

An ICF Catalog White Paper

From Mainstar Software Corporation



System Programmer Maintenance for ICF Catalogs *Using the Catalog RecoveryPlus™ System Programmer Toolkit*

By Ronald K. Ferguson

Preface: Access Method Services (IDCAMS), and other storage management facilities within MVS, are designed with maximum safety of operation in mind - simply put, safety features that try to protect you from yourself.

For example, you cannot uncatalog data sets anymore, because SMS-managed data sets must be cataloged at all times. You cannot catalog a data set in more than one catalog because of the problems that result if a data set were to be accessed from two locations. You cannot delete a data set of the same name as one that is already allocated to the system, because there is no way to differentiate them.

The problem is, these are all things that happen in real life, and are pain-in-the-neck situations that, frankly, keep you from doing your job. There are ways to solve these problems, but some solutions may require significant time and effort.

Mainstar: Catalog RecoveryPlus was designed to let you do what you need to do - quickly and easily - with the caveat that you'd better know what you're doing (i.e., that you don't need to be protected from yourself).

This White Paper highlights the System Programmer Toolkit, a series of new commands within **Catalog RecoveryPlus**, designed specifically to help the MVS system programmer do the things that need to be done in day-to-day work on MVS system volumes, data sets, and catalogs. This paper presents some ideas for how to use these important new commands - when the going gets tough.

Deleting an Unwanted VVDS - Scenario 1

Here's a situation that many MVS system programmers might be familiar with:

You have just built a set of sysres volumes using IBM's ServerPac dialogs. The ServerPac job stream defined a SYS1.STGINDEX data set

on the primary IPL volume, and it is the only VSAM data set on the volume. As a result, a VVDS is automatically allocated on the sysres volume.

Your plan is to run with VIODSN=IGNORE in the IEASYS00 member in SYS1.PARMLIB, which means that SYS1.STGINDEX is not needed. You delete SYS1.STGINDEX without problems, but you are now left with an empty and unnecessary SYS1.VVDS data set on a non-SMS managed volume; taking up space on the tightly-packed IPL volume.

One procedure to remove the unwanted VVDS, is to execute an IDCAMS DELETE command, specifying SYS1.VVDS.Vvolser as the file to delete, with the FILE(ddname) and RECOVERY keywords. This command certainly gets the job done, but it requires a) intervention from the MVS console operator, and b) RACF ALTER access to the master catalog.

The alternative would be to run the following **Mainstar: Catalog RecoveryPlus** ALTER SYS1-VVDS command, as it does not have either of the above two requirements; the logic being that this operation has absolutely nothing to do with the master catalog, nor should it require anything from the console operator:

```
ALTER SYS1-VVDS      -  
      DSN(SYS1.VVDS.Vsysres) -  
      VOLSER(RES001) -  
      DELETE
```

Caution: You really need to know what you're doing if you run this command (whether you use the IDCAMS or the **Catalog RecoveryPlus** method), as you can very easily 'orphan' any VSAM data sets that may be on the volume. You should only run this if there aren't any VSAM data sets on the volume, or after generating pseudo SMF records in preparation for a specialized VVDS forward recovery, as described in Scenario 2, on the next page.

This command cannot be used to delete the valid VVDS from a volume (a valid VVDS is defined as one whose data set name matches the current volser of the volume). If you attempt it, you'll receive an error message, and the command fails.

This command can be run in SIMULATE mode, in case you are uncomfortable with its function and want to see what it will do before actually deleting the invalid VVDS.

Deleting an Unwanted VVDS - Scenario 2

Here's another scenario that you may have faced before:

A residual VVDS has been left on a newly-built sysres volume (possibly due to a Scenario 1 situation that was not dealt with). The sysres packs are cloned to a second set of volumes for maintenance propagation to another sysplex. After running a DFSMSdss full volume copy to the new volsers, residual VVDS data sets now exist on those volumes, and carry the source volser in their VVDS data set names, rather than the target volser.

In this scenario, the IDCAMS DELETE RECOVERY (that would have worked in Scenario 1) does not work. Here's what you'll get if you try it:

```
//DD1 DD DISP=SHR,UNIT=3390,
      VOL=SER=volser

DELETE SYS1.VVDS.Vvolser FILE(DD1)
RECOVERY
IDC2895I ALL REQUIRED VOLUMES NOT
        INCLUDED IN DD STATEMENT
        SPECIFIED IN FILE PARAMETER
IDC3014I CATALOG ERROR
IDC3009I ** VSAM CATALOG RETURN CODE
        IS 76 - REASON CODE IS
        IGGOCLFM-8
IDC0551I ** ENTRY SYS1.VVDS.Vvolser
        NOT DELETED
```

Furthermore, executing an IDCAMS DELETE command without the RECOVERY parameter causes an SVC dump on the system. Catalog Management is not designed to accept the idea that the VVDS's own self-defining VVR might be inaccurate (i.e., might be based on a volser that isn't the actual volser of the current volume on which the VVDS is physically located).

With standard MVS utility functions, the only (ill-advised) way to delete this VVDS is to ZAP the VTOC Format 1 DSCB for the VVDS itself, to change the data set type from VSAM to nonVSAM (which you really don't want to be doing), and then

scratch the VVDS with IEHPROGM or ISPF 3.4 (you cannot use IDCAMS DELETE, as it will not accept a delete request for any data set having a name that matches the rules for a VVDS).

With the **Catalog RecoveryPlus** System Programmer Toolkit ALTER SYS1-VVDS command, as shown in Scenario 1, you can delete this errant VVDS with no problem.

Deleting a Duplicate VVDS

There are several ways that you can find yourself with a duplicate VVDS on a volume; one VVDS being valid (VVDS name matches the actual volser of the volume), and the other being invalid (VVDS name does not match the volser).

This may commonly occur when you clone a volume that already has a VVDS on it, and changing the original volser (let's call it OLDVOL) to a new one (we'll call NEWVOL).

Another way for a duplicate VVDS to occur on a volume, is if someone simply clips a volume with a VVDS already allocated on it. Without realizing it, they now have a VVDS named to match the original volser.

In either case, you now have an invalid VVDS, named, for the previous volser. The second (and valid) VVDS will match the new volser, and all newly allocated data sets on the volume will be defined in it.

Also, if VSAM data sets existed on the volume at the time this occurred, their descriptive VVRs would be in the original VVDS, and since the VVDS's name doesn't match the actual volser of the volume, there's no way to access these data sets. At the very least, there will be a VVR in the VVDS, self-describing the VVDS itself.

Any new data set allocations will use the new VVDS, and you rapidly find yourself in the situation where some data sets on the volume can be accessed, and others can't. Worse, you can't delete the data sets you can't access, because Catalog Management doesn't recognize the VVDS on the volume. (Sure, you can use the 'ZAP the VTOC and use IEHPROGM' method mentioned above, but this is a slippery slope). Besides, the data sets in the original VVDS might be data sets that you want to keep.

Here's a more detailed description of the situation:

1. Volser OLDVOL has a VVDS allocated on it, with a name of SYS1.VVDS.VOLDVOL. This VVDS is perfectly valid at this time, and contains VVRs for all of the VSAM data sets allocated on this volume right now (as well as

NVRs for any nonVSAM data sets on the volume, if it is SMS-managed).

2. The volume is then cloned or clipped to NEWVOL, and now the VVDS' data set name does not match the new volser of the volume.
3. Furthermore, assuming all current data sets on the volume are cataloged, the catalog records for these data sets point to them on the original volser, OLDVOL, making the data sets inaccessible from a catalog search.
4. At the first allocation of a VSAM data set on the volume, we'll then have a second VVDS automatically allocated on the volume, and named SYS1.VVDS.VNEWVOL to match the current volser. For any subsequent data set allocations on the volume, this new VVDS will contain VVRs for all of the VSAM data sets allocated since it was renamed to NEWVOL, and if the volume is SMS-managed, NVR records for any new nonVSAM data sets allocated.
5. The original VVDS 'knows' nothing about the new data sets, and the new VVDS 'knows' nothing about the old data sets.
6. The self-defining VVR for each VVDS is inside itself, so therefore, the original VVDS doesn't 'know' about the new VVDS, and the new VVDS doesn't 'know' about the original VVDS.

This is another 'chicken and egg' paradox. Any attempt to clean up the volume will run into problems, unless you bite the bullet and entirely clean everything off the volume, reinitialize it to get proper VVDSs allocated, then reload the data sets. You cannot delete the old VVDS, because Catalog Management will not allow any data set with the name SYS1.VVDS.Vxxxxxx to be deleted or renamed, even if it is not correctly named for the volume. You cannot access any of the old VSAM data sets because you cannot access the original VVDS to get to them.

The solution? The **Catalog RecoveryPlus** command called ALTER SYS1-VVDS has a powerful feature for just this type of situation (coupled with the ALTER BCS-VOLSER command when it's time to change the volser values in the catalog).

ALTER SYS1-VVDS essentially creates pseudo SMF records from the old (invalid) VVDS, allowing you to 'forward recover' them into the new (valid) VVDS. You can then delete the invalid VVDS.

Here's a detailed description of what the ALTER SYS1-VVDS feature does:

1. Reads the contents of the original VVDS named SYS1.VVDS.VOLDVOL (standard OPEN will not work on this data set, do ALTER if using proprietary logic in accessing the VVDS).
2. Uses a very specialized forward recovery-type of technique, updating all of the original VVRs from the invalid VVDS, SYS1.VVDS.VOLDVOL, into the new, valid VVDS, named SYS1.VVDS.VNEWVOL. This facility makes all of the data sets that were previously accessible from the original VVDS, now accessible through the new VVDS.
3. Scratches SYS1.VVDS.VOLDVOL from the volume, or if you prefer, renames it to a &&TEMP naming convention, and converts it to a nonVSAM data set that no longer presents a problem because of its name. (This is a one-way rename - once it is named &&TEMP, it cannot be renamed back to a VVDS data set naming convention).

To execute this, here's what the ALTER command would look like:

```
ALTER SYS1-VVDS -  
      DSN(SYS1.VVDS.VOLDVOL) -  
      VOLSER(NEWVOL) -  
      SMFFILE(OUTFILE(SMFDD))
```

4. When this step is complete, you need to run the following **Catalog RecoveryPlus** ALTER BCS-VOLSER command to change all occurrences of the original volser OLDVOL to NEWVOL:

```
ALTER BCS-VOLSER -  
      BCS(SYST80.MSTRCAT) -  
      OLD-VOLSER(OLDVOL) -  
      NEW-VOLSER(NEWVOL)
```

Deleting ENQ'd Data Sets

When building IPL volumes, the system data sets you're working on often need to be reallocated (sometimes several times). However, many of these data sets will have the same names as the libraries of the currently running system, from which you are doing the system build (as opposed to the new volumes that are being built). Unfortunately, there are SYSDSN ENQs outstanding for the production libraries, and the system does not differentiate the volser that the production data sets reside on; and consequently, you cannot scratch or rename the new system data sets using normal methods.

The ALTER NONVSAM command is a situation-specific feature of **Catalog RecoveryPlus** that allows a scratch or rename of a nonVSAM, non-SMS managed data set, even when that data set's name (not the data set itself, but the data set name) is in use by another task. The function serves as an alternative to the SCRATCH and RENAME commands provided by IBM's IEHPROGM utility.

When **ENQ** is specified (or left to default), ALTER NONVSAM functions identically to the corresponding IEHPROGM commands, but with a command syntax that is much more forgiving.

The **NOENQ** parameter of the ALTER NONVSAM command allows a scratch or rename of a non-VSAM, non-SMS data set to occur when another task owns the SYSDSN ENQ for that data set name. All other integrity features, such as security and serialization are maintained just as they are for the IEHPROGM SCRATCH command.

Since the NOENQ parameter bypasses an existing integrity feature of the operating system, utmost care and good judgment must be used to ensure that the physical target data set is not in use, as ALTER NONVSAM cannot check for this. As a result of this, a decision was made to implement this command function in batch execution only, and an ISPF panel is not included for it.

To use the NOENQ parameter, you must have full access (ALTER) to the SAF CLASS=DASDVOL profile for the volume where the data set to be deleted resides. If SAF indicates that CLASS=DASDVOL is inactive, or that the specific volume is not protected by this class, then the NOENQ parameter is allowed, regardless of whether you have necessary access authority.

ALTER NONVSAM cannot be used for data sets that are VSAM components. It also cannot be used for data sets that are SMS-managed and/or reside on SMS-managed volumes.

Here is an example that might help illustrate how the process works:

Let's assume you are building a new TSTRES volume. You find you've run out of PDS directory blocks on the SYS1.HELP data set, and it is also in 16 extents. You need to delete it, reallocate it larger, and give it more directory blocks. However, the SYS1.HELP data set on PRDRES is currently allocated by all TSO users, and this prohibits your attempts to scratch the failing SYS1.HELP data set.

Here's the sequence of steps you'll need to execute:

Step 1 Allocate data set 'SYS1.HELPX' on the TSTRES volume, as large as you require, and with as many directory block as you'll need, for the final SYS1.HELP data set.

Step 2 Copy SYS1.HELP on TSTRES to the newly allocated SYS1.HELPX.

Step 3 Uncatalog SYS1.HELPX, with the following IDCAMS command:

```
DELETE SYS1.HELPX NOSCRATCH
```

Step 4 Run the following **Catalog RecoveryPlus** ALTER commands:

```
ALTER NONVSAM DSN(SYS1.HELP) -  
NEW-NAME(SYS1.HELP2) -  
VOL(TSTRES) NOENQ
```

```
ALTER NONVSAM DSN(SYS1.HELPX)-  
NEW-NAME(SYS1.HELP) -  
VOL(TSTRES) NOENQ
```

Step 5 Scratch the old SYS1.HELP2 data set using a normal IDCAMS DELETE command.

Setting Up Symbolic Volsers For SYSRES Creation

When preparing for a new sysres creation, using symbolic volsers is the only way to go. With the ALTER BCS-VOLSER command, there is support to change current volsers to symbolic using the NEW-VOLSER keyword.

The sample command below is a simulation run to check out whether the command will execute successfully when it is issued for real. When you are satisfied with the simulated results, all you would need do is remove the SIMULATE keyword, and the real-life command is ready to run.

The command will change all occurrences of IPLT81 in the catalog BCS SYST80.MSTR.CAT, to the symbolic value &SYSR1. The key of each record that is changed will be printed.

```
ALTER BCS-VOLSER -  
INCLUDE-BCS(SYST80.MSTRCAT) -  
OLD-VOLSER(IPLT81) -  
NEW-VOLSER(&SYSR1) -  
PRINT(KEY) -  
SIMULATE
```

MCNVTCAT Replacement

MVS system programmers who support ICF catalogs are lamenting the loss of the IBM MCNVTCAT from the CBIPOs. In a recent search of the IBM Main Listserv archives, there were 299 posts concerning this, so it is evident that many people would like to see this type of support continue. Well, now you have it - and the support is all that MCNVTCAT ever provided, and more!

The new GENERATE BCS-UNLOAD command in **Catalog RecoveryPlus** provides a fast, easy, and convenient means to achieve the support that MCNVTCAT provided. In simple terms, the command generates IDCAMS control statements from existing records in a specified BCS, enabling you to convert all current master catalog entries to a new master catalog when implementing new MVS systems.

To execute the GENERATE BCS-UNLOAD you specify an existing BCS for processing, identify the 'type' of BCS record for which you wish IDCAMS control statements to be generated (or all records if you wish), and they will be written to your specified output data set.

You can execute the GENERATE BCS-UNLOAD command multiple times within a single jobstep execution, directing different types of IDCAMS DEFINE statements to different output data sets or PDS members.

Extensive filters are available, for example:

INCLUDE-DSN | EXCLUDE-DSN – Identify specific data set records within the BCS to be included or excluded.

INCLUDE-TYPE | EXCLUDE-TYPE – Filter by certain types of catalog entries, such as:

ALTERNATEINDEX	AIX-PATH
ALL	CLUSTER
CLUSTER-PATH	GDG-BASE
NONE	NONVSAM
NONVSAM-ALIAS	PAGE-DATASET
SYMBOLIC-RELATE	USERCATALOG
USERCATALOG-ALIAS	VSAM

INCLUDE-VOLSER | EXCLUDE-VOLSER – Filter based upon the original volsers referenced by that entry.

SUBSTITUTE-VOLSER – Change the volser values, and/or change their devicetypes, for the generated output IDCAMS control statements, based upon the volser in the original catalog entry. This is useful when using system symbolics for nonVSAM entries that reside on the sysres volumes, and are cataloged in the master catalog.

Conclusion

The **Catalog RecoveryPlus** System Programmer Toolkit commands described in this White Paper are just the beginning. As operating system maintenance procedures and practices change, and as we hear from customers of new features and functions that are needed, we plan to incorporate them into the product. Our goal is to make **Catalog RecoveryPlus** the product that you turn to whenever any ICF catalog, DASD volume, or storage management function needs to be performed. Watch for more White Papers introducing you to new additions to the **Catalog RecoveryPlus** System Programmer Toolkit.

If you are already a licensed user of **Catalog RecoveryPlus**, I encouraged you to give these new commands a try, let us know what worked for you, what didn't, what improvements you would like to see, and information about any new commands that would be useful in your Catalog Management methodology.

If you aren't yet a licensed user of **Catalog RecoveryPlus**, it's time to swap out whatever tired, out-of-date product or procedures you're currently using, and give **Catalog RecoveryPlus** a try. A free trial is available and can be requested from our web site: www.mainstar.com.

Ronald K. Ferguson - Founder, President & CEO of Mainstar Software Corporation

Ron Ferguson has a technical background in large-scale MVS systems. As a software instructor for 20+ years, he has presented over 600 courses on VSAM and ICF catalogs, and is recognized worldwide as an expert in these areas. Ferguson travels widely, meeting with customers and presenting at national and international conferences.