

ICF Catalog

White Paper

From Mainstar Software Corporation



ICF Catalogs:

Best Practices with Catalog RecoveryPlus

By Ronald K. Ferguson

Preface: Many installations have ICF catalogs that are not optimally defined and managed. The problem is manifold: (a) many believe "If it ain't broke, don't fix it"; (b) changing catalog attributes requires 'quiet' time to re-org the catalog and scheduling catalog maintenance can be difficult; and (c) the word isn't getting out about the best and most widely accepted attributes and methodologies.

This White Paper highlights the importance of 'biting the bullet' to improve catalog definition attributes. It provides succinct recommendations to improve your catalog environment – for best performance, ease of day-to-day management, and position you so you can recover and fix catalog problems when (not if) they occur.

Defining Your ICF Catalog with Optimum Attributes

In many installations the catalogs have been around for decades. Over the years, many different people have managed them, and most likely, they've gone through several definition 'styles' as MVS opinions and installation configurations change. As a result, they have become a hodge-podge and, all too often, are untouched as a result of the adage, "If it ain't broke, don't fix it!"

Here are some rules of thumb that will help you get your catalog definitions back on track. (If a keyword attribute isn't mentioned below, the recommendation is to *not* specify it in your catalog DEFINE.)

Rule 1: Use a catalog naming convention

Having a good catalog naming convention is important. Your current number of ICF catalogs most likely grew over many years,

with different people involved in catalog naming conventions, and with evolving applications that share use of the catalogs.

A good naming convention ensures that all catalogs will be backed up, as even name masks in your catalog backup step can accidentally miss one or more catalogs if their name doesn't fit the mask (unless, of course, you're using ** as your mask). It also ensures against accidental damage to the catalog (let's call it a "finger check"). Finally, starting a naming convention and best practices project could improve your frame of mind – and skill set – for actively managing your catalogs.

Until **Catalog RecoveryPlus (CR+)** came along, changing the name of a catalog was difficult. The process required all applications accessing data sets through the catalog to be quiesced. The IDCAMS commands to accomplish it are difficult and time consuming to set up. But the biggest reason is, most people are afraid to touch their catalogs, thinking, "They're running just fine now, so let's leave them alone".

With the **CR+** RECOVER NEW-NAME command, changing a catalog's name is easily accomplished during a standard re-org of the catalog – and it's very safe, quick, and trouble-free. The process involves the following steps:

1. Quiesce all application tasks (batch and online) that are currently accessing data sets through the catalog (you'll likely want to run this during a scheduled maintenance time).
2. Issue a MODIFY UNALLOCATE command from all *other* systems that have shared access to the catalog – the syntax is:
F CATALOG, UNALLOCATE(*bcs.name*)

3. Execute the **CR+** BACKUP command to create a backup copy of the catalog. *Check the return code from this command before proceeding.*
4. Execute the **CR+** RECOVER command, specifying the current catalog name in the BCS keyword, and your new catalog name in the NEW-NAME keyword. With other *NEW-attribute* keywords, you can change virtually any attribute of the catalog, such as primary and secondary space allocation, data and index component CI size, CI and CA free space percentages, and buffer values.

CR+ RECOVER NEW-NAME updates the BCS back-pointers in all VVDS records for non-migrated disk data sets in the catalog. This results in command execution that is longer than for a simple restore operation, and you should allow time for this when you schedule your catalog outage. If the catalog is average size (let's say, not larger than 100 cylinders), you might plan on 10 to 15 minutes elapsed time for RECOVER NEW-NAME. One benchmark was a catalog with 3 million data sets, requiring just under one hour to rename. **CR+** uses a journal file to obtain best performance during this process, and you will need to allocate the journal file prior to the command, specifying the JOURNAL keyword on RECOVER.

Note: When using NEW-NAME, RECOVER *does not* delete the existing catalog (it does delete when NEW-NAME is *not* specified). If you use DFSMSHsm and there are any migrated data sets in the catalog, it is important to issue an IDCAMS IMPORT DISCONNECT for the old catalog immediately after the RECOVER command completes execution. Otherwise, DFSMSHsm recall of any migrated data set in the catalog will fail when it attempts to change the VOLSER of MIGRAT to the new VOLSER.

During standard RECOVER processing, the UCAT connector record in the master catalog on the system you execute RECOVER on will be updated to the new BCS name. If the BCS is shared across multiple systems – and you have individual master catalogs on those systems – you should specify the MASTERCATALOG keyword on the RECOVER command, specifying the additional master catalogs to be updated with

the new BCS name.

Rule 2: Specify CISZ of 4096

There are no hard and fast rules, but we (the VSAM technical experts at Mainstar) recommend 4096 for both the data and index component – and you'll need to explicitly specify the CISZ keyword on both components to ensure this.

For the data component, 4096 provides a good, all-around size that doesn't require excessive data transfer to read/write records, isn't so small that it results in a high number of spanned records, and matches up nicely with catalog address space buffers.

With the index component CISZ of 4096, coupled with the same value on the data component – and typical key compression on catalog keys – there's no risk of dead CI problems. Catalog keys (data set names) typically compress to 13 to 17 bytes, and multiplied by 180 CI/CA (which is what you get with a 4096 data CISZ), you can almost always fit 180 key/pointer entries into the sequence set record for each CA.

Rule 3: Provide generous primary and secondary allocation and know when to re-org

Provide a generous primary *and* secondary allocation, and – very important – *make sure every catalog has a secondary allocation value specified.* It's astonishing how many catalogs are currently allocated with just a primary value – CYL(xx) – presumably because of a long-ago belief that secondary allocations were a problem. Well, consider this: a catalog that is filling its primary and cannot obtain a secondary allocation *will result in an outage at the worst possible time* – could there be any worse performance problem than that? If you have any catalogs without a secondary value specified, cleaning them up now is reason enough to schedule a catalog re-org as soon as possible.

Contrary to popular belief, having a catalog in multiple extents is not a problem, and most likely not worth re-orging it to remove them. Chances are, if the catalog has extended into secondary extents, it has also completed CA splits *at the areas of activity*

inside the catalog, and re-orging will only remove them – and the splits will then have to be done over again. The only time a catalog should be re-orged due to extents is when it is in danger of hitting the maximum extent count (123, and it must reside on a single volume). If a catalog is relatively stable and sitting in, for example, 50 extents, that's simply not a problem. If, on the other hand, it's at 110 extents and growing by a new extent each week, you'd better get a re-org scheduled for sometime in the next 2 to 3 months (and also find out what is causing it to grow that rapidly).

The **CR+** REORG While Open command allows you to re-org the catalog while it's in an open state, so this will allow you to create some breathing space in keeping the catalog under control until you can get its attributes changed to better values.

Rule 4: Specify FREESPACE(0 0)

It's difficult to determine an optimum value, and most often, FREESPACE(0 0) is as good as anything you could specify.

Here's the basic problem: the key of the catalog is data set names. Knowing the names of new data sets, and the number and distribution of new record insertions across the current data set name keys is very unlikely. Whatever you specify for FREESPACE will be evenly distributed across the CIs and CAs – and it's doubtful that your actual record insertions will ever match the percentage amount and granularity you specify.

By specifying FREESPACE(0 0), you are accepting that your new record insertion pattern will likely be uneven, and therefore, CI and CA splits will be used to dynamically create your free space, where and when it is needed. Catalog management will suffer a very slight performance "hit" in the near term after a re-org of the catalog, but once the splits at the active location(s) are complete, you'll have lots of free space exactly where you need it. The key here is not to worry about the splits within the catalog, and let basic VSAM design work to your advantage.

Rule 5: Remove obsolete attributes IMBED and REPLICATE

These two options are going away, and

you should prepare for this now.

Both attribute keywords have been ignored on new DEFINE USERCATALOG commands since OS/390 2.10 – but here's the problem: any catalog that already has IMBED or REPLICATE specified will still have the attributes until the catalog is re-orged. IMBED and REPLICATE indicate physical placement of index records, so the attributes cannot be turned off by the IDCAMS ALTER command. IBM started warning users to remove these attributes several years ago now, but many installations have ignored this. In August 2004, IBM officially stated that any VSAM data set or catalog containing the IMBED and/or REPLICATE attribute will no longer open as of z/OS 1.8.

Rule 6: Use the correct SHAREOPTIONS value for level of sharing

SHAREOPTIONS values (3,3) or (3,4) are allowed for catalogs. Specifying the correct value is very important for performance, but even more importantly, for catalog integrity reasons.

The first subparameter, "cross-region" shared access control, defines the level of sharing within a system (LPAR). The value 3 (which is the only value allowed) indicates that any number of users can have the catalog open for both read and write, but it is the user's responsibility to provide protection against multiple updating. Since we're talking about a catalog here, this is not a problem, as the Catalog Address Space (CAS) will typically be the only task within the system that is open to the catalog, as all catalog updates take place from within the single address space, CAS. (Catalog management products, such as **CR+**, must utilize the proper interfaces with CAS to prevent damage to the catalog from read/update access in conjunction with CAS activity to the catalog. **CR+** has the necessary integrity mechanisms, allowing you to perform catalog management while the catalog is open to CAS on any number of systems, regardless of VVDS or ECS sharing support.)

The second subparameter, "cross-system" shared access control, defines the level of sharing across multiple systems (LPARs).

- The value 3 implies that you will not be sharing the catalog across multiple systems – and note the word “implies”. Catalog management within one system’s CAS assumes that all previous in-use buffers are valid and does not refresh them from the physical catalog. It also assumes that the in-storage control blocks for the catalog are accurate and in-synch with the real catalog, since no other system is expected to be accessing/updating the catalog while this system has it open. The value 3 provides the best performance, but if it is specified and the catalog *is* actually shared, corruption to the catalog itself, or out-of-date catalog records from buffer lookasides will be used. The result is, the catalog will be broken!
- The value 4 indicates the catalog is expected to be shared across multiple systems, and catalog management is expected to ensure integrity of both catalog access and update from all systems. Specifying a value of 4 does not, in itself, make a catalog shared, as it must also reside on a DASD volume defined as shared. If both conditions are met, CAS ensures that each system uses an up-to-date, valid copy of the catalog at all times, by reading the physical catalog records for each request.

Rule 7: ECSSHARING vs. VVDS-mode sharing

If your catalogs are shared across multiple systems in a parallel sysplex environment, and a coupling facility structure has been created, the ECSSHARING value can be specified (or ALTER’d) for the catalog. This attribute stands for Enhanced Catalog Sharing (ECS), and provides significant performance benefit for catalog access. More importantly, it maintains catalog integrity across the shared systems.

There is an IBM Redbook, *Enhanced Catalog Sharing and Management*, publication number SG24-5594, that explains ECS in very good detail and should be downloaded (for free) from the IBM web site. In this Redbook, there are some very interesting ECS sharing performance statistics benchmarks.

In one test, an IDCAMS LISTCAT ALL

was run against a master catalog containing 4,554 data set entries. In VVDS sharing mode, the total elapsed time was 30 seconds; when the master catalog was defined to ECS sharing mode, the elapsed time for the same command was 18 seconds.

In a second test, an IEFBR14 step was run to create and catalog 300 nonVSAM, nonSMS data sets. In VVDS sharing mode, the elapsed time was 18 seconds; in ECS sharing mode it was 12 seconds.

Should You Worry About CI and CA Splits? When Should You Re-Org?

The short answer is, don’t worry about splits within your catalog. As mentioned above in the discussion on FREESPACE, it’s just not possible to know ahead of time where record insertion activity will occur within your catalog, so letting CI and CA splits dynamically allocate the necessary free space – exactly where it is needed – is the best way to go.

Splits cause performance loss *only* at the time they occur – not afterwards. A CI split is probably on the order of 10 to 15 ms on today’s DASD; a CA split might cost upwards of 300 to 500 ms. Therefore, a split of either kind is a very tiny cost, for a very large benefit – not only at the time it is occurring, but in the future when other record insertions are made at the same location. In general, if you re-org to remove splits from the catalog, when the next record insertion comes along at that same location, the same split will have to be done over again.

This doesn’t mean that you shouldn’t keep an eye on CA split activity inside certain catalogs. Rapidly growing catalogs, catalogs that get a high number of data set deletions, and catalogs that experience the “creeping key” problem should be closely watched with the **CR+** MAP BCS command, and re-orged as necessary.

For example, one catalog we know of required re-org every few months, as it grew from 250 to 2,500 cylinders in size. There were 122 application aliases assigned to the catalog, and the re-org required every application to be quiesced. The cause of this was a high usage of an installation data set

naming convention using something we call 'pseudo GDGs' – in other words, not real MVS GDGs, but an application-generated numeric sequence number in the data set names created within each application. That resulted in high-volume, pinpoint clustering of new records at many locations across the catalog, causing CA splits in virtually every CA across the catalog. As with real GDGs, the applications then deleted the oldest (lowest numbered) data sets, creating empty (dead) spaces within the catalog at each data set name location. The end result was, the catalog grew larger and larger (10 times its original size), yet contained roughly the same number of records at all times.

To determine how to best manage this, the **CR+** MAP BCS command was run once a week, tracking with the reports exactly what was happening inside this catalog. These reports showed where record insertion and deletion activity was occurring, that CA split activity was evenly spread across the entire catalog, and the high number of record deletions were subsequently creating 'dead CAs' (where CA splits had originally occurred).

The solution was to separate the records in this catalog out into several catalogs (in other words, screw up the courage – and schedule the time – to run MERGECAT!). Having 122 application aliases on a single catalog is a recipe for disaster – lose or break that catalog and 122 applications are dead in the water. Another solution is to use the **CR+** REORG While Open command, allowing the catalog to be re-orged without quiescing the applications.

Catalogs used with the Mobius ViewDirect product (formerly InfoPac) are the ones most commonly found with the creeping key problem. Hundreds, possibly thousands of data sets are created every day and cataloged, all with the same first and second qualifiers in the data set name. By default, a date is in the 3rd level qualifier, and a timestamp is in the 4th qualifier. Well, the basic fact about a date is that it gets higher, day by day, and within each day, the time steadily increases in value until you roll over to the next day. Each of these newly cataloged data set names is inserted immediately after the previous one, resulting

in a "mod" type of record addition at (or near) the end of the catalog. Because the SYS1.VVDS.xxxxxx records in the catalog are usually higher in key value than the ViewDirect records, each insertion is in physical location immediately after the previous one, but in front of the first VVDS catalog record. This results in CI splits on a very frequent basis, followed by CA splits when each CA fills up, and the catalog grows very rapidly. Throughout the catalog, though, the space utilization is a mere 25%, as every used CI is just 50% full, and each used CA is just 50% full of CIs (and 50% of 50% is 25%).

At the point the catalog reaches the maximum space you can afford for it, there's no choice but to re-org. If ViewDirect is a 24x7 application, it must be quiesced for the re-org. If you have **CR+**, you can run REORG While Open on the catalog while it is still active with ViewDirect.

Some installations delete ViewDirect reports as fast as they create them, and this results in creeping deletions right behind the insertions. In this case too, the catalog gets larger and larger, yet is virtually empty.

There's also a ViewDirect solution to this: the product can be installed with an option to reverse the 3rd and 4th qualifiers in the generated data set names. This randomizes the record insertions/deletions across the catalog (because, for example, 10:00 AM occurred yesterday, today, and will occur again tomorrow, and all 10:00 AM records, for any day, will be in the same general location inside the catalog). The downside is, you no longer have LISTCAT capability on the catalog records by date value – if that is important.

Spread the Load Across Your Catalogs – Don't Invite Disaster

Most z/OS systems have hundreds of thousands, and more likely millions, of data sets cataloged in a mere handful of catalogs. The typical number of catalogs on a system is 25 to 100. Do the math. Assume you have 1 million cataloged data sets and 25 catalogs (a fairly common ratio). That works out to 40,000 data sets – and that's assuming the spread of data sets across catalogs is even.

In the best of worlds, if any one of the 25 catalogs suffers an outage during production processing, access to 40,000 data sets is lost until the catalog is recovered.

The trouble is, data sets are rarely distributed evenly across catalogs. Most often, there are just a handful of catalogs that have a high percentage of the system's application data sets cataloged in them.

To find out what the catalogs' environment looks like on your systems, you simply need to look at the **CR+** BACKUP BCS output – the summary page provides a list of each catalog and the number of total records in it, but you'll need to go through the detail of each BCS backup report to determine the number of aliases connected to it.

For an example of a typical catalog configuration, see *Figure 1*. The system has a total of 1.3 million data sets, and 25 user catalogs. Given these numbers, you'd expect about 55,000 data sets per catalog (which, in itself, is high). Instead, UCAT #1 has a whopping 377,441 data sets cataloged in it, representing 27% of the total data sets on the entire system. That catalog has 3 aliases, and unless there's further imbalance within these aliases, the logical solution would be to split all data sets for two of the aliases out to two other, brand new user catalogs.

UCAT #2 isn't much better, having almost 250,000 data sets cataloged in it – or 18% of the total data sets on the system – but note how many aliases this user catalog has: 141. Chances are, these are not TSO userid aliases, but rather, real application alias values that represent production data sets within that catalog. Splitting this catalog into a reasonable number of new catalogs would be a project in itself, but one that would be well worth it to lessen the impact of an outage to this catalog.

As a final note on this catalog environment, UCATs #1 to 4 as an aggregate, have over 1 million data sets in them – or 73% of the total data sets on the system.

The solution? In rough terms, for this particular catalog a reasonable goal might be to split out the first 7 user catalogs, so that no single catalog has more than 30,000 records in it. This might increase the total number of

catalogs by about 40, for a new total of 65 catalogs, but that certainly isn't an unreasonable number of catalogs. In the same project, standardizing on a good catalog naming convention would be another goal, enabling easy backup of these catalogs with the **CR+** BACKUP BCS command through a mask value.

Figure 1: Typical catalog configuration

	# DATA SETS	# OF ALIASES	% OF TOTAL
UCAT #1	377,441	3	27%
UCAT #2	247,703	141	18%
UCAT #3	216,289	57	16%
UCAT #4	159,021	40	12%
UCAT #5	85,531	34	6%
UCAT #6	70,621	15	
UCAT #7	65,061	361	
UCAT #8	40,072	19	
UCAT #9	35,527	272	
UCAT #10	21,488	10	
UCAT #11	12,163	5	
UCAT #12	11,928	15	
UCAT #13	10,968	75	
UCAT #14	7,013	10	
UCAT #15	5,939	200	
UCAT #16	3,059	1	
UCAT #17	1,510	1	
UCAT #18	1,061	3	
UCAT #19	618	2	
UCAT #20	385	3	
UCAT #21	219	5	
UCAT #22	11	1	
UCAT #23	9	1	
UCAT #24	5	1	
UCAT #25	3	0	
Total data sets	1,373,645		
# of catalogs	25		
# of aliases	1275		
Avg data sets/catalog	54,946		
Top 4 total	1,000,454	241	73%

CAS Caching – ISC versus VLF

ICF catalog caching (a \$2 word for lookaside buffering) comes in two 'flavors'. All open catalogs on the system can use one or the other of the caching techniques, but an individual catalog cannot use both techniques simultaneously. If you do not define ISC or CDSC for a catalog, neither technique is used.

For either technique, the following caching rules apply:

- For a user catalog, only records accessed by key are cached.
- For a master catalog, records accessed either sequentially or by key are cached, except for alias records, which are kept in the alias table.

ISC Caching (ISC)

ISC establishes cache buffers that reside within CAS. Catalogs that are best suited for ISC caching are those that are *infrequently updated*, such as a master catalog that is rather inactive (although a *shared* master catalog is not a good candidate as it can have degraded performance within ISC). With ISC, each assigned catalog is allocated a fixed amount of buffer space within the cache area (except for master catalogs, which do not have a fixed amount), and utilizes the least-recently-used (LRU) logic for “stealing” buffer area when the space is used up.

A major problem with ISC is, any update to a shared access catalog detected from another system causes the complete ISC cache to be purged on the current system. The buffer space dedicated to caching for ISC is also very limited for each catalog.

Catalog Data Space Caching (CDSC)

The CDSC is more commonly known as VLF (Virtual Lookaside Facility), as it is defined through the COFVLFxx member of SYS1.PARMLIB, and uses cache buffers that are established in a data space by issuing the START VLF operator command. With CDSC (VLF), a fixed caching area is not used. Instead, a catalog caches records until no space is left in the data space cache. Once the data space is full, then LRU logic is used to make room for new records. CDSC is best

suited for highly volatile (i.e., frequently updated) catalogs, such as user catalogs for data sets used in heavy-access batch and online systems, and also for shared master catalogs.

When VLF is used, only the actual records that were updated on the other system(s) are purged from the VLF cache. VLF allows better tuning, based on actual access patterns for that catalog.

(Note: For z/OS 1.4, there is a HIPER APAR, OA06920, with a title of “Performance degraded after VLF stops IRRACEE caching”. There are indications of a serious performance problem without this APAR, so ensure you have it on your system if you are z/OS 1.4.)

Keep Your Catalogs Clean

There is no substitute for catalogs that are error free.

Many people say they haven't had a broken catalog for years, only to find that one or more catalogs *are* currently broken – it's just a matter of defining what 'broken' means. For many people, a broken catalog means a catalog that's dead in the water. In fact, if a catalog has structural damage within it, but only causes problems for certain types of situations – well, it's broken!

For example, a catalog might work without any problems on a daily basis, but when you attempt to do a LISTCAT ALL on it, it fails. Chances are, you have a broken sequence set chain pointer, that only causes problems when you sequentially access the catalog (which IDCAMS LISTCAT does), but doesn't cause problems for keyed searches for a data set locate.

Another possibility is a catalog that doesn't exhibit any problems – until you attempt a MERGECAT. When the MERGECAT operation fails in the middle of it because of some garbage record that you never access on a daily basis, you have a problem on your hands. When a MERGECAT operation fails in the middle of execution (assuming it's a LEVEL-type of request), you are in never-never land, with some of the records for the LEVEL now in the target catalog, and some still in the source catalog.

Until you either back out the failed MERGECAT, or fix the problem and then restart to get it to run to completion, you have an outage on your hands for the application(s) assigned to the LEVEL (in other words, an entire application).

The solution to this is to run all possible diagnostics on your catalogs, as frequently as you can afford them. This includes running an EXAMINE on every BACKUP BCS command execution and preferably specifying the DATATEST option – it's a fraction more expensive than INDEXTEST, but it will identify errors that INDEXTEST won't detect. Set up a monthly BACKUP BCS(**) command that specifies DIAGNOSE-BCS, so that all of your BCSs are thoroughly diagnosed inside themselves (this is a logical record synchronization diagnostic, rather than a physical structure test). Run DIAGNOSE BCS-VVDS and VVDS-BCS on a regular basis – each with the ALLRELATED keyword.

Most importantly, for any and all errors that are detected, get them fixed and cleaned up. You'll probably be surprised at how many errors you have the first time you run a full-blown DIAGNOSE with ALLRELATED specified, but once they're cleaned up, subsequent runs should find minimal errors.

Conclusion

ICF catalogs are often a forgotten item: they run ... and run ... and run, without problems, and without giving you any worries. Suddenly, one breaks, or one unexpectedly hits 123 extents and stops all further allocations. Or worse, your system grinds to a halt because of some hiccup within CAS – and you find out just how important catalogs are to you.

The truth is, catalogs should become a part of your z/OS system that you pay attention to regularly. They form an important system metadata structure that should be properly defined and managed. And, they should be a part of your system that you aren't afraid to touch.

To get there requires that you carve out some time to devote to their care and feeding. I hope that this White Paper will provide a bit of insight that will get you thinking of ICF

catalog areas to tune and improve.

With **CR+**, you have a tool that you can safely use to:

- Learn about catalogs – for example, use the ZAP PRINT command to format BCS and VVDS records right on your ISPF screen.
- Become sufficiently comfortable with the CR+ MERGECAT command to begin the project to spread the cataloged data set load out across your catalogs.
- Use the BACKUP and RECOVER commands in SIMULATE mode to ensure the proper procedures and knowledge are available when a catalog crisis occurs.

If you aren't already a licensed user of **CR+**, it's time to give it a try. A free trial of **CR+** can be requested at 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 OS/390 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. Ron travels widely, meeting with customers and presenting at national and international conferences.

Mainstar is a registered trademark and Catalog RecoveryPlus is a trademark of Mainstar Software Corporation. OS/390 and z/OS are registered trademarks and DFSMSHsm is a trademark of IBM. ViewDirect is a registered trademark of Mobius Management Systems, Inc.