SPERRY UNIVAC
COMPUTER SYSTEMS

Technical Bulletin

Name OS/3

Bulletin #10

Order No. UP-8605.10

OS/3 TECHNICAL BULLETIN

This bulletin contains information on the use of:

FILE CATALOGING

Request additional copies by submitting Sales Help Requisition form (UD1-578) through your local Sperry Univac representative to:

CUSTOMER INFORMATION DISTRIBUTION CENTER (CIDC)
Sperry Univac
555 Henderson Road
King of Prussia, PA 19406

Lists:	Bulletin No.:	Date:
18, 19, 20, 21, 75, 76, CZ	10	July, 1979

OS/3 TECHNICAL BULLETIN SUMMARY

The following Technical Bulletins are published for the OS/3 system. Current items are identified with an "*" in column one; scheduled items are identified with an "**" in the date column:

SYSTEM	REL.#	DATE	ORDER#	ITEM and DESCRIPTION
*OS/3	4.3	1/78	UP-8605.1	OS/3 Technical Bulletin #1 (This document presents an overview of the UTS 400 support and gives some user guidelines.)
*OS/3	4.3,5.0	3/78	UP-8605.1-A	OS/3 Technical Bulletin #1-A (This update contains page replacements to UP-8605.1.)
*OS/3	ALL	4/78	UP-8605.2	OS/3 Technical Bulletin #2 (This document provides a list of the options that can affect the performance of an OS/3 IMS 90 system.)
*OS/3	4.3	1/78	UP-8605.3	OS/3 Technical Bulletin #3 (This document is a User Guide for the UTS 400 CHARACTER PROTECTION MODE available with release 4.3.)
*OS/3	5.0	7/78	UP-8605.3-R1	OS/3 Technical Bulletin #3-R1 (This document contains updated guidelines for the UTS 400 CHARACTER PROTECTION MODE available with release 5.0.)
*OS/3	5.0,5.2 5.2.1,6.0	11/78	UP-8605.4	OS/3 Technical Bulletin #4 (This document contains information on the use of the 8413 DISKETTE FILE CREATION UTILITY.)
*OS/3	5.2	5/79	UP-8605.5	OS/3 Technical Bulletin #5 (This document contains information on the use of DATA UTILITIES for OS/3 Release 5.2.)
*OS/3	ALL	12/78	UP-8605.6	OS/3 Technical Bulletin #6 (This document contains information on the use of IMS 90 Multi-Thread.)

OS/3 TECHNICAL BULLETIN SUMMARY

SYSTEM	REL.#	DATE	ORDER#	ITEM and DESCRIPTION
*OS/3	ALL	3/78	UP-8605.7	OS/3 Technical Bulletin #7 (This document contains information concerning techniques for processing unordered IRAM files.)
*OS/3	5.2/5.2.1 6.0	5/79	UP-8605.8	OS/3 Technical Bulletin #8 (This document contains information on the use of CHARACTER PROTECTION MODE UTILITY for the UTS 400; this utility is available with Releases 5.2/5.2.1 and 6.0.)
*OS/3	5.2/5.2.1 6.0	5/79	UP-8605.9	OS/3 Technical Bulletin #9 (This document contains information on the use of the IBM 3741 MEDIA COMPATIBILITY UTILITY for the UTS 400; this utility is available with Releases 5.2/5.2.1 and 6.0.)
*OS/3	ALL	7/79	UP-8605.10	OS/3 Technical Bulletin #10 (This document contains information concerning OS/3 FILE CATALOGING.)

Note: Technical Bulletins are issued as they become available, and may or may not be issued in sequential order.

OS/3

FILE

CATALOGING

TABLE OF CONTENTS

ABSTRACT

This report is written to explain OS/3 File Cataloging. The purpose of this report is to provide the advanced OS/3 Job Control User with specific information about the file catalog and generation files. OS/3 Generation Files are easy to use once the cataloging mechanism and formulas are understood. This report will attempt to provide the necessary information to the Programmer to make the best use of OS/3 Generation Files.

INTRODUCTION

The purpose of this report is to explain OS/3 Generation File cataloging and usage to Systems Analysts and Programmers. This report is intended for advanced OS/3 Job Control Users who are currently using software release levels R4.Ø through R5.2. This report also covers correction 304 and enhancements for Release 6.Ø. The report covers the following topics, as related to generation files: Catalog, Generation Files, Job Control Statements, Considerations for Cataloging, Generation Cataloging Formulas, Generation File Usage Examples, Correction 304 to Rel. 5.Ø (all levels) and Rel. 6.Ø Enhancements.

II. FILE CATALOG

What is the OS/3 File Catalog? The OS/3 File Catalog is a system resident file that contains information about files. YCAT is the name of the File Catalog. The information contained in YCAT is used by the Run Processor to process OS/3 job control statements. The File Catalog offers data file protection by restricting unauthorized users from obtaining data from files. It also enables authorized users to retrieve or create data files with a minimal number of job control statements.

A catalog entry is identified by a // LBL job control statement and cataloged by a // CAT job control statement. Once cataloged, files are accessible by a simple reference in a subsequent job stream. For example, a portion of a stream to catalog a file may look like this:

Example 1:

```
// DVC 60 // VOL DISK01
// LBL PAYABLES
// LFD CATFILE
// CAT CATFILE
```

In this stream, file PAYABLES is specified as being on Volume DISK01 of an 8416 disk subsystem (logical unit number of 60).

To catalog this file, it is necessary to specify the same logical file name on the CAT statement. By matching the LFD parameter, the CAT statement links the device, volume, and file identifier to the logical file. The logical file name also corresponds to the DTF label in Data Management, the FD entry in COBOL, the READ or WRITE statement in FORTRAN, or the file description specification in RPG.

After cataloging a DVC-LFD sequence, the file definition may be used in other JCL streams by a reference to the file identifier only. For example:

```
// JOB USEIT
// LBL PAYABLES
// EXEC XXX
```

will allow program XXX to use the file PAYABLES as cataloged in example 1.

III. GENERATION FILES

A. Generation File Definition

What are generation files? Basically, a Generation File is a group of files with the same file identifier and a unique two digit generation number affixed to the end of the file identifier. The two digit generation number is affixed sequentially starting with zero and incremented by one until ninety-nine is reached, thus allowing one hundred unique labels. For example, ØØ is the original file, Ø1 is the next generation, and so on. By allowing the Run Processor to control the generation number which is attached to the file identifier, a common file identifier can be used for a group of files having a common use.

B. Generation File Usage

1. Data Protection

Data Protection is one application where generation files can be used. By having the Run Processor maintain a group of files for back-up purposes, operations personnel no longer need to perform this function. As each file is created, it receives a new generation number. This procedure allows for the old data to be retained as back-up. The back-up data is actually any previous generation. Generation file usage will be explained in Section VII.

2. Ease of Use

One of the benefits of using generation files and the file catalog is that once the file is cataloged, only one job control statement (LBL statement) is required to identify the data file. Through the use of cataloged generation files, it is possible to eliminate a majority of the job control statements that are normally required for program execution.

IV. JOB CONTROL STATEMENTS

A. Job Control Statement Requirements

This section presents specific items that should be considered when using the following job control statements for generation files:

DVC job control statement

1. The logical unit number on the // DVC job control statement should not be of the general type for disks, but must be one for a specific device type. This information is obtained from the system generation listings.

2. The IGNORE option is not cataloged.

VOL Job control statement

The PREP option is not cataloged.

DD Job control statement

1. The // DD statement is not cataloged.

EXT job control statement

1. The // EXT statement is not cataloged.

LBL job control statement

1. All parameters of the // LBL statement are cataloged.
2. The total number of characters for the file identifier field is seventeen for tape files and forty-four for disk files. In the use of generation files, only fifteen characters for tape files and forty-two characters for disk are permitted. Two characters are used for the generation numbers.
3. See V - Considerations for cataloging sections 5,6,7, and 8 for more details on use of LBL.

LFD job control statement

1. INIT, RELOAD, EXTEND, and PREP parameters are not cataloged.

B. Catalog Control Statements

1. Entering Files in a catalog

To catalog a file, a CAT job control statement is used. When the CAT job control statement is encountered, the file defined in a previous device assignment set is marked for cataloging at Run Processor termination. The file is cataloged with any qualifiers or passwords contained on the LBL statement.

The format of the CAT job control statement is:

```
//symbol CAT lfdname   ,catpw    ,SCR    ,GEN=nn
```

The CAT statement identifies the logical file name as used in the LFD statement. Thus, the lfdname parameter of the CAT statement must agree with the filename parameter of a previous LFD statement.

In some cases, the File Catalog itself may have its own password. Whenever the catalog is password protected, you must specify the correct password parameter consisting of one to six alphanumeric characters or access to the file catalog is denied. This procedure prevents unauthorized users from access to the File Catalog. The catalog password must be specified on all CAT and DECAT statements, if the catalog is password protected.

The SCR parameter indicates that, when a subsequent DECAT job control statement for that file is encountered, the file should be scratched, not just removed from the catalog. This option is effective only for files specifically referenced by their full compound name in the LBL statement; if the last character of the LBL statement is a period or slash, the SCR parameter is not effective.

The SCR parameter has no effect in the CAT statement in which it is found but does have an effect later when the file is decataloged.

The GEN parameter indicates the maximum number of generations maintained on the catalog; nn may range from 01 to 99.

In Example 2, the file defined in Example 1 is cataloged as a generation file.

Example 2:

```
// DVC 6Ø // VOL DISKØ1
// LBL PAYABLES
// LFD CATFIL
// CAT CATFIL,,SCR,GEN=1Ø
```

The CAT statement in Example 2 indicates that the file defined by the logical file name of the LFD statement is to be put in the catalog. When the file is subsequently removed from the catalog, the file will also be scratched. A maximum of 1Ø generations is to be maintained in the catalog.

A file that has been cataloged can now be referenced by another job.

A cataloged file may not be referenced in the same stream in which it was entered into the catalog. The file is cataloged at the end of the job stream, when /& is encountered by the Run Processor.

2. Removing Files From a Catalog

To remove a file from the catalog, a DECAT job control statement must be used. The file to be removed is defined by the LBL statement and designated by the logical file name on the DECAT statement. The LFDname parameter of the DECAT statement must agree with the file name parameter of the LFD statement in a previous device assignment set.

The format of the DECAT job control statement is:

```
//symbol DECAT lfdname [,catpw] [,SCR] [,GEN]
```

The catpw parameter is a 1 to 6 character alphanumeric password to allow changes to the File Catalog.

Once the catalog is password protected, subsequent CAT or DECAT statements cannot be used without using this password. The SCR parameter, if previously specified on the CAT statement, will scratch the file as well as remove it from the catalog. The GEN parameter removes all generations specified for the file from the catalog.

Example 3 removes a file identifier from the catalog.

Example 3:

```
// LBL PAYABLES
// LFD CATFILE
// DECAT CATFILE,,SCR
```

V. CONSIDERATIONS FOR CATALOGING

This section addresses those areas that may cause misunderstandings and problems for the user. By highlighting, and in some cases, repeating the requirements for Generation File usage, it may prevent the occurrence of major problems during implementation. Presenting these items before the examples are shown, (Section VII), should make the examples easier to follow.

1. The catalog entries representing files and the files themselves are not one and the same. Remember, that operations on the File Catalog do not effect the actual data file. Specific functions can be performed, such as the SCR function, which clears both the File Catalog and the VTOC of the file identifier. It must be emphasized that the catalog entries representing files and the files are separate entities.

2. // CAT and // DECAT work together in scratching files. If the SCR parameter is specified on the CAT statement, a single file (or one generation of a file) is scratched and removed from the catalog automatically whenever a subsequent DECAT job control statement is encountered for the named file. If the SCR parameter is only entered on the DECAT statement, the file is removed from the catalog and scratched. If the GEN parameter is specified on the DECAT statement, all generations of a file are removed from the catalog, but no files are scratched.

3. Each cataloged file must have a unique file identifier (LBL). An uncataloged file can only be cataloged, by the CAT job control statement. When cataloged, the entire device assignment set for the file must be given, but the file space is not allocated (i.e., no EXT statement is required).

4. Any previously cataloged file may be referenced by a single LBL statement containing only the file identifier. No other device assignment statements are required since the information is now contained in the catalog. DVC and VOL statements are not required. Referring to Example 1 to reference the file PAYABLES, all that is needed is:

```
// LBL PAYABLES
```

5. When a // LBL job control statement is processed, the catalog is searched for that file identifier (whether the file is being accessed through the catalog or not) so password verification can be done.

6. To provide the necessary catalog information about a file, the LBL job control statement is expanded to include file information for a catalog. The basic format of the LBL statement is:

```
// LBL file-id
```

To append file information for a catalog, the file-id field is further defined to be:

```
                    { file qualifier          qual
                    { file identifier         level-id
file-id=            { generation number       { +n
                    {                         { -n
                    {                         { nn
                    { passwords               (rpw/wpw)
```

Thus, the format showing the file-id parameters for a catalog definition becomes:

```
// LBL [qual/] level-id1 [.level-id2] ... [.level-idn][{+n}][(rpw/wpw)]
                                                       {-n}
                                                       {nn}
```

7. Following the file-id parameter on the // LBL statement is the generation number parameter. The generation number can be specified in one of three ways: +n, -n, or nn. The symbol nn represents an absolute generation. The +n provides the increment value for the creation of a new generation of a file. The plus sign (+) must be coded. The -n is used as a relative reference number, (relative to current), when accessing an older or previous generation number. The minus sign (-) must be coded. If 99 generations are exceeded, the generation number goes back to zero (00). For generation files, a catalog update is performed when a positive (+) relative generation is given.

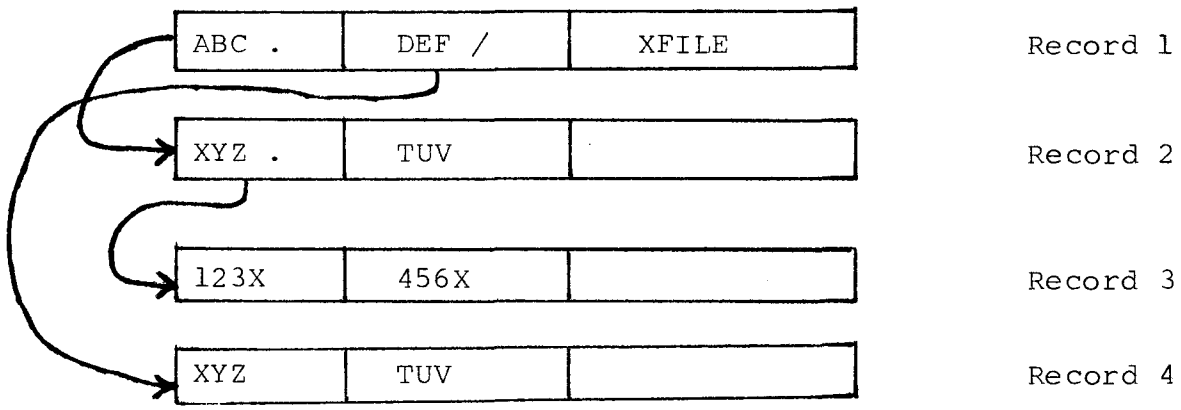
8. The labels are stored in the catalog using a hierarchical method to facilitate faster scans of the catalog. With a label name, the levels of the hierarchy are separted by a '/' (for qualifiers) or by a '.'.

The catalog is scanned by first looking for a match on the first level. When this is found, the first level entry points to all entries at the next level; etc.

For example, assume the following file labels were cataloged:

```
ABC . XYZ . 123X
ABC . XYZ . 456X
ABC . TUV
DEF / XYZ
DEF / TUV
XFILE
```

The internal structure of the catalog would be:



If the program requests label DEF/TUV the first level of the heirarchy is searched for 'DEF/', when this is found it points to the second level which is at record 4 in this example.

The next level is searched for 'TUV', when found, it contains the cataloged file information.

Most catalog users have a large number of entries in the first level. The first level typically spans a number of records and is time consuming to scan. The // QUAL qualname job control statement specifies the first level of the heirarchy which is automatically prefixed to all file labels in the job stream. The run processor also notes the position in the catalog of the qualname, thus speeding up the first level search of the catalog.

9. When a generation file on disk is being created, an EXT statement must be included in the file assignment set to allocate disk space for the new file. If a new generation is to reside on a volume different from the current generation, DVC and LFD statements must also be included. A new generation member is only created via a positive relative reference.

10. If a DVC or VOL job control statement is included in a device assignment set which references a cataloged file, then the entire device assignment set, (the DVC through LFD sequence), must be specified. If a permanent change is to be made to the device assignment set for a cataloged file, the file must be recataloged.

11. A generation file has to be explicitly cataloged only once (by the CAT job control statement). For example, the following two jobs, when run in the order shown, result in the cataloging of the multigeneration file that has a file identifier of GEN:

Example 4:

```
// JOB CAT
// DVC 60 // VOL DSP001
// LBL GEN  // LFD A
// CAT A,,,GEN=3
/&
// FIN
// JOB CAT1
// LBL GEN+1
/&
// FIN
```

Notice that a CAT job control statement is not needed in the second job.

12 . The actual catalog update for CAT and DFCAT job control statements or automatic generation operations (+n) is performed when the /& job control statement for that job is processed by the Run Processor If the control stream contains errors severe enough to inhibit the queuing of the job, the catalog operation is not performed. The reasons for updating the File Catalog at this point are to avoid locking out other jobs which reference the catalog, and to ensure that generation references within a job are consistent. In general, this means that while a file may be referenced (by its file name) in the job which catalogs it, the information to be cataloged is not available to job control in the same job. For instance, assume that the file identified as GEN is cataloged. The following control stream would result in an invalid reference:

Example 5:

```
          // JOB GEN2
          // DVC 60   // VOL DSP028
1         // LBL GEN+1 // LFD B
2         // LBL GEN+1 // LFD C
          /&
          // FIN
```

The file name of B in the LFD job control statement (1) would explicitly generate a file control block for volume DSPØ28. The file name of C in the LFD job control statement (2) would obtain the most recently cataloged information in volume DSPØØ1. If two LBL job control statements with new DVC and VOL information are in the same job (as in Example 5), the second LBL reference obtains the DVC and VOL information from the previous generation. If these two references had been in separate jobs, no conflict would have occurred; or, the DVC and VOL job control statements for volume DSPØ28 could have been repeated in the second reference (2).

In a job where more than one update to the catalog occurs, three courses of action are possible for catalog manipulation. You can:

1. Run the catalog updates as separate jobs;

2. Use only one device assignment set for the file to be cataloged, one LBL statement, and make all references to the file name on one LFD job control statement; or

3. Duplicate the device assignment set (with different file names) for each reference.

VI. GENERATION CATALOGING FORMULAS

The following formulas are used by the run symbiont in determining which generation file information should be used for creating the next catalog entry, when no DVC and VOL information is specified in the job stream. The formulas are only used when the generation number on the // LBL statement is plus (+n).

The formulas are:

1. If T+1>N take the first (oldest) entry.

2. If T+1≤N take the last (newest) entry.

T is equal to the number of generation members in the catalog for a specific file-id.

N is equal to the number of generations specified on the GEN parameter of the // CAT statement.

Formula number one is read as follows:

If T (number of generation members in the catalog) plus one is greater than N (the number specified by the GEN parameter), then take the information from the oldest active entry in the catalog for that file-id and place it in the entry to be created.

Formula number two is read as follows:

If T (number of generation members in the catalog) plus one is less than or equal to N (the number specified by the GEN parameter), then take the information from the latest active entry in the catalog for that file-id and place it in the entry to be created.

VII. EXAMPLES OF GENERATION FILE USAGE

The purpose of this section is to provide the user with actual job control streams and visual representatives of the way the generation catalog mechanism actually works. The section is divided into three parts: Creating Generation Files, Maintaining Generation Files, and Generation File Problems and Solutions.

A. Creating Generation Files

This part presents examples that show how to enter a generation file into the File Catalog. Included in the examples are step-by-step procedures showing the effect that specific OS/3 job control statements have on the catalog, and the data files represented by those catalog entries. All of the illustrations attempt to parallel generation catalog structure and file usage between single volume users and separate volume users Comparing the two different catalog structures side by side shows the benefits of each type.

Example 6 Step 1 shows the job control required to catalog a file called INV. The maximum number of generations maintained are three (GEN=3). Notice that the job control and catalog entries for single volume and separate volume users are the same. The example also shows that an entry can be made in the catalog without creating a file. However, working with the catalog and the file separately should be avoided.

Step 1 consists of putting an entry of INV as a base file identifier into the catalog.

Example 6:

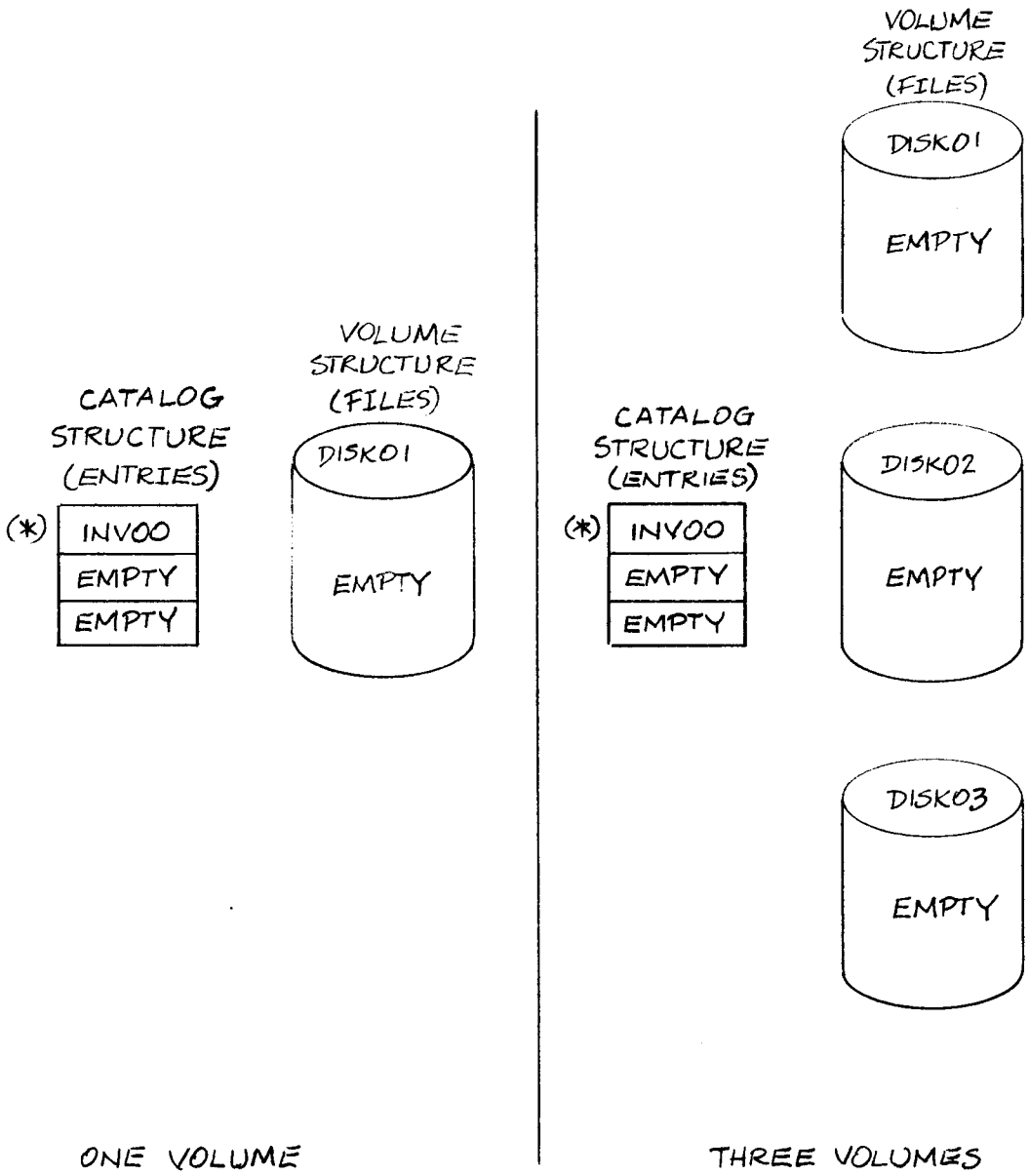
single volume

```
// JOB CATALOG
// DVC 60
// VOL DISK01
// LBL INV
// LFD NEWINV
// CAT NEWINV,,,GEN=3
/&
```

separate volume

```
// JOB CATALOG
// DVC 60
// VOL DISK01
// LBL INV
// LFD NEWINV
// CAT NEWINV,,,GEN=3
/&
```

The job control statement // CAT produces a three generation catalog file with the structures shown in Figure 1:



*means current generation

FIGURE 1. MULTIGENERATION CATALOG FILE BUILDING STEP 1

A catalog file entry has been generated that links the file identifier to the disk volume, even though no extent has been allocated and no file created.

The second step (Example 7) creates the file. Tape and cards are used to build a file. The file identifier as defined and cataloged in the first step is generation updated in the second step. It is only necessary to specify the // LBL, // EXT and the // LFD to create the new output file.

Example 7:

single volume

```
// JOB CREATE
// DVC 90
// VOL INSAV
// LFD INPUT
// LBL INV
// EXT ,,,CYL,1
// LFD OUTPUT
// EXEC BUILD
/$
  new records
/*
/&
```

Separate volume

```
// JOB CREATE
// DVC 90
// VOL INSAV
// LFD INPUT
// LBL INV
// EXT ,,,CYL,1
// LFD OUTPUT
// EXEC BUILD
/$
  new records
/*
/&
```

Figure 2 shows the state of the catalog and the disk files after an update has been performed.

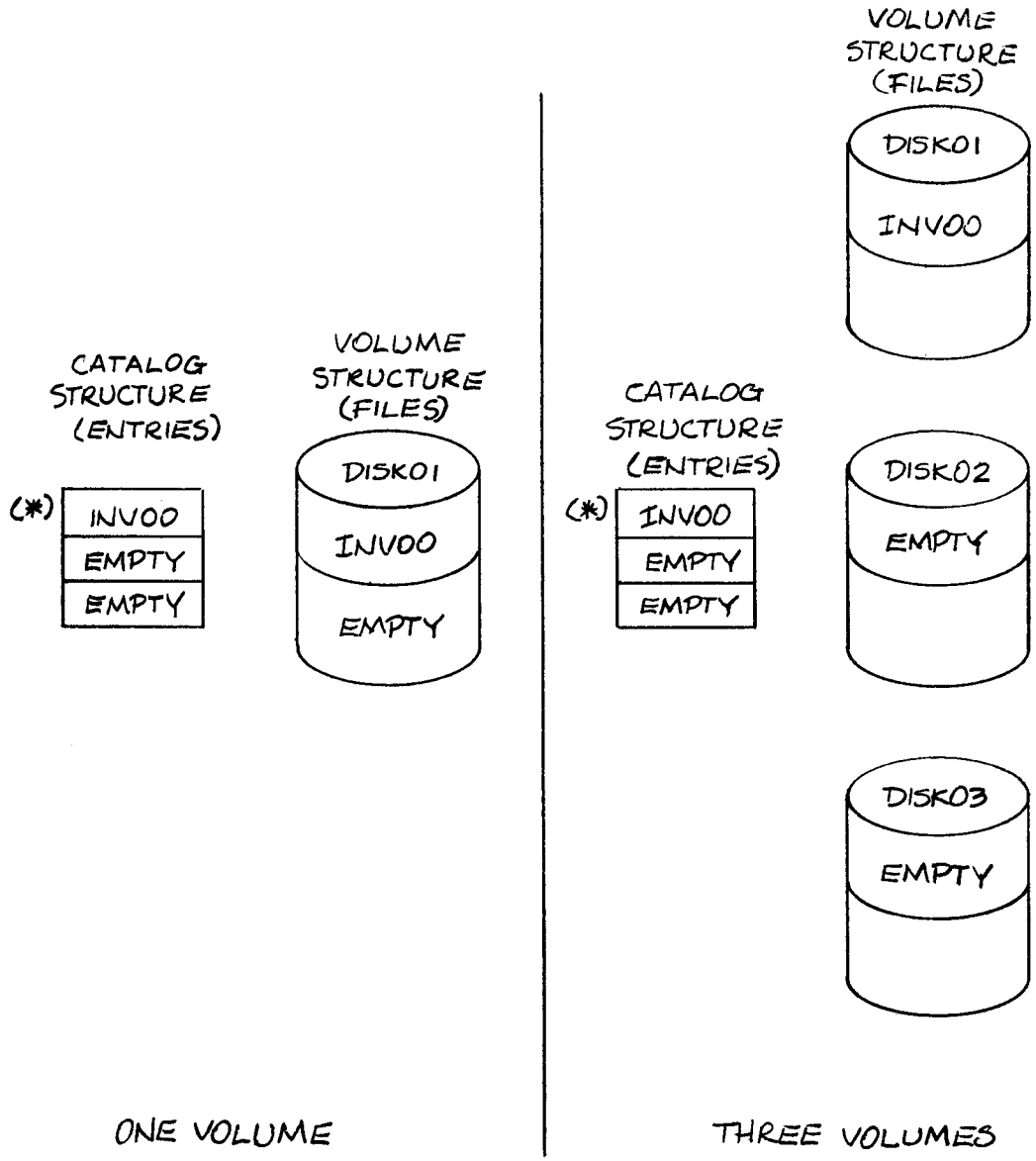


FIGURE 2. MULTIGENERATION CATALOG FILE BUILDING STEP 2

The third step (Example 8) is to update file INV to a second generation and allocate the file on volume DISKØ1 for single volume usage or DISKØ2 for use with three volumes.

Example 8:

single volume

```
// JOB UPDATE
// LBL INV
// LFD INPUT
// LBL INV+1
// EXT ,,,CYL,1
// LFD OUTPUT
// EXEC MERGE
/$
  new records
/*
/&
```

Separate volume

```
// JOB UPDATE
// LBL INV
// LFD INPUT
// DVC 6Ø
// VOL DISKØ2
// LBL INV+1
// EXT ,,,CYL,1
// LFD OUTPUT
// EXEC MERGE
/$
  new records
/*
/&
```

Figure 3 illustrates the basic difference between one volume and three volume processing. Note, that the second generation (INVØ1) of this file is on a separate disk (DISKØ2) The use of several disks for processing helps protect data.

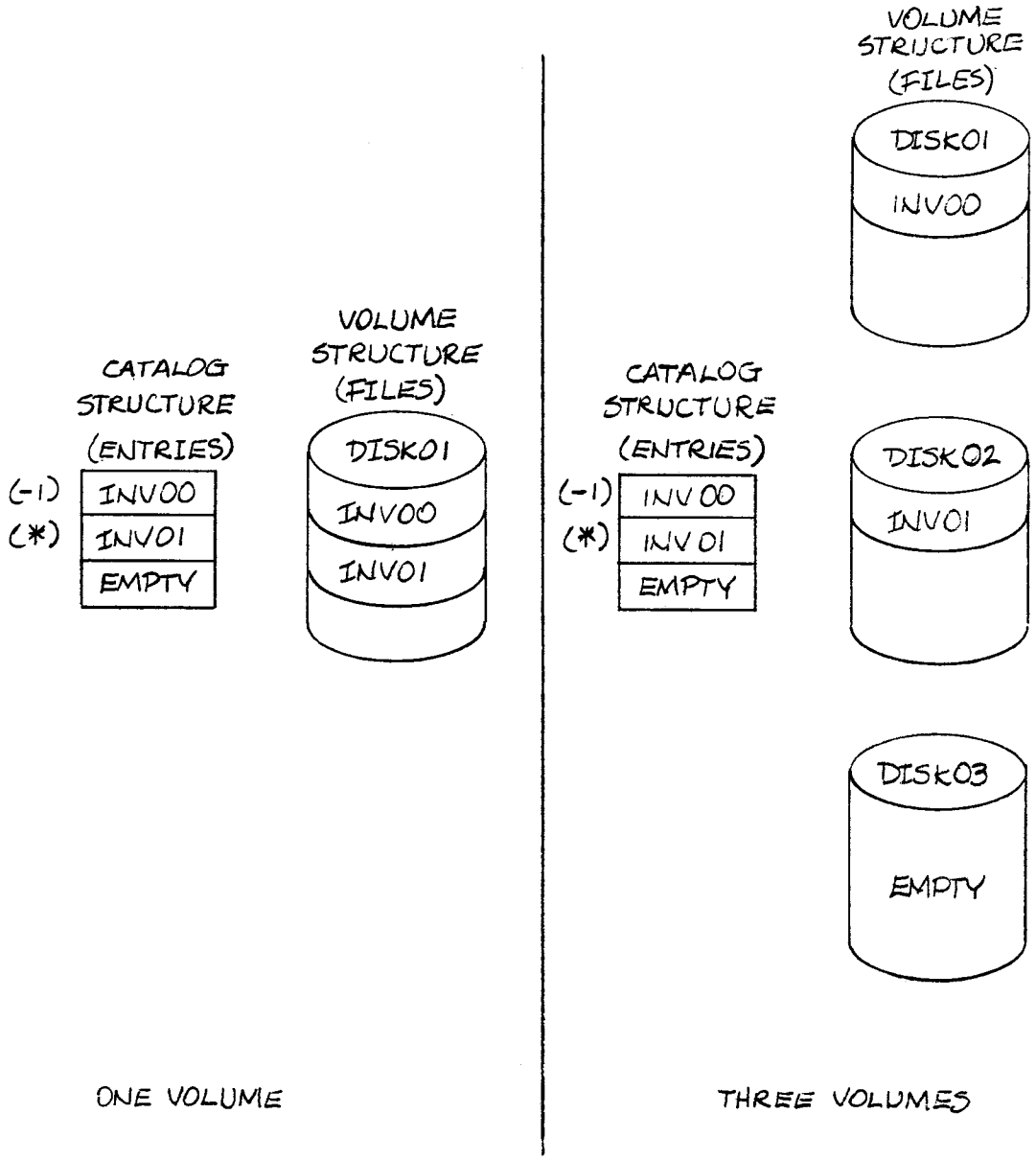


FIGURE 3. MULTIGENERATION CATALOG FILE BUILDING. STEP 3

The fourth step (Example 9) in this process completes the generation sequence for building the catalog. Figure 4 shows that both one and three volume users have three generations in the catalog.

Example 9:

single volume

```
// JOB UPDATE
// LBL INV
// LFD INPUT
// LBL INV+1
// EXT ,,,CYL,1
// LFD OUTPUT
// EXEC MERGE
/$
  new records
/*
/&
```

Separate volume

```
// JOB UPDATE
// LBL INV
// LFD INPUT
// DVC 60
// VOL DISK03
// LBL INV+1
// EXT ,,,CYL,1
// LFD OUTPUT
// EXEC MERGE
/$
  new records
/*
/&
```

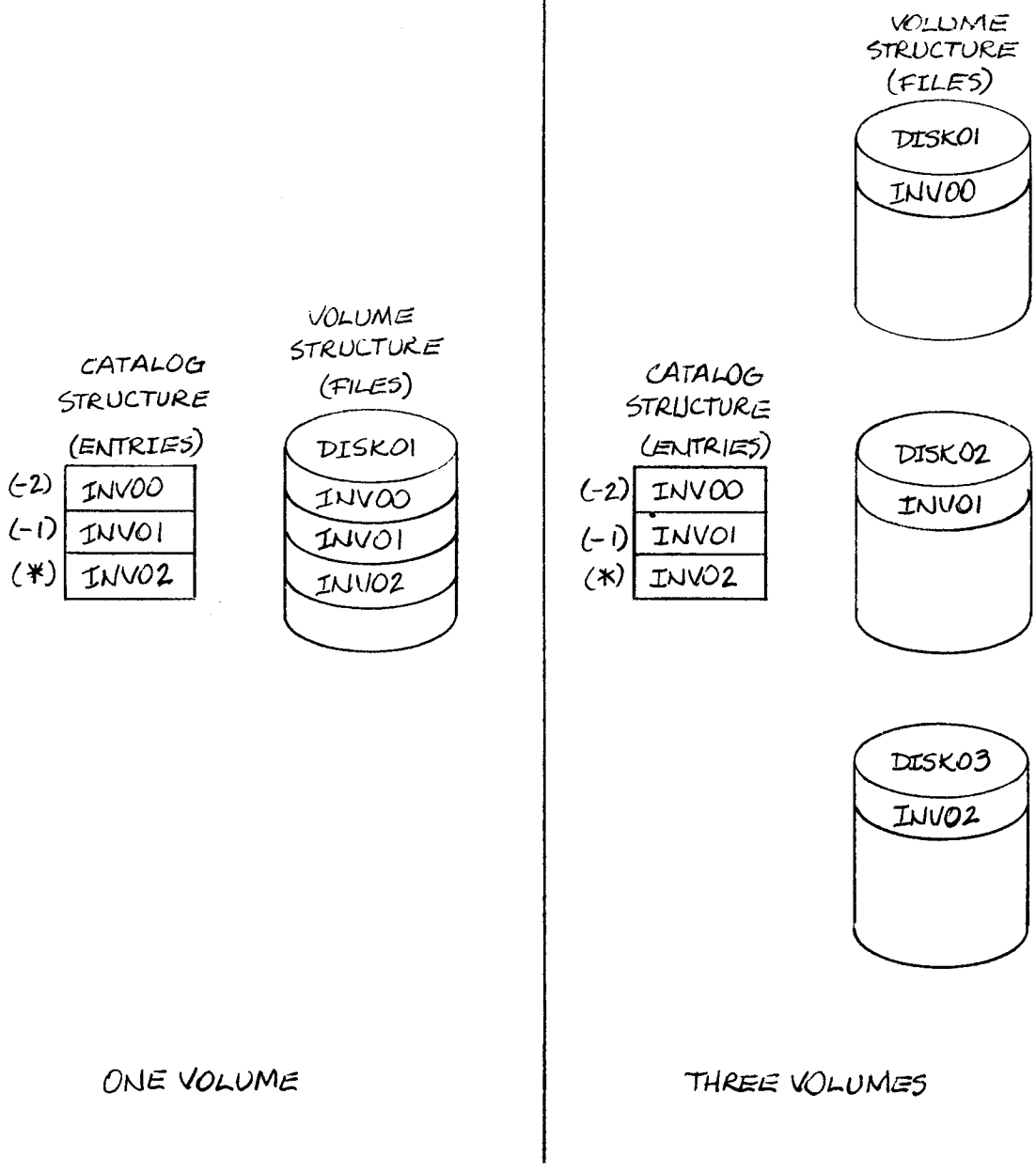


FIGURE 4. MULTIGENERATION CATALOG FILE BUILDING STEP 4

Figure 5 shows the catalog and file structure that would result if updating of the catalog and allocating of files would continue.

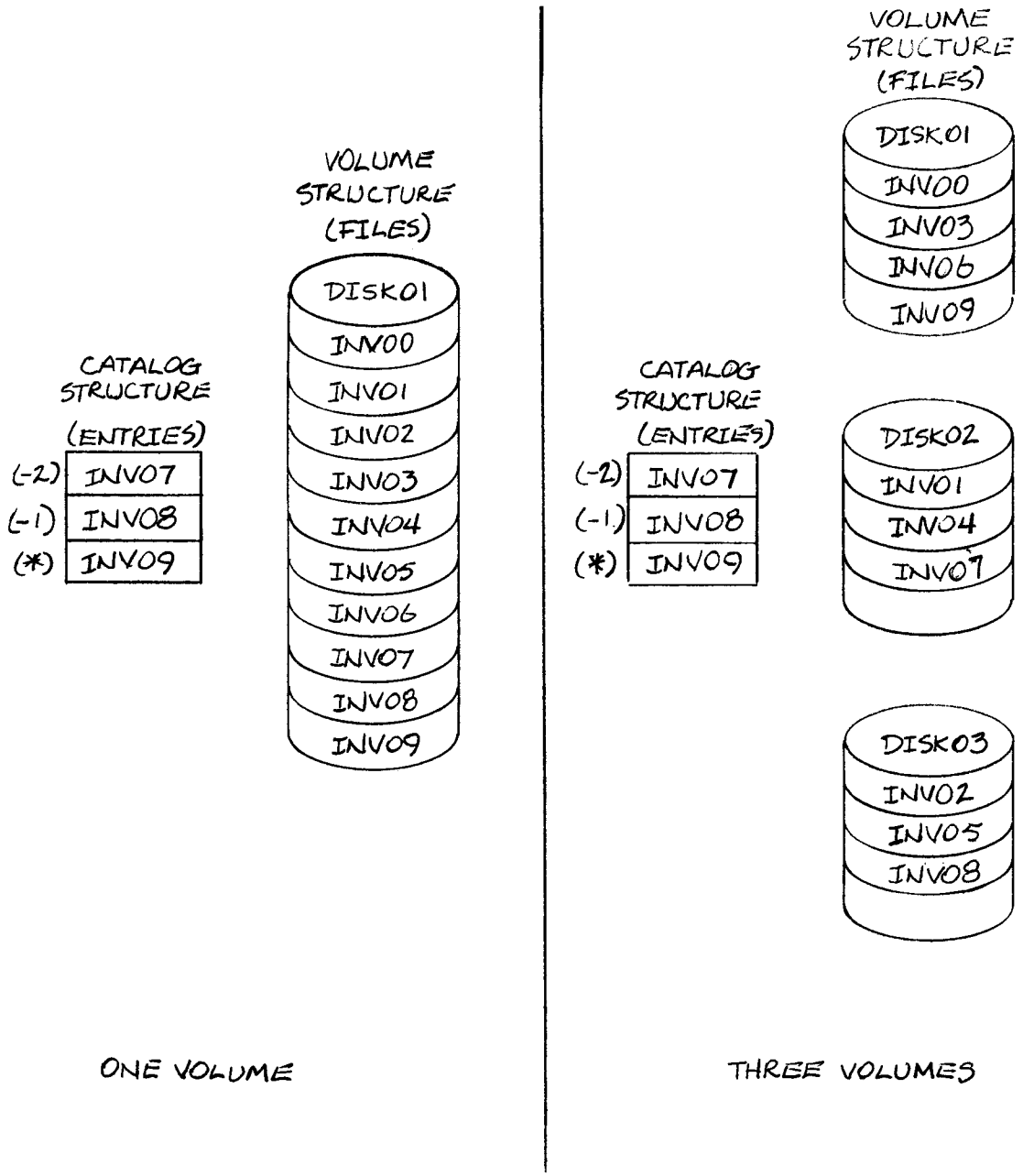
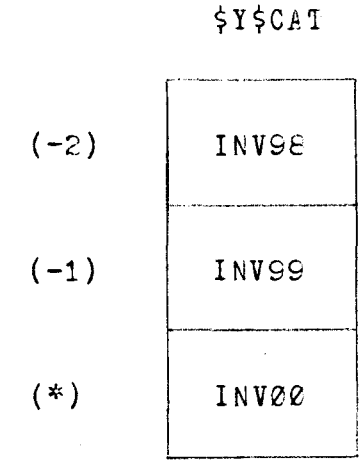


FIGURE 5. MULTIGENERATION CATALOG FILE BUILDING STEP 5

A point to keep in mind about generation numbers is that after generation number 99, the numbers begin again with ØØ. The catalog structure would look like:



B. Maintaining Catalog Files

This section shows how to maintain the catalog and the data files. Through the use of job control, it is possible to build job control streams that maintain the catalog and files at their best operating efficiency. Examples 1Ø through 14 in this section present only one method of catalog and file maintenance.

Example 1Ø uses the same basic first step that Example 6 used for cataloging the file. However, the next steps are different. The job control stream is constructed to scratch the oldest generation file after each update to the new file.

Example 10:

```
// JOB CATALOG
// DVC 60   // VOL DISK01
// LBL INV
// LFD CATFILE
// CAT CATFILE,,,GEN=3
/&
```

Example 11 shows how to set up a volume by allocating three files in one job step, without actually placing any data in the files. The structures are shown in Figure 6.

Example 11:

```
// JOB ALLOC
// LBL INV+1
// EXT ,,,CYL,1
// LFD FILE1
// LBL INV+2
// EXT ,,,CYL,1
// LFD FILE2
// LBL INV+3
// EXT ,,,CYL,1
// LFD FILE3
/&
```

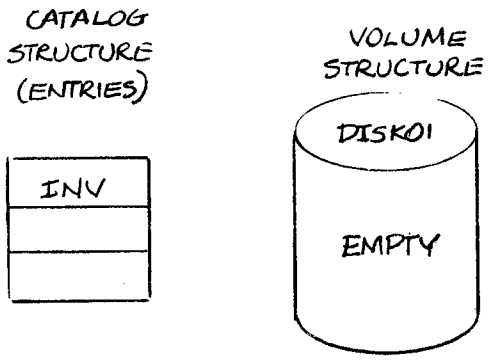


FIGURE 6. MULTIGENERATION UPDATE CATALOG FILE BUILDING. STEP 1
(Refer to Example 10)

The run processor **has been instructed to maintain three generations.** The file identifiers are automatically placed in the catalog as shown in Figure 7. The files are also allocated but have no data written to them.

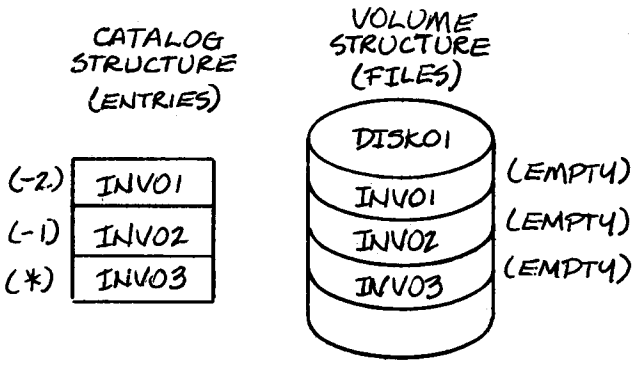


FIGURE 7. MULTIGENERATION UPDATE CATALOG FILE BUILDING. STEP 2
(Refer to Example 11)

The file can now be built or filled by constructing the control stream shown in Example 12:

Example 12:

```
// JOB CREATE
// DVC 90
// VOL INSAV
// LFD INPUT
// LBL INV
// LFD OUTPUT
// EXEC BUILD
/$
   new records
/*
```

This of course, assumes that a build program has been previously constructed to build the file. The job stream would then place it on the Disk01 volume. The volume would now look like Figure 8.

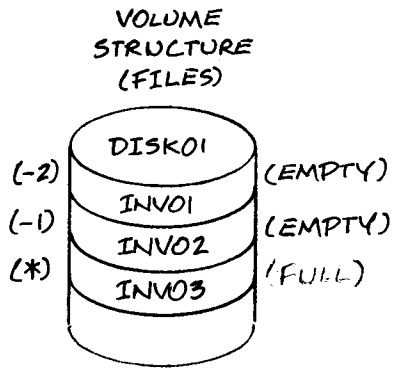


FIGURE 8. MULTIGENERATION UPDATE CATALOG FILE BUILDING. STEP 3

(Refer to Example 12)

INVØ3 would be the current file (*) and receive the data. To add an updated generation, the oldest file would be scratched at the same time and thus retain only the three active files listed in the catalog. This would have the advantage of keeping the number of files on the volume to a minimum.

In addition to scratching the oldest file, a SKIP job control statement could be placed in the control stream. If a fatal error occurred while updating, the newly created generation would automatically be decataloged. This would keep the catalog generation number current with the active files on the volume. Example 13 would accomplish this action for the user.

If the new file is built by merging the current file with the new records, the catalog and file structures look like this:

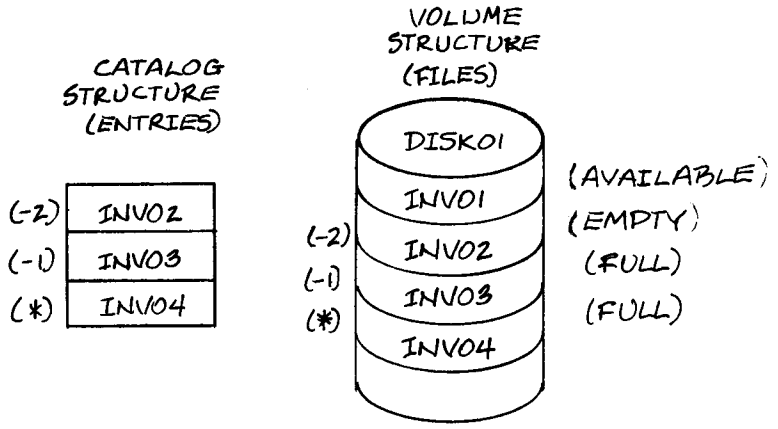


FIGURE 9. MULTIGENERATION UPDATE CATALOG FILE BUILDING. STEP 4

Take notice of the fourth file allocated to the volume in this step.

The SKIP job control statement (Example 13) checks the UPSI byte to determine if that file has been accepted. The file identifier with generation four has been added to the catalog and generation four now becomes the current generation. If the file is accepted, the branch to ERR is not taken and processing goes on to scratch the oldest file. This is accomplished by referencing INV-2 in the // LBL statement with an // LFD of OLDEST. Another SKIP statement will get us over the error RUN job stream call to a NOP job control statement, which is merely a convenient nonoperation used as a target branch. If an error had been encountered during the merge operation, the SKIP statement would have bypassed to the job control statement that tells us to run the job BACKUP.

Example 13:

```
// JOB UPDATE
// LBL INV
// LFD INPUT
// LBL INV+1
// EXT ,,,CYL,1
// LFD OUTPUT
// EXEC MERGE
/$
 new records
/*
// SKIP ERR,1
// LBL INV-2
// LFD OLDEST
// SCR OLDEST
// SKIP OUT
//ERR RUN BACKUP
//OUT NOP
/&
```

At this point we would want to decatalog the file identifier for generation four and also scratch the file built. This job stream would look like:

Example 14:

```
// JOB BACKUP
// LBL INV
// LFD NEWBAD
// DECAT NEWBAD,,SCR,ROL
/&
```

Refer to Section VIII. Changes for Current Release for a description of the ROL parameter.

This would return the catalog to its previous state where INVØ1, INVØ2, and INVØ3 would be the only files.

If the job stream in example 13 ran successfully, the following structures (Figure 1Ø) would be produced if a fourth entry for the catalog were made, but only three maintained.

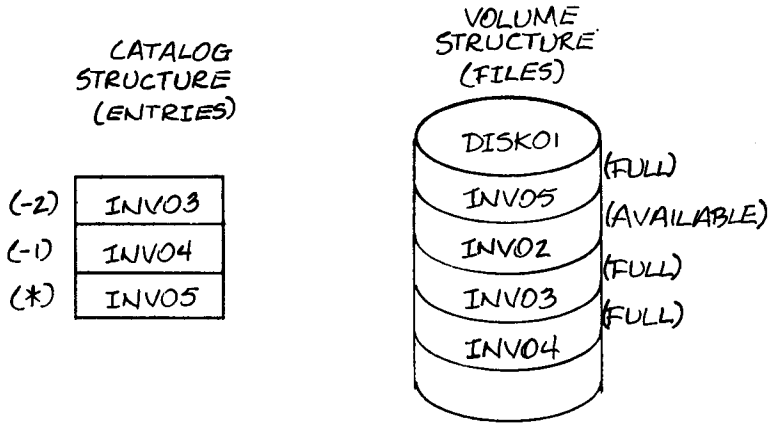


FIGURE 10. MULTIGENERATION UPDATE CATALOG FILE BUILDING. STEP 5
(Example 13 successful)

INVØ5 would become the current generation, but since there are only four files specified, INVØ5 goes into the only available catalog space (previously specified as INVØ1). INVØ2 is deleted from the catalog, since the catalog only maintains three, and now the INVØ2 catalog space becomes available. INVØ2 is also scratched from volume DISKØ1.

C. Generation File Problems and Solutions

This last section shows how to manipulate the catalog and files to circumvent normal problems of day to day operations. The examples show the outcome when no action is taken and when a recovery method is used. The solutions presented are not the only solutions available but can be used as a basis for other solutions. One volume generation files and three volume generation files are used to parallel the example in Section VII A. Generation File Usage. The following problems will be used:

1. in a one volume application, a file, INVØ2, is accidently scratched and

2. in a three volume application, DISKØ2, is accidentally destroyed.

The first step of this procedure is to DECAT the catalog entries (as shown in Example 15). This creates a gap in the generation sequence (Figure 11). The first alternative is to continue processing and ignore what has happened. This is totally acceptable. However, it should be noted what happens to the volume rotation when a gap is left in the catalog for a generation sequence.

Example 15:

single volume

(file INVØ2 accidently scratched)

```
// JOB DECAT
// LBL INV-1
// LFD COR
// DECAT COR
/&
```

multivolume

(disk pack DISK Ø2 accidently destroyed)

```
// JOB DECAT
// LBL INV-1
// LFD COR
// DECAT COR
/&
```

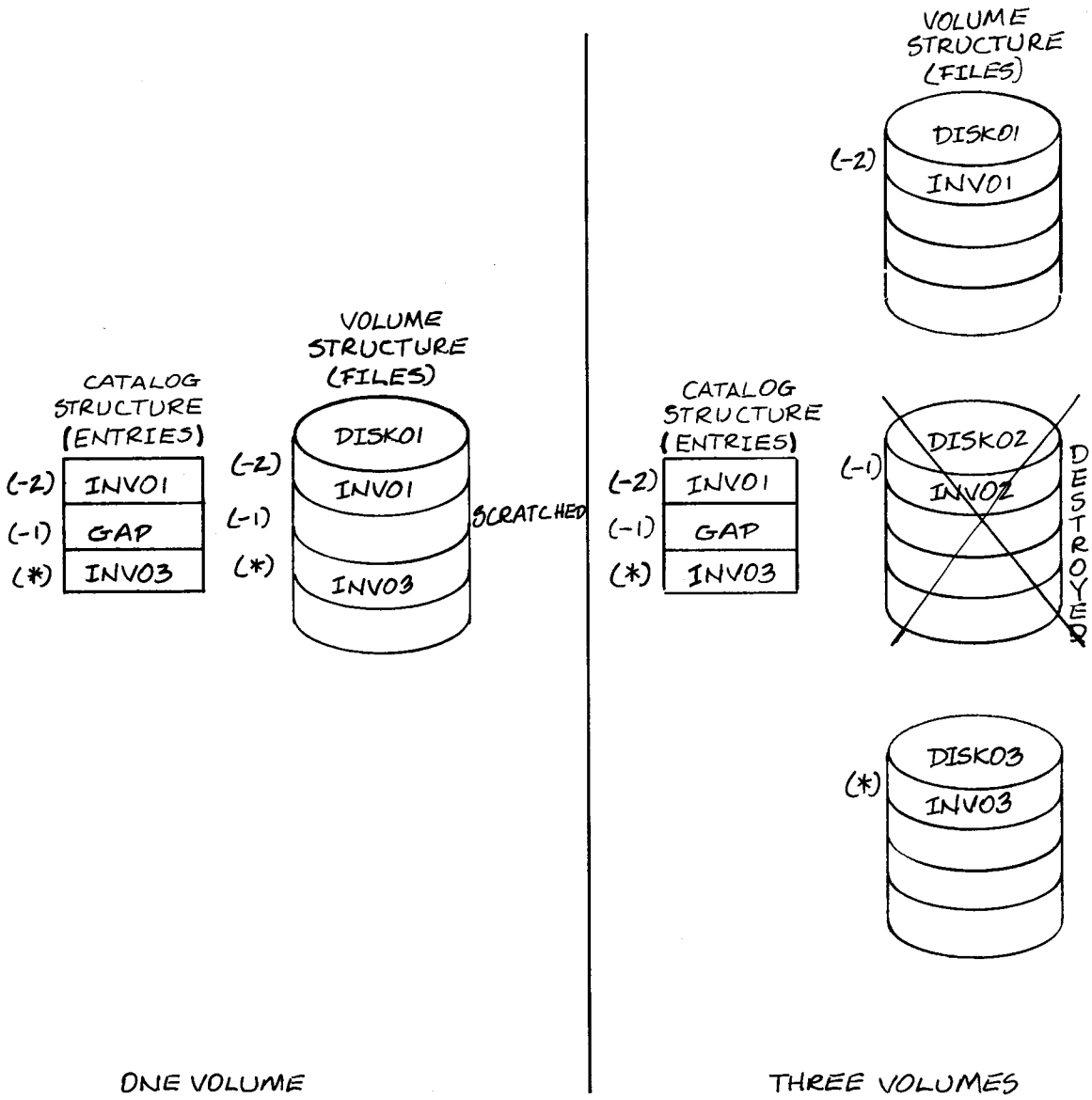


FIGURE 11. MULTIGENERATION CATALOG FILE USAGE. STEP 1

To get around the problem of a missing file or disk pack from the generation sequence, requires no special control statements (see Example 16).

Example 16:

single volume

```
// JOB UPDATE
// LBL INV
// LFD INPUT
// LBL INV+1
// EXT ,,,CYL,1
// LFD OUTPUT
// EXEC MERGE
/$
 new records
/*
/&
```

multivolume

```
// JOB UPDATE
// LBL INV
// LFD INPUT
// LBL INV+1
// EXT ,,,CYL,1
// LFD OUTPUT
// EXEC MERGE
/$
 new records
/*
/&
```

The gap created by a missing file has no real effect in the one volume environment except that the file cannot be accessed without its catalog entry. The file is no longer available for processing (scratched accidently). This causes no problem. The same holds true for the missing disk pack. However, note where INVØ4 is placed. See Figure 12.

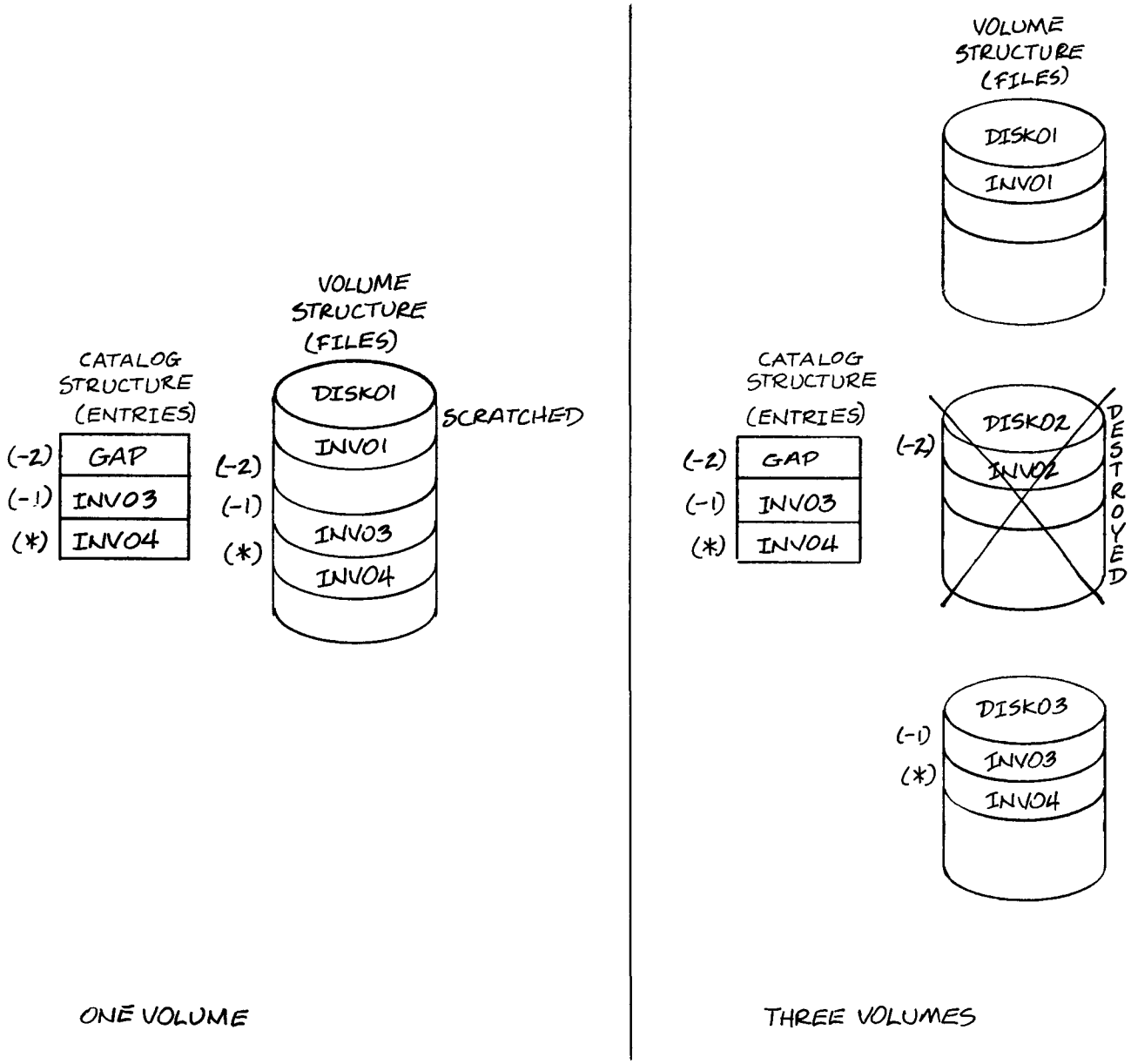


FIGURE 12. MULTIGENERATION CATALOG FILE USAGE. STFP 2

Example 17 and Figure 13 show what can be done to re-create an entry into the catalog so that all previously programmed catalog functions remain operational. Notice in the three volume processing, that replacing the gap with another usable file-id and disk volume, that the volume rotation abilities are retained and that INVØ4 is automatically placed on DiskØ1. Also, note the difference in the job control streams for this example from the previous one.

Example 17:

single volume

```
// JOB UPDATE
// LBL INV
// LFD INPUT
// LBL INV+1
// EXT ,,,CYL,1
// LFD OUTPUT
// EXEC MERGE
/$
 new records
/*
/&
```

multivolume

```
// JOB RECAT
// DVC 6Ø
// VOL DISK2A
// LBL INV-1
// LFD REDO
// CAT REDO,,,MEM
/&
// JOB UPDATE
// LBL INV
// LFD INPUT
// LBL INV+1
// EXT  ,,,CYL,1
// LFD OUTPUT
// EXEC MERGE
/$
 new records
/*
/&
```

Note the location of INVØ4 in the three volume illustration.

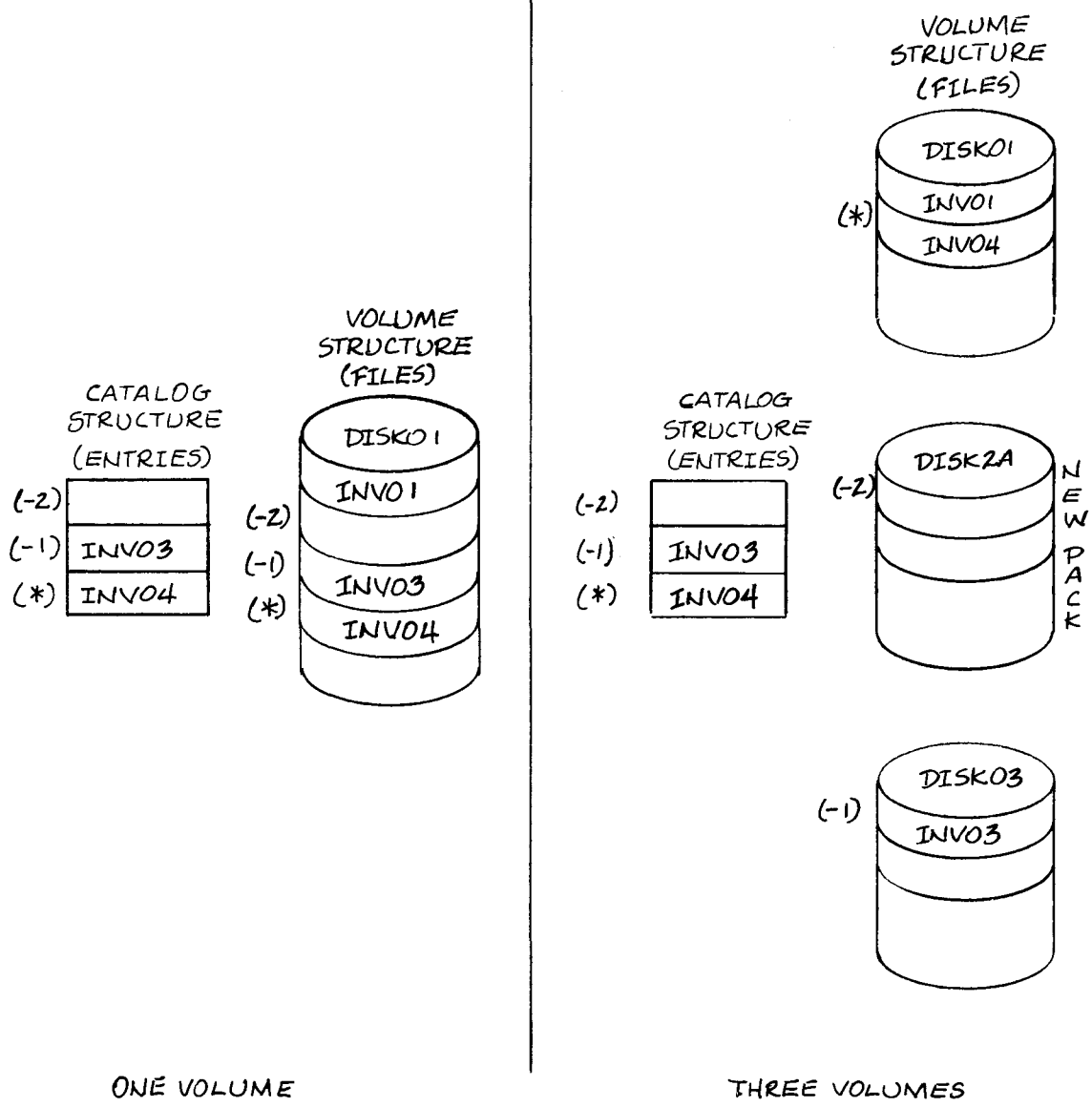


FIGURE 12. MULTIGENERATION CATALOG FILE USAGE. STEP 3

When a File Catalog problem is encountered, follow accepted procedures for the Software User Report (SUR). Helpful information that should be supplied with the SUR is:

1. JC$CAT display of file catalog

2. disk print of YCAT, using the System Utility and

3. job stream where the problem was encountered.

VIII. CHANGES FOR CURRENT RELEASE

The following are changes in Rel. 6.Ø and correction number 3Ø4 for Rel. 5.Ø.

A. DECAT Statement

A new parameter value has been established for the fourth positional parameter of the DECAT statement. This is the ROLLBACK parameter. Only the first three characters (ROL) of the parameter are required.

The format of the DECAT statement is:

// symbol DECAT lfdname [,catpw][,SCR] [,{GEN / ROL}]

Rollback is used when there is a need to reset a generation file's status to a previous condition, i.e., back to the status which existed prior to the last update of the file catalog for the file in question.

Rollback will remove the current member of a generation file, making the previous member current, and insert the current member as the oldest member of the generation, with appropriate adjustment to label.

Rollback can only be used against the current member of a generation file, and can only apply once to the current member within a single job. Thus, within a single job a particular generation file can be reset by one member. More than one unique generation file can, however, be rolled back within a single job.

NOTE:

The new status of the generation file being rolled back can only be accessed in a subsequent job.

As an example of rollback, assume that a generation file (// LBL FILE) has been established and used with four generations (GEN=4) and maintained in the file catalog. The current status of the generation file is:

```
FILE25      current-3      VSN=A
FILE26      current-2      VSN=B
FILE27      current-1      VSN=C
FILE28      current        VSN=D
```

It has been determined that the last update of the file catalog must be reprocessed.

The following statements can then be used:

```
// LBL FILE
// LFD RESET
// DECAT RESET,,,ROL
```

The new status of the generation file is:

```
FILE24      current-3      VSN=D
FILE25      current-2      VSN=A
FILE26      current-1      VSN=B
FILE27      current        VSN=C
```

The statements in this example restore the generation file to the condition that existed before the last member (VSN=D) was added to the file.

During rollback, if the current member to be removed is also to be scratched from its volume, the following sequence could have been used:

```
// LBL FILE
// LFD RESET
// DECAT RESET,,SCR,ROL
```

If a new volume-serial-number, X, is desired for the oldest member established during rollback, the following sequence could be used:

```
// DVC 67
// VOL X
// LBL FILE
// LFD RESET
// DECAT RESET,,,ROL
```

The new status of the generation file is:

```
FILE24      current-3      VSN=X
FILE25      current-2      VSN=A
FILE26      current-1      VSN=B
FILE27      current        VSN=C
```

If a generation file does not have a full complement of members, the current member will be removed, but a new oldest member will not be inserted. A warning message of partial rollback (R521) will be provided by the Run Processor for this condition (see Job Control Error Messages).

The absence of a full complement can be indicated by the following for a file created with GEN=4:

```
FILE00      current-2      VSN=A
FILE01      current-1      VSN=B
FILE02      current        VSN=C
```

A rollback against this file will produce:

```
FILE00      current-1      VSN=A
FILE01      current        VSN=B
```

B. CAT Statement

A new parameter value has been established for the fourth positional parameter of the CAT statement. This is the MEMBER parameter. Only the first three characters (MEM) of the parameter are required.

The general format of the CAT statement is:

```
//symbol CAT lfdname[,catpw][,SCR][, GEN=nn ]
                                  [  MEM    ]
```

Member is used when a gap has to be filled among members of a generation file. This gap must have been established in a job prior to the one containing the MEM parameter. A DECAT statement and a CAT statement with the MEM parameter cannot be used against the same member of a generation file in the same job.

A full device assignment set (DVC-LFD sequence) must be used to reestablish a member.

For example, a generation file (// LBL FILE) has been established and used with three generations (GEN=3). A non-current member has been decataloged. The current status of the generation file is:

```
FILE32        current-2        VSN=A
(gap)
FILE34        current          VSN=C
```

It is desired to fill the gap.

The following statements can be used:

```
// DVC 90
// VOL B
// LBL FILE-1
// LFD FILL
// CAT FILL,,,MEM
```

The new status of the generation file is:

```
FILE32      current-2      VSN=A
FILE33      current-1      VSN=B
FILE34      current        VSN=C
```

The most likely use of the MEM parameter is to replace the oldest member of a generation, in order to insert a new volume-serial-number to be picked up at the next incrementation of that generation.

Thus, if the status of the generation file is

```
FILE75      current-2      VSN=A
FILE76      current-1      VSN=B
FILE77      current        VSN=C
```

The oldest member of the decataloged using

```
// LBL FILE-2
// LFD REMOVE
// DECAT REMOVE
```

leaving the generation file status as

```
(gap)
FILE76      current-1      VSN=B
FILE77      current        VSN=C
```

In a separate job, a new member can be inserted using

```
// DVC 90
// VOL X
// LBL FILE-2
// LFD REP
// CAT REP,,,MEM
```

The new status of the generation file is then

```
FILE75      current-2      VSN=X
FILE76      current-1      VSN=B
FILE77      current        VSN=C
```

The next update, using

```
// LBL FILE+1
```

will pick up volume-serial-number X.

C. LFD Statement

It is frequently required that a tape volume be prepped prior to its use as an output file. When the tape volume in question contains a generation file, it is advantageous to prep the volume without having to give the explicit volume-serial-number designation.

A new parameter value has been established for the third positional parameter of the LFD statement in order to accomplish this. The parameter is PREP, and indicates that the tape volume(s) for the file in question are to be prepped prior to use.

The format of the LFD statement is:

```
//[symbol] LFD {filename }[, n ] [,{ACCEPT }]
             {*filename}[  8 ]  [  { EXTEND }]
                                [  { INIT   }]
                                [  { RELOD  }]
                                [  ,PREP    ]
```

For example, given the generation file (// LBL FILE, GEN=3) status:

```
FILE15      current-2      VSN=A
FILE16      current-1      VSN=B
FILE17      current        VSN=C
```

the statements

```
// LBL FILE+1
// LFD OUTPUT,,PREP
```

will have the same effect as

```
// DVC 90
// VOL A(PREP)
// LBL FILE+1
// LFD OUTPUT
```

The new status of the generation file is:

```
FILE16          current-2          VSN=B
FILE17          current-1          VSN=C
FILE18          current            VSN=A
```

R531

A write password has been specified on a LBL statement, but there is no write password for this file in the file catalog.

R530

A read password has been specified on a LBL statement, but there is no read password for this file in the file catalog.

R528

Rollback is being requested in a DECAT statement against a file which is either not a generation file or not the current member of a generation file.

R527

A request has been made to decatalog an entire generation file, but the file identified in the DECAT statement is not a generation file.

R526

A member of a generation file, which is to be inserted through the use of the MEM parameter of the CAT statement, is already present in the file catalog.

R525

A CAT or DECAT statement is being used more than once against the same file.

R524

The label for the file to be cataloged as a generation file is too long. The maximum for tape files is 15 characters; the maximum for disk files is 42 characters.

R522

The file to be cataloged does not reside on tape or disk.

R521

True rollback cannot be achieved for the file identified in a DECAT statement. Only partial rollback will be performed.

R520

The file identified for decataloging is not in the file catalog. This message will only appear when a full DVC-LFD sequence is used to identify the file.

R518

A MEMBER parameter has been encountered on a DECAT statement, or if on a CAT statement, the sign used with the LBL statement is not negative.

R517

A ROLLBACK parameter has been encountered on a CAT statement. It can only appear on the DECAT statement.

R463

A non-current member of a generation file has not been found in the file catalog. This indicates the presence of a gap between consecutive generation members.

USER COMMENT SHEET

Your comments concerning this document will be welcomed by Sperry Univac for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

_____	_____	_____
(System)	*(Release)*	*(Level)*

_____

(Document Title)

_____	_____	_____	_____
(Issue Number)	*(Revision Number)*	*(UP- Number)*	*(Revision Number)*

Comments:

CUT

From:

_____

(Name of User)

_____

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY

SPERRY UNIVAC

ATTN.: SERIES 90
SOFTWARE CONTROL

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

CUT

FOLD