.A.CATINDEX
CAT08000I ZAP FUNCTION COMPLETE.
RETURN CODE 0
```

```
ZAP VVDS(TSTSM2)PATCH(RBA(X'5744') -
                        VER(X'0021',X'C2') -
                        REP(X'0021',X'C1'))
```

```
CAT08200I COMP: UCAT.A.CATINDEX
CAT08206I RECORD MODIFIED: UCAT.A.CATINDEX
CAT08000I ZAP FUNCTION COMPLETE.
RETURN CODE 0
```

Step 5: Now the customer could proceed with his BACKUP EXCP-MODE NOEXAMINE job. The surrogate index is completely wrong from a KSDS construct point of view, but EXCP-MODE will tolerate this condition. Most likely, BACKUP (normal mode) would also work, but in the event spanned records are encountered, a severe error could occur, which would stop the BACKUP.

The last few steps are as follows:

Step 6: DELETE the original catalog.

Step 7: DEFINE the replacement for the original catalog.

Step 8: Execute RECOVER BCS(...)

Step 9: Use EXPORT DISCONNECT for the dummy catalog (whose index became the surrogate for the old catalog).

Step 10: DELETE the new catalog's data and index components, using DELETE...VVR. Both the data and the index must be deleted, as the data VVR and its VTOC Format 1 DSCB still exist, as does the index VTOC Format 1 DSCB which also exists.

Conclusion

This problem was unusual in that the customer had intentionally deleted the index of an ICF catalog, thinking he was doing the right thing to achieve a backup, not realizing that he was actually creating an untenable situation. Worse, due to the nature of the index error, his IDCAMS EXPORT backups since the time of the index corruption were not valid, even though they were giving a return code of zero.

Without **Catalog RecoveryPlus** and Mainstar's Technical Support to the rescue, this user might have faced a very serious problem. Indeed - one that might have required a lot of manual recovery.

If you aren't a licensed user of Catalog RecoveryPlus then perhaps now is the right time to give it a try and evaluate the benefits for yourself. A free trial can be requested from our web site: www.mainstar.com.

Geoff Littlewood - Technical Sales Engineer for Mainstar Software Corporation.

Before joining Mainstar, Geoff worked for IBM in the UK for 31 years as a Senior Systems Engineer and as a Large Systems Storage Specialist. While with IBM, Geoff was a field SE and an instructor on Assembler programming, VSAM, and IMS. In addition, Geoff has provided support in many customer installations on HSM, SMS, and advanced storage products, such as IBM's RVA, Shark, and Virtual Tape Server Systems.

Mainstar is a Registered Trademark of Mainstar Software Corporation. Catalog RecoveryPlus is a trademark of Mainstar Software Corporation.