


Storage Strategies

BY STEVE PRYOR

Where's the Data? Obtaining Information from the Catalog



Know your subject...every aspiring writer or speaker has heard this piece of advice. An understanding of the subject matter is a requirement for success in any discipline, including storage management. The storage administrator must understand not only what data exists, but must also have an understanding of the metadata, that is, the data about the data—where files are stored, what their characteristics are, who owns them, when they will expire, and so on.

In enterprise systems, the basic repository for information about most datasets is the ICF (Integrated Catalog Facility) Catalog. Prior to the year 2000, it was possible to store this information in old-style non-ICF catalogs and CVOLs, but these have thankfully now disappeared. Today's ICF catalogs are much more reliable and can contain a vast amount of information about each dataset, including the names of its various components, associations and aliases, as well as such details as stripe-count, SMS classes and sharing status.

Catalogs also contain the answer to one of the first questions that the storage administrator asks when presented with a problem—where is the dataset? Probably the most important function of the catalog is simply to allow the system to locate a file without the need for the programmer to specify the volume on which it resides. Indeed, in a large modern data center, this would be impossible given the sheer number of datasets and volumes present. Under DFSMS, datasets may also reside on different volumes or different media (tape vs. DASD, for example) at various points in their life cycle. The location and characteristics of a dataset as recorded in the catalog are important to both the normal operation of the system and the storage administrator who is trying to write a report or solve a problem.

While the format and content of catalogs has been modified over the years, the basic tool for getting information out of catalogs has changed remarkably little. For both application programmers and storage administrators, one of the most common means of obtaining dataset information is a simple IDCAMS LISTCAT job. LISTCAT has the advantages of being widely understood and relatively simple to use, and it provides complete catalog information in an output format that has remained largely stable, with only minor changes from release to release to accommodate new fields in the catalog records. Many installations have taken advantage of this fixed-format output to write LISTCAT post-processors, usually in REXX or CLIST, to reformat the data to meet installation needs.

The LISTCAT control statement is a very simple one for the amount of data it can return. There are no required parameters for LISTCAT. The simplest form, specifying the command with no other operands, will list every entry in the master catalog. Usually, a little filtering is desirable, and adding the CATALOG(catname) parm will list all of the entries in a particular catalog. (Under TSO, just those entries with the user's high-level qualifier are shown). Another common requirement is to restrict the information returned to that which describes particular types of entries. LISTCAT ALIAS with no other operands, for example, will list all of the alias entries in the master catalog. Similarly, LISTCAT NONVSAM will show just the entries for non-VSAM datasets, while LISTCAT CLUSTER will display the VSAM entries—just the thing for finding 'rogue' user datasets catalogued in the master catalog.

Often the desired dataset name is known, and it's the information about a particular

dataset that's of interest. In this case, the ENTRY parameter is used to provide the dataset name, which might even be the name of the catalog itself. LISTCAT ENTRY(catname) CATALOG(catname), for example, will list the catalog's own self-describing entry. In this case the cluster name will be binary zeroes. This sort of LISTCAT might be useful in determining whether a catalog is full or nearly full, as indicated by the high-allocated and high-used RBA fields and the number of extents. (APAR OW54162 also provides a means of getting a warning when a catalog is running out of extents, via the new MODIFY CATALOG, NOTIFYEXTENT command).

Often one is interested in more than just a single dataset. A certain amount of generic selection can be performed with the LEVEL or ENTRIES parms. Specifying LEVEL implies that the catalog should be searched for datasets with the specified high-level qualifier(s), i.e., LEVEL(SYS1.TEST) will display all entries with those first two qualifiers. A limited amount of dataset name masking can also be specified in listing catalog entries—an asterisk can be used as a placeholder for a single qualifier in the ENTRIES parameter, and for multiple qualifiers in the LEVEL parameter. The last qualifier, however, cannot be an asterisk (this results in a syntax error, and it is implicitly present, anyway with LEVEL). Thus LIST NONVSAM ENTRIES(PROD.*.JUNE) will list all non-VSAM datasets having 3 qualifiers in the name, where the first qualifier is 'PROD' and the third, 'JUNE'. If LEVEL rather than ENTRIES were specified, the same selection criteria are used, except only the first 3 qualifiers of the dataset are examined, and datasets with names such as 'PROD.XYX.JUNE.SYS.FOUR' would also be listed.

In some cases, the reason for running LISTCAT is just to get a list of dataset names; in others, some information beyond just name is the object of the catalog search. LISTCAT output is presented in groups, as obtained from the various 'sub' records or cells in the catalog components (the BCS, or Basic Catalog Structure of which only one exists per catalog, and the VVDS, or VSAM Volume Dataset, which resides on each DASD volume). For example, specifying LISTCAT ENTRY(somedsn) VOLUME returns not only the volume serial number the dataset resides on, but additional information from other cells, such as HISTORY information (ownerid, creation/expiration dates, and DFSMSdfp release number). ALL is probably the most commonly specified parameter, resulting in a formatted output of all the BCS/VVDS information for the selected dataset and its associated aliases, generations, paths, and so on.

The volume serial information returned from LISTCAT is usually the volume(s) that the dataset resides on, but for a dataset that has been migrated by DFSMSHsm or FDRABR, the volume serial number will be MIGRAT. This differs slightly from the same information returned by ISPF option 3.4, which will indicate a volume serial number of MIGRAT1 or MIGRAT2 for a migrated dataset. This is because MIGRAT is the actual volume serial number recorded in the catalog, and ISPF appends a '1' or a '2' depending upon whether the DEVTYPE field, also in the catalog entry, indicates the dataset is on disk or on tape. Also, for a migrated VSAM dataset, LISTCAT will indicate that the dataset is non-VSAM. The dataset will be recatalogued as a VSAM dataset if it is recalled.

The volume serial number in the LISTCAT output for a dataset might also be all asterisks ('*****'), indicating that the dataset was catalogued using a symbolic reference to the system residence volume, or it may contain a system symbol such as '&SYSR2'. In these cases, the DEVTYPE field will be zeroes, since these values are not resolved until the system actually needs to refer to the dataset.

LISTCAT is subject to a number of limitations, of course—the only allowable search criteria are entry name and type, and the generic search is limited (no partial qualifiers are allowed, for example). Also, while the output format changes only infrequently,

FIGURE 1: SOME SAMPLE LISTCAT STATEMENTS

```
List all entries in the Master Catalog
LISTCAT

List Alias entries in the Master Catalog
LISTCAT ALIAS


List all entries in a User Catalog
LISTCAT CAT(USER.CATALOG)

List Generation Datasets in a User Catalog
LISTCAT GDG CAT(USER.CATALOG)

List a Catalog Self-Describing Entry
LISTCAT ENTRY(SOME.CATALOG) CAT(SOME.CATALOG) ALL
```

each DFSMSdfp release usually requires annoying modifications to homegrown LISTCAT output processors.

One way of dealing with these limitations, which has the advantage of costing no more than LISTCAT itself (i.e., nothing, as it's included with the system), is the Catalog Search Interface. CSI is a documented, supported interface into the Generic Filter Locate function of SVC 26. This interface, which is documented in DFSMS:Managing Catalogs (SC26-7409), provides much better selection criteria than LISTCAT (partial qualifiers and wildcards can be used), and can return only the information requested, eliminating extraneous unwanted data. CSI is normally invoked from an assembler language program, but it can be called from REXX, as shown in some of the samples available in the SYS1.SAMPLIB library. (For a more detailed discussion of CSI, see the May 1998 Storage Strategies column at www.naspa.com).

Knowledge of the subject matter is important when embarking on any enterprise. When the subject matter is an installation's data, an understanding of the details of both basic tools such as LISTCAT and more sophisticated mechanisms such as the Catalog Search Interface can make understanding and solving problems much easier. 

NaSPA member Steve Pryor is a senior software developer with DTS Software, Inc., a vendor of enterprise storage management products. Steve has been involved in software development, storage management, and disaster recovery for more than 20 years. He can be contacted at pryor@mailatlanta.net.