UNISYS

OS/3 File Cataloging

Technical Overview

Relative to Release
Level 8.2

Priced Item

August 1987

Printed in U S America
UP-9982

UNISYS

OS/3 File Cataloging

Technical Overview

Previous Title: OS/3 File Cataloging
Concepts and Facilities

Relative to Release
Level 8.2

Priced Item

August 1987

Printed in U S America
UP-9982

PAGE STATUS SUMMARY

ISSUE: Update A – UP-9982
RELEASE LEVEL: 8.2 Forward

Part/Section	Page Number	Update Level
Cover		A
Title Page/Disclaimer		A*
PSS	1	A
Preface	1, 2	Orig.
Contents	1 thru 3	Orig.
1	1 thru 4	Orig.
2	1 thru 13	Orig.
3	1 thru 10	Orig.
4	1 thru 15	Orig.
5	1 thru 6	Orig.
6	1 thru 4	Orig.
7	1 thru 10	Orig.
8	1 thru 6	Orig.
Appendix A	1, 2	Orig.
Appendix B	1	Orig.
Appendix C	1, 2	Orig.
Index	1 thru 4	Orig.
User Comment Form		

Part/Section	Page Number	Update Level

Part/Section	Page Number	Update Level

*New page

All the technical changes are denoted by an arrow (⟹) in the margin. A downward pointing arrow (⇓) next to a line indicates that technical changes begin at this line and continue until an upward pointing arrow (⇑) is found. A horizontal arrow (⟹) pointing to a line indicates a technical change in only that line. A horizontal arrow located between two consecutive lines indicates technical changes in both lines or deletions.

Preface

This manual is one of a series designed to instruct and guide the user in the operation of the SPERRY Operating System/3 (OS/3). It specifically describes the concepts of the catalog file and its effective use. Its intended audience is the system administrator (or other programmers authorized to control the contents of the catalog file).

This concepts and facilities manual is divided into the following sections:

- Section 1. File Cataloging Overview

  Presents an overview of the advantages of file cataloging, the distinction between the catalog file and cataloging files, and what types of files can be cataloged.

- Section 2. Cataloging and Decataloging Files

  Details the procedures for cataloging, referencing, and decataloging files in the catalog file.

- Section 3. Cataloging and Decataloging Generation Files

  Explains generation files. It parallels Section 2 in detailing the additional procedures required for cataloging, referencing, and decataloging generation files.

- Section 4. Maintaining Generation Files

  Describes how to build and maintain a sample generation file in the catalog file.

- Section 5. The Catalog Manipulation Utility JC$CAT

  Presents a general discussion of the catalog manipulation utility, JC$CAT. Describes what JC$CAT can perform and the basic requirements for its use.

- Section 6. Protecting the Catalog with a Password

  Explains the function of a catalog password and describes specifically how JC$CAT is used to assign, change, or void a catalog password.

- Section 7. Other Functions of JC$CAT

  Details how JC$CAT is used to print the catalog file, and dump and restore the catalog file.

- Section 8. Cataloging Related Files

  Describes how to group related cataloged files through the use of a qualifier.

- Appendix A. Statement Summary

  Presents the job control statements for file cataloging and the control statements of the catalog manipulation utility.

- Appendix B. Job Control Options to Be Respecified when Processing Cataloged Files

  Lists job control options that must be respecified when processing cataloged files.

- Appendix C. Statement Conventions

  Presents statement format conventions.

The current version of the Job Control User Guide, UP-9986 is helpful when cataloging files.

Contents

8. CATALOGING RELATED FILES

APPENDIXES

FIGURES

TABLES

1. File Cataloging Overview

1.1. WHY CATALOG FILES?

Cataloging device assignment

Cataloging is a procedure using job control statements to store the device assignment sets of files in a centralized directory called the catalog file. The catalog file is an independent system file under the control of the system administrator. The cataloging procedures give the system administrator and designated programmers a way to protect files against unauthorized access, and a convenient way to keep information about a file or multiple generations of a file: a file is cataloged for reasons of security or convenience.

File security

The catalog file allows an OS/3 user (normally the system administrator) to build a file that contains data sensitive to the operation or well-being of the company or its employees, such as payroll data, and to restrict its use and accessibility by unauthorized users. This is achieved by cataloging the file with read and write passwords. A file that is cataloged with a read password can only be read if the read password is supplied. Likewise, a file that is cataloged with a write password cannot be written to unless the proper write password is supplied. In addition, once a file is cataloged, its physical location remains confidential: the only information a programmer needs to know when referencing a cataloged file is its name and any passwords. The system administrator determines which files are cataloged, what passwords they are assigned, and who will know the passwords.

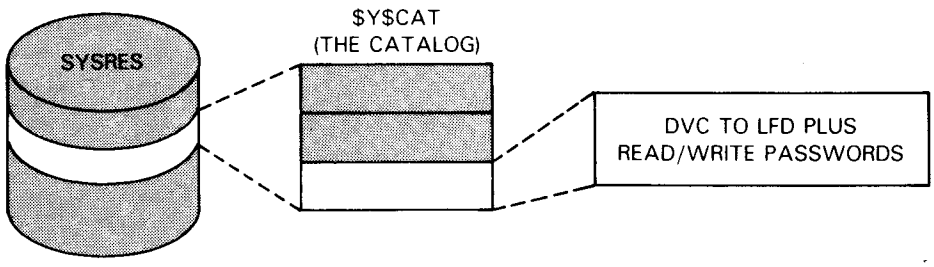
Convenience for individual files

The catalog file allows convenient referencing of files. It allows a user to catalog a file that is expected to be used by multiple users. Once cataloged, a file can be referenced in multiple job streams by simply referring to its cataloged name. The fact that a file is cataloged eliminates the need to have the explicit device assignment set for the file in each job stream that references the file.

Convenience for generation files

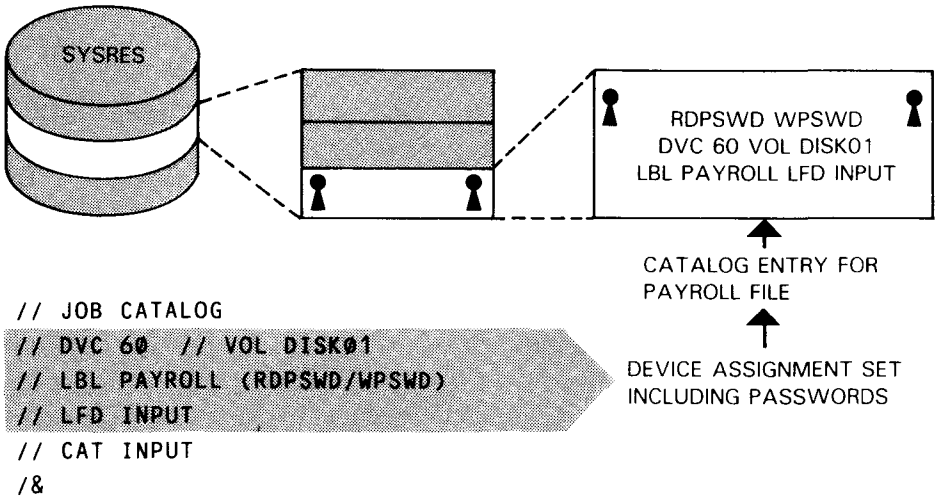
The catalog file also allows you to conveniently record, reference, and process generation files. In OS/3, a generation file is created from a previous file that is processed in combination with current data to create an entirely new copy of the file with up-to-date material, while the old copy remains intact. The updating of an inventory file is a common example of this type of file: the existing file and current inventory data are used as input to create the updated file (new generation), while the old generation is maintained in case it needs to be referenced at a later time. Although each generation of a generation file is a physically separate file, they can be logically linked through the catalog file. The files that make up a generation file can then be processed by the same job that processed the original version of the generation file, and no changes to the job control statements or program code contained in the job need occur. The latest version of a cataloged generation file is always available for processing unless the user specifically requests otherwise.

1.2. THE CATALOG AND CATALOGING



Catalog file

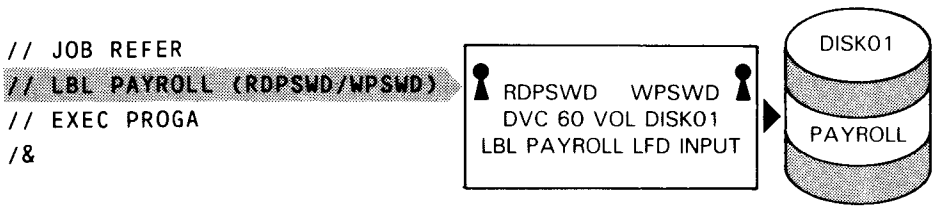
The catalog file of the SPERRY Operating System/3 (OS/3) is a system file that is created automatically on your system resident volume during the installation of the operating system. The name of the system catalog file is YCAT, and it is used to store the device assignment sets of files that have been cataloged. In this manual we will refer to the system catalog file as the catalog.



Cataloging a File (with Passwords)

Cataloging

Cataloging is a process that uses job control language to place a file's device assignment set into the catalog. Accordingly, when a file is cataloged, its device assignment set, not its contents, is recorded in the catalog. If a file is cataloged with read/write passwords, the passwords are also stored in the catalog. The CAT job control statement is used to catalog a file.



Referencing a Cataloged File

Referencing a catalog file

After a file has been cataloged, it can be identified in a job control stream through its file label on the // LBL job control statement: the device assignment information is not required since it is already contained in the catalog. Also, if the file was cataloged with passwords, the appropriate password must be supplied on the // LBL job control statement or the catalog will deny access to the file.

A specific set of job control statements are used to catalog and decatalog (remove) device assignment sets in the catalog. For the convenience of the programmer who will be cataloging and decataloging files, the job control statements applicable to file cataloging that are described in this manual (and presented in summary form in Appendix A) are also contained in the job control programmer reference.

1.3. WHO CAN CATALOG FILES

Limiting access to the catalog

If you are a system administrator and want to limit access to the catalog for reasons of security, you can assign a password to the catalog. The ability to catalog files will then be limited to yourself and those users knowing the password. On the other hand, if you decide to use the catalog as a convenience facility, you don't need a password. In this case, anyone can catalog (and decatalog) files and print or make copies of the catalog.

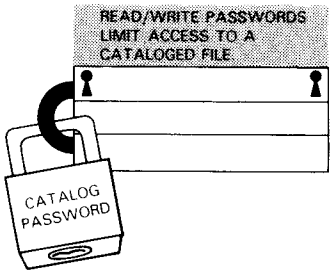
1.4. THE JC$CAT CATALOG UTILITY

JC$CAT function

The catalog password is assigned through the catalog manipulation utility, JC$CAT. JC$CAT is designed for use by the system administrator because it enables him to have full control over access to the catalog. In addition to assigning a password to the catalog, JC$CAT is used to print a list of the catalog contents, duplicate the catalog, and change or void read/write passwords that have been assigned to cataloged files. If your catalog is password protected, that password must be specified whenever you use JC$CAT.

Catalog password

The catalog password should not be confused with the read/write passwords assigned to cataloged files. The catalog password protects the catalog from unauthorized use by limiting access to the contents of the catalog, and restricting who can catalog and decatalog files. Read/write passwords, on the other hand, limit access to read or write to a cataloged file, and they are not assigned through JC$CAT.



CATALOG PASSWORD LIMITS ACCESS TO THE CATALOG

1.5. WHAT FILES CAN BE CATALOGED

Any disk, diskette, or tape file can be cataloged.

2. Cataloging And Decataloging Files

2.1. CATALOGING FILES

Introduction

In this section, we discuss the procedures you use to catalog and decatalog files. But before we discuss these topics, you should know that if you are going to use the catalog for security purposes, you may use JC$CAT to assign a catalog password before any files are cataloged. If you want to assign a password to the catalog prior to cataloging files, see Sections 5 and 6.

Catalog with CAT

To catalog a file, you prepare a job stream containing the device assignment set (DVC–LFD sequence) and the CAT job control statement.

CAT function

The CAT job control statement directs OS/3 job control to catalog the file.

CAT format

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

lfdname parameter

The only required parameter of the CAT job control statement is the logical file name (LFD name). The LFD name you specify for this parameter must agree with the lfdname parameter in the previous LFD job control statement. Thus, the CAT job control statement must follow the device assignment set of the file being cataloged.

catpw is the catalog password

If you are cataloging a file into a catalog that is password protected, you must specify the password in the catpw parameter of the CAT job control statement. In Figure 2–1b, you see an example of the specification of the catalog password.

SCR, GEN, and MEM parameters

The SCR parameter allows you to scratch a file and is the last topic in Section 2. The GEN and MEM parameters apply only to generation files and are discussed in Sections 3 and 4, respectively.

Examples: Cataloging individual files

Some typical examples of job control streams used to catalog files are shown in Figure 2–1.

```
// JOB CATALOG
// DVC RES
// LBL PAYROLL
// LFD INPUTA
// CAT INPUTA
/&
```

a. File on SYSRES

```
// JOB CATALOG
// DVC 6Ø
// VOL DISKØ1
// LBL INVENTORY
// LFD CATFIL
// CAT CATFIL,PSWD1
/&
```

b. File on a specific disk, catalog password protected

Figure 2–1. Job Control Streams Used to Catalog Files

A few parameters of the device assignment set are not retained in the catalog and must be respecified later when you reference the file. See Appendix B for a listing of those parameters.

ESCORT users

NOTE:

All data files used with ESCORT programming language may be cataloged using the method described in this section. If you are already in ESCORT programming language, you can catalog your data file using the RV command without exiting ESCORT language. In addition, the only time the ESCORT programming language library file ESC$ESCORT.LIBRARY.FILES can be cataloged is when there are no other ESCORT programming language users on the system.

2.2. ASSIGNING READ/WRITE PASSWORDS TO FILES

Function of read/write passwords

Whether or not your catalog is password protected, the catalog offers you a security option that enables you to assign read and write passwords to an individual cataloged file. The ability to assign read/write passwords is an important security feature because a cataloged file without password protection can be read, written to, or scratched by any user knowing the file's name. Access to read or write to a file that has been assigned read/write passwords, however, is limited to those people who have been supplied with the passwords.

Special LBL format to assign passwords

To assign a read and write password to a file for the purpose of controlling file access, you add a specification to the LBL job control statement:

```
// LBL file-identifier[(rpw/wpw)]
```

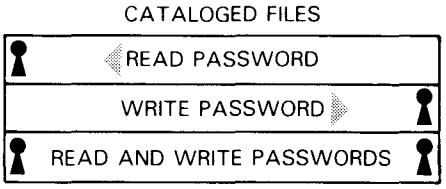
Password parameters

The read password is specified first and separated from the write password by a slash. Each password can be from one to six alphanumeric characters, and the parameter must be enclosed in parentheses. For example:

```
// LBL PAYROLL(PAYR/PAYW)
```

Assigning passwords

Passwords can be assigned individually or in combination; that is, you can assign to a file a read password, a write password, or both. If only a read password is assigned to your file, just the read password must be specified to read from the file. The same is true for the write password. If both the read and write passwords are assigned to your file, you need specify only the password required for that particular operation. We discuss file referencing in 2.5.



Example: Assigning read/write passwords

To catalog a file with read/write passwords into a password protected catalog, use the following job stream:

```
// JOB CATALOG
// DVC 60  // VOL DISK01
// LBL PAYROLL(PAYR/PAYW)
// LFD INPUTA
// CAT INPUTA,PSWD1
/&
```

Other implications of passwords

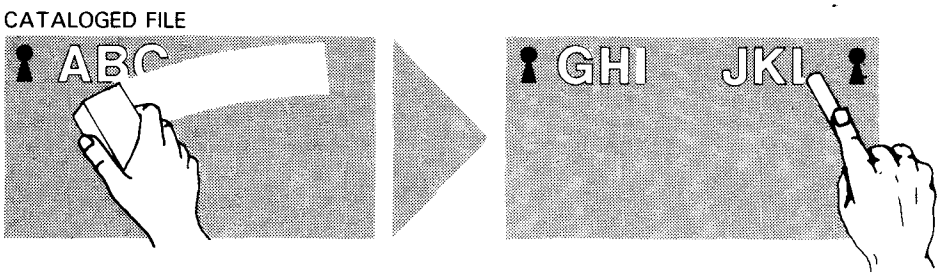
NOTE:

There are two points to remember when you catalog a file with read/write protection:

- *Password verification takes place even if the file is not referenced through the catalog file.*
- *When renaming or scratching a cataloged disk file, you must specify the write password. (Since anyone can rename or scratch a disk file that is protected only by a read password, it's a good idea to assign a write password along with your read password.)*

2.3. ADDING, CHANGING, OR VOIDING READ/WRITE PASSWORDS

Changing read/write passwords on files is particularly convenient for the system administrator, who needs the ability to control passwords on each file in the system. You may, for example, want to add a write password to a cataloged file for which only a read password existed. Or, you may want to change an existing password on a file or delete the read/write passwords protecting a file.



Changing Read/Write Passwords

JC$CAT requirement

The catalog utility, JC$CAT, is used to add, change, or void read/write passwords for individual cataloged files; this is the only instance where a JC$CAT operation is performed on an individual cataloged file entry (not the catalog itself). The other functions of JC$CAT and the general requirements for executing JC$CAT are discussed in Section 5.

Executing JC$CAT with FIL and CFP

To add, change, or void a read or write password, you execute JC$CAT using the FIL and CFP control statements.

FIL precedes CFP

The FIL job control statement precedes the CFP statement in the job control stream. (We discuss the FIL statement in detail in 5.2.) Its format is:

FIL format

```
FIL{Dn}=filename-1[/password-1][...,{Dn}filename-n[/password-n]]
   {Tn}                           [   {Tn}                    ]
```

CFP function

The CFP control statement is used to add, change, or delete a read or write password. Its format is:

CFP format

```
CFP Dn,file-id{[,RDOLD=old-password],RDNEW={new-password}}
              {                        {NOPASSWORD    }}
              {[,WROLD=old-password],WRNEW={new-password}}
              {                        {NOPASSWORD    }}
```

Required parameters

The CFP control statement consists of the 2-character identifier for the logical catalog file name (Dn), the cataloged file identifier, and the read or write passwords that need to be added, changed, or voided.

In the following job stream, both the read and write passwords of the PAYROLL file are changed:

Example: Changing both passwords

```
// JOB ADJUST
// DVC 20  // LFD PRNTR
// DVC RES  // LBL $Y$CAT  // LFD CATFIL
// EXEC JC$CAT
/$
      FIL D1=CATFIL
      CFP D1,PAYROLL,RDOLD=XXXpay,RDNEW=YYYpay,WROLD=XXXyap,
            WRNEW=YYYyap
/*
/&
```

The next job stream shows a write password being added to a file that was originally assigned only a read password:

Example: Adding a password

```
// JOB ADDPW
// DVC 20  // LFD PRNTR
// DVC RES  // LBL $Y$CAT  // LFD CATFIL
// EXEC JC$CAT
/$
      FIL D1=CATFIL
      CFP D1,NAMES,WRNEW=new-password
/*
/&
```

Voiding a password

To delete either a read or write password, identify the catalog file and the file entry, specify the old password, and, in the area for the new password, place the keyword NOPASSWORD.

In the following example, a previously assigned read password is voided:

Example: Voiding a password

```
// JOB VOIDPW
// DVC 20  // LFD PRNTR
// DVC RES  // LBL $Y$CAT  // LFD CATFIL
// EXEC JC$CAT
/$
      FIL D1=CATFIL
      CFP D1,NAMES,RDOLD=NAMRPW,RDNEW=NOPASSWORD
/*
/&
```

NOTE:

A separate CFP control statement is needed for each file requiring a password change.

2.4. RULES FOR FILE CATALOGING

General rules

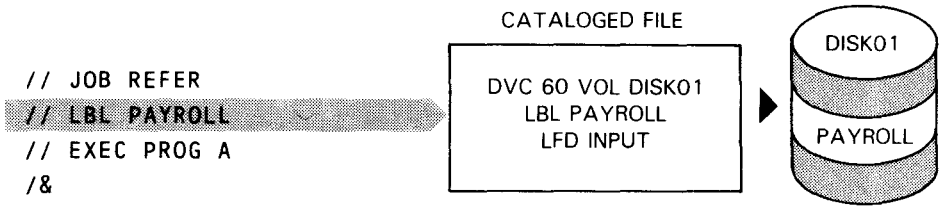
As you can see from the previous paragraphs, it's easy to catalog a file. There are some general rules, however, that you must observe:

1. The name of the file being cataloged (as it appears on the LBL job control statement) cannot be the same as that of a previously cataloged file.
2. The device assignment set for the file to be cataloged must precede the CAT job control statement in the job control stream.
3. A cataloged file may not be referenced in the same job control stream in which it was entered in the catalog. This is because a file is cataloged at the end of the job (/& job control statement), not as it is encountered in the job control stream.

2.5. REFERENCING CATALOGED FILES

Referencing

Once a file is cataloged, you can reference (access) it in a job control stream by identifying the file's name in the LBL job control statement (// LBL file-id). There is no need to include the device assignment set information (DVC and VOL statements) because the catalog already contains that information. The elimination of a majority of job control statements is one of the benefits of cataloging.



Referencing a Cataloged File

Referencing an unprotected file

To reference a cataloged file that is not password protected, you use the LBL job control statement. The format of this LBL statement is the same as the LBL statement in the device assignment set when the file was originally cataloged:

LBL format without passwords

```
// LBL file-identifier
```

Examples: Cataloging and referencing an unprotected file

Figure 2–2 presents an example of three job control streams. In Figure 2–2a, a file is cataloged, and in 2–2b and 2–2c the cataloged file is referenced.

```
// JOB CATALOG
// DVC 60
// VOL DISK 01
// LBL PAYROLL
// LFD INPUTA
// CAT INPUTA
/&
```

a. Cataloging

```
// JOB REFER
// LBL PAYROLL
// EXEC PROGA
/&
```

b. Referencing

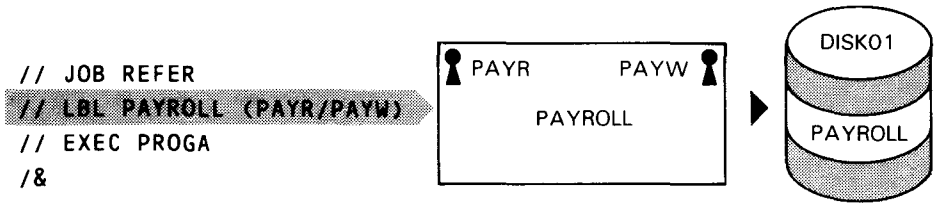
```
// JOB REFER
// LBL PAYROLL
// LFD INPUTX
// EXEC PROGA
/&
```

c. Referencing when your program uses a different LFD name

Figure 2–2. Job Control Streams Used to Catalog and Reference a File (Unprotected File)

When to use // LFD

The catalog in Figure 2–2a is not password protected, nor is the PAYROLL file protected by a read or write password. So, only the LBL job control statement with the file identifier is needed to reference the cataloged file (Figure 2–2b) as long as the LFD name in your program matches the cataloged LFD name. If, however, you are referencing a cataloged file and the LFD name in your program is different from the cataloged LFD name, you will need to supply the LFD job control statement. For example, if the LFD name in your program is INPUTX, and your program is going to reference the PAYROLL file that was cataloged in Figure 2–2a, your job control stream would look like Figure 2–2c. You would also use the LFD statement if you want to override a specification (for example, extent space) on the LFD statement.



Referencing a protected file

Referencing a cataloged file that is password protected also requires the LBL job control statement. The format of the LBL statement is the same as when the file was originally cataloged, except you need to specify a read password for a read operation or a write password for a write operation.

Specifying only the required password

LBL format with passwords

```
// LBL file-identifier(rpw/wpw)
```

Examples: Referencing protected files

In Figure 2–3, two password protected files are cataloged and referenced.

```
// JOB CATALOG
// DVC 20 // LFD PRNTR
// DVC RES // LBL PAYROLL(PAYR/PAYW)
// LFD INPUTA
// CAT INPUTA
// DVC 64 // VOL SPL040
// LBL NEWTAX(TAXR)
// LFD INPUTB
// CAT INPUTB
/&
```

a. File cataloging

```
// JOB REFER
// DVC 20 // LFD PRNTR
// LBL PAYROLL(PAYR/PAYW)
// LBL NEWTAX(TAXR)
// EXEC PROGA
/&
```

b. File referencing

Figure 2–3. Job Control Streams Used to Catalog and Reference Password Protected Files

The job control stream for file cataloging in Figure 2–3a shows that both files (PAYROLL and NEWTAX) are cataloged with passwords. The job control stream for file referencing shows how the passwords are specified when referencing the files. Here again, you need only a reference by the LBL job control statement to access the file entries PAYROLL and NEWTAX (assuming that their LFD names in program PROGA are also INPUTA and INPUTB).

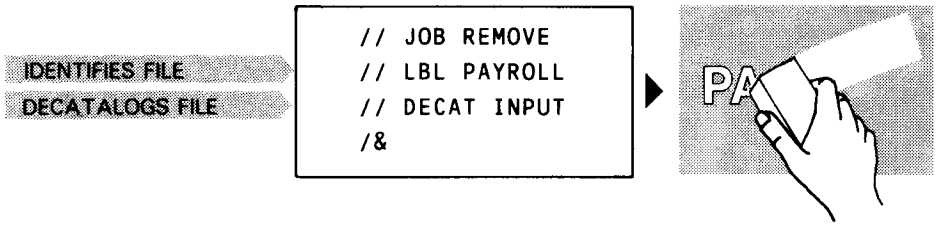
Referencing restriction

NOTE:

You can't reference a cataloged file in the same job stream in which it was entered in the catalog. A file is cataloged at the end of the job (/& job control statement), not as it is encountered in the job stream.

2.6. REMOVING FILE ENTRIES FROM THE CATALOG (DECATALOGING FILES)

It is as simple to remove a file entry from the catalog as it is to catalog the file. The job control stream used to decatalog a file includes the LBL job control statement followed by the DECAT job control statement. The device assignment set for the file is not required since it is already contained in the catalog.



Specifying write password

To remove a file from the catalog, you must first identify the file on the LBL job control statement. The parameters required on the LBL job control statement to decatalog a file are the same as those needed to catalog a file. There is one exception to this: if you cataloged a file with read/write passwords, you must specify at least the write password to remove the file from the catalog.

DECAT function

The DECAT job control statement directs OS/3 job control to decatalog a file.

DECAT format

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

lfdname and catalog password parameters

The lfdname parameter of the DECAT job control statement must be the same as the filename parameter of the LFD job control statement when the file was originally cataloged. Also, if the catalog is protected by a password, specify that password on the DECAT statement or access to the catalog will be denied.

Other DECAT parameters

Of the remaining parameters of the DECAT job control statement, the SCR parameter allows you to scratch a file when it is decataloged. The GEN and ROL parameters apply only to generation files.

Example: Cataloging and decataloging a file

Figure 2–4a shows a job control stream used to catalog a file, and Figure 2–4b shows the job control stream used to decatalog that same file.

```
// JOB CATALOG
// DVC 6Ø
// VOL DISKØ1
// LBL EMPNUM
// LFD INPUTN
// CAT INPUTN,PSWD1
/&
```

a. File cataloging

```
// JOB REMOVE
// LBL EMPNUM
// DECAT INPUTN,PSWD1
/&
```

b. Decataloging

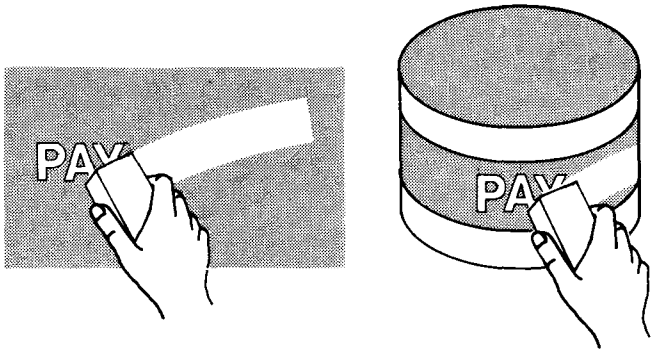
Figure 2–4. Job Control Streams Used to Catalog and Decatalog a File (Unprotected File)

The following subsection tells you how to scratch a file in addition to removing its entry from the catalog.

2.7. SCRATCHING CATALOGED FILES

When scratching occurs

The procedures described in the preceding section tell you how to remove a file entry from the catalog so the file no longer can be accessed through the catalog. You have the option of scratching the file at the same time it is decataloged. When a file is scratched, the file identification is removed from the disk or diskette (tapes are not scratched). Once the file identification is removed, the system will no longer recognize the file, and the area occupied by the file can be used for other purposes.



CAT or DECAT scratch option

The optional scratch parameter SCR is offered on both the CAT and DECAT job control statements. You can specify it during one of two points in the cataloging/decataloging process – when the file is cataloged, or when it is decataloged.

CAT scratch parameter

Choosing to use the SCR parameter on the CAT statement will not cause your file to be immediately scratched; it will be automatically scratched when it is decataloged. (It is not necessary to include the scratch parameter in the DECAT job control statement at that time). The optional SCR parameter is shown in the CAT format:

CAT format for scratch option

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

Example: Using CAT scratch option

Figure 2–5a shows a job control stream that can be used when the file is cataloged to specify automatic scratching of a file when it is decataloged. Figure 2–5b is the job control stream used to decatalog and scratch the file.

```
// JOB CATALOG
// DVC RES
// LBL PAYROLL
// LFD CATFIL
// CAT CATFIL,PSWD1,SCR
/&
```

a. Cataloging using SCR parameter

```
// JOB REMOVE
// LBL PAYROLL
// DECAT CATFIL,PSWD1
/&
```

b. Decataloging and scratching

Figure 2–5. Job Control Streams Using the CAT Scratch Option

DECAT scratch option

If automatic scratching was not established when a file was originally cataloged, you have the option of specifying it when you decatalog the file by including the SCR parameter on the DECAT job control statement. If you use this parameter, a single file is both removed from the catalog and scratched. Remember, if you originally cataloged the file using the SCR parameter on the CAT job control statement, you do not need to specify it again in the DECAT job control statement:

DECAT format for scratch option

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

Example: Using DECAT scratch option

Figure 2–6a shows a sample job control stream used to catalog a file. Figure 2–6b is the job control stream used to decatalog and scratch the file.

```
// JOB CATALOG
// DVC RES
// LBL PAYROLL
// LFD CATFIL
// CAT CATFIL,PSWD1
/&
```

a. Cataloging without using the scratch parameter

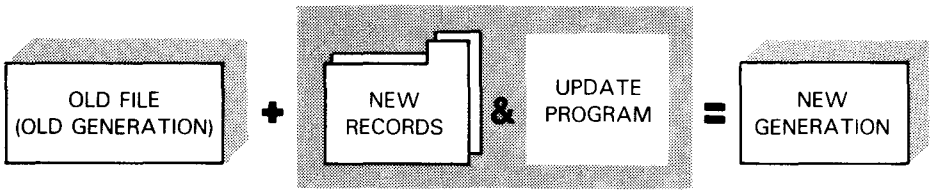
```
// JOB REMOVE
// LBL PAYROLL
// DECAT CATFIL,PSWD1,SCR
/&
```

b. Decataloging using the scratch parameter

Figure 2–6. Job Control Streams Using the DECAT Scratch Option

3. Cataloging and Decataloging Generation Files

3.1. DEFINING A GENERATION FILE



So far, our discussion of the catalog has been limited to the cataloging of individual (nongeneration) files. Now, we will discuss the cataloging of generation files. The process of cataloging, decataloging, and referencing generation files is built upon the same basic principles used by individual files. However, additional specifications are needed for performing these same functions with generation files. In this section, we discuss the additional material you need when cataloging generation files.

Generation files defined

First, an explanation of generation files. A generation file is a file created from a previous file. All you need to create a generation file is the previous file and the new material you want to add. The old file is kept intact, while the new generation of the file is an entirely new copy of the file with up-to-date material. Contrast this with a case where the new generation is created by reading the old generation up to the end and then adding the new material to the end of the old file. Such a process is not reproducible, since the *end* of the old information is no longer identifiable.

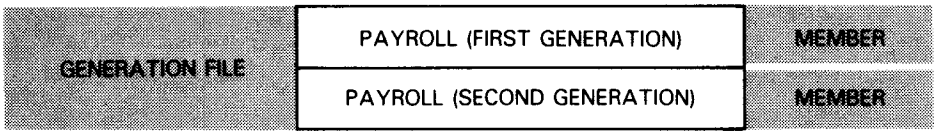
Purpose of generation files

Accepted practice is to keep enough previous generations on hand so that any operator, equipment, or program error can be detected and corrected before the backup or current information is discarded. So, as you can see, a major advantage of generation files is data protection.

Cataloging generation files for convenience

Without the catalog, the creation and maintenance of generation files is cumbersome. For instance, simply creating a new generation involves a lengthy job control stream that includes the unique file identifier, and the device assignment sets for both the previous file and the file being created (new generation). In this section, you will see how cataloging eliminates this complexity as well as many of the other problems normally associated with generation files.

3.2. CATALOGING GENERATION FILES



Generation file consists of members

The set of independent files that make up the generation file are related by a label in the catalog. Each independent file represents a distinct generation and is called a member of a generation file. Cataloging the first member of a generation file is similar to cataloging a nongeneration file: the job control stream contains the device assignment set and a CAT job control statement.

Use CAT once

However, the job streams you use to catalog subsequent members of a generation file do not contain the CAT statement, unlike nongeneration files that require a CAT statement for each file being cataloged. Once again, you can catalog disk, diskette, and tape files.

To review, the format of the CAT job control statement is:

CAT format

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

CAT parameters

You may recall from 2.1 that when a file is being cataloged, the logical file name (LFD name) is always required, and if the catalog is password protected, the password must be specified on the CAT statement. (The SCR parameter applies to the automatic scratching of files and is optional when cataloging files.) Up to this point, we have not discussed the GEN or MEM parameters. The GEN parameter is an additional specification required for generation files and is discussed here. The MEM parameter is explained in 4.5.

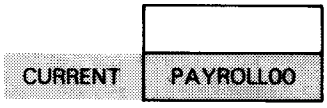
GEN function

You use the GEN parameter to specify the maximum number of generations to be maintained in the catalog (GEN=nn). The catalog can maintain up to 100 generations (members) for a generation file. The first member of a generation file is called generation 00, the second generation is 01, and so on.

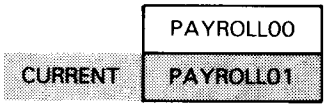
Example: Illustration of GEN function

This example illustrates how the catalog follows the instruction of the GEN parameter. Here we are instructing the catalog to maintain two generations of the file, PAYROLL.

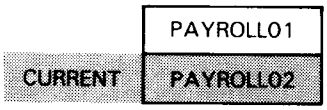
This is the catalog after we have cataloged the first generation.



This is the catalog after we have cataloged the second generation. Notice that it is now the current generation.



This is the catalog after we have cataloged the third generation. The first generation has been dropped, the second generation has replaced the first generation, and the third generation is now the current file.



Cataloging first generation

The GEN parameter of the CAT job control statement is used in the same job control stream that catalogs the first member of the generation file. For example, suppose you decide to use the catalog to conveniently maintain a frequently updated file called PAYROLL. Assume that the first generation has been created and is now to be cataloged. The catalog is to maintain two generations for this file, GEN=02. The following job stream is used to accomplish both of these functions.

Example: Cataloging first generation

```
// JOB CATALOG
// DVC 60  // VOL DISKØ1
// LBL PAYROLL
// LFD CATFIL
// CAT CATFIL,PSWD1,,GEN=Ø2
/&
```

The above job stream catalogs the first generation of the PAYROLL file, which is named PAYROLL00. Remember, PAYROLL00 has already been created.

+n parameter to catalog a new generation

Adding a subsequent generation to the generation file requires an additional specification to the file identifier on the LBL job control statement: the generation number, +n. The +n parameter is an optional parameter of the // LBL job control statement, and as far as cataloging goes, the +n generation number provides the value for the creation of a new member of a generation file. The file identifier affixed with a +1 generation number indicates to the catalog that another member is to be added. For example, // LBL PAYROLL+1 adds a new generation to the payroll generation file.

Example: Creating and cataloging the second generation

To continue the PAYROLL example, when it is time to update the payroll file, you only need one job control stream to both create and catalog the second generation. This is shown in the following job stream:

```
// JOB UPDATE
// LBL PAYROLL
// LFD INPUT
// LBL PAYROLL+1
// EXT MI,C,,CYL,1
// LFD OUTPUT
// EXEC UPDATE
/$
.
.  new records
.
/*
/&
```

In the preceding job stream, the update program (UPDATE) is executed using the first generation and new records to create the second generation, PAYROLL+1, which is cataloged. Because a new file is being created, the EXT job control statement is used. Notice that the CAT job statement is not used; it is only needed to catalog the first generation. The second generation is named PAYROLL01.

Major advantages of cataloging

At this time we can point out two advantages of using the catalog for generation files:

1. Using the same file identifier that the file was originally cataloged under will always reference the most current generation of the file.
2. The same file identifier that the file was originally cataloged under, in combination with the +n generation number, can always be used when creating and cataloging a new generation.

Cataloging subsequent generations

With these advantages in mind, you can see that the job control stream used to create and catalog the second generation of the PAYROLL file is also used to create and catalog the third generation. In this case, // LBL PAYROLL will reference the second (current) generation, and the second LBL job control statement // LBL PAYROLL+1 is used to create and catalog the third (next) generation. In the preceding example, the third generation is named PAYROLL02. (The two digits attached to the file identifier are discussed in 3.3.)

Single and multiple volumes

NOTE:

Members of a generation file may be located on a single volume or multiple volumes. For example, instead of storing the entire generation file on a single volume, you can assign the first generation to DISK01, the second to DISK02, the third to DISK03, and so on. The two methods of storing generation files do not affect your ability to catalog them. In fact, the job streams used to catalog members of generation files that reside on multiple volumes are similiar to those used to catalog members of generation files that reside on a single volume. We will discuss this in detail in Section 4.

3.3. REFERENCING CATALOGED GENERATION FILES

–n and nn for generation file referencing

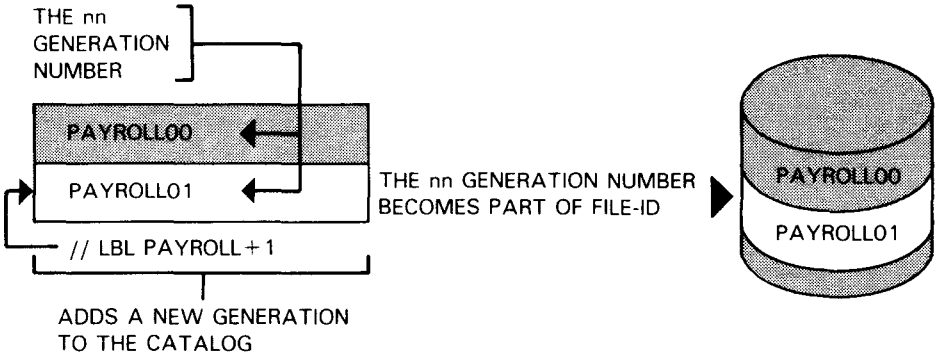
As we stated previously when describing how to catalog a generation file, the file identifier on the LBL job control statement is expanded to include generation numbers for generation files only. We spoke of the +n designation for cataloging. For example, PAYROLL+1 catalogs a new generation of the PAYROLL file. There are two other generation number specifications that you use to reference (access) a cataloged generation file: –n or nn. These two specifications, in addition to +n, are included as optional parameters in the LBL job control statement:

Special LBL format (with generation numbers)

```
//[symbol] LBL file-identifier [{+n}] [(rpw/wpw)]
                               [{-n}]
                               [{nn}]
```

File identifier limitations

When the generation number parameter is used, the total number of characters for the file identification is limited to 15 for tape and diskette and 42 for disk file.



Origin of nn parameter

You have the option to reference a file with the –n parameter or the nn parameter. First, the nn parameter. Each time a member of a generation file is cataloged using the +n parameter, the catalog automatically assigns to the new generation a 2-digit generation number, the nn parameter. This generation number becomes part of the file identifier, and each member is assigned a generation number that is one greater than the previous generation. Generation numbers begin at 00 and go to 99, and if the generation number exceeds 99, it goes back to 00. For example, the original file (first generation) of the PAYROLL file is named PAYROLL00, the second generation is PAYROLL01, and so on.

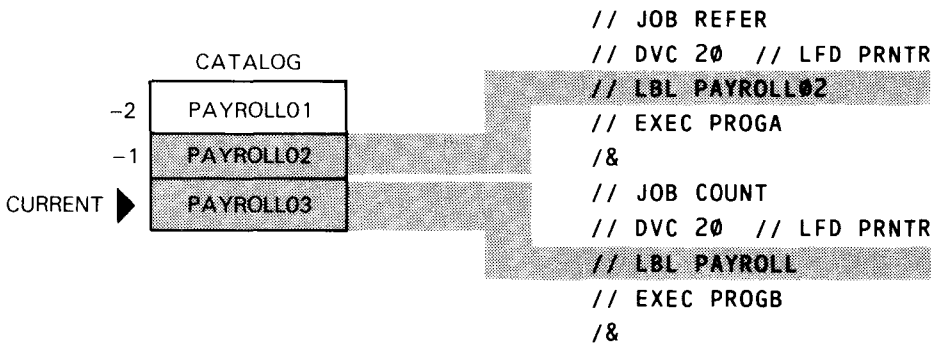
Using nn to reference a specific generation

To reference a specific generation, use the file identifier with the nn generation number:

```
// LBL file-identifier nn
```

Example: Referencing with nn

The following job control streams use the nn parameter to reference a generation file identified as PAYROLL. The first stream references the third generation of the file, and the second references the current generation of the file.



Referencing the current generation

As shown in the second job stream, if you do not specify a generation number when referencing a generation file, you receive the current generation of the file. In this way, you can always be assured of getting the most up-to-date file available.

–n parameter

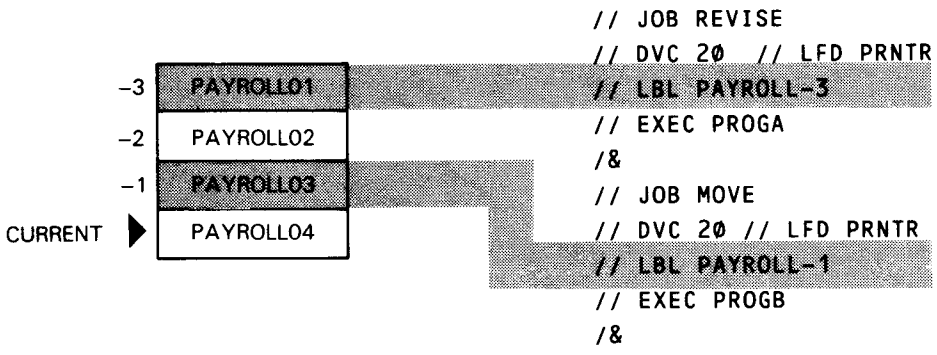
Another preferred way to reference a generation file is by using the –n parameter. You use the –n parameter to reference a noncurrent generation that is being maintained by the catalog. For example, if the catalog file is maintaining three generations of the payroll file (GEN=03),

```
// LBL PAYROLL-2
```

accesses two generations previous to the current generation of the PAYROLL file.

Examples: Referencing using –n

The following job control streams use the –n parameter to reference the generation file, PAYROLL, where GEN=04. The first job stream references the third generation previous to the current generation, and the second job stream references one generation previous to the current generation.



We have now covered all the variations of the generation number parameter of the LBL job control statement.

NOTES:

-n restrictions for interactive facilities

1. *The -n parameter cannot be used to reference a file when using the interactive read or write command. You must use the nn parameter.*

Referencing a member when not maintained by the catalog

2. *When a file is dropped from the catalog because of the limitations set in the GEN parameter, it is not scratched and it keeps its file identifier, including the nn portion. Therefore, you must include the nn generation number with the file identifier when referencing the file. In addition, since the file is no longer being maintained by the catalog, the job control stream used to reference it must also include the full device assignment set.*

3.4. DECATALOGING GENERATION FILES

LBL precedes DECAT

The process used to remove (decatalog) a generation file from the catalog is similar to that used to remove a nongeneration file (2.6). To decatalog a generation file entry from the catalog, the file must first be identified in a job stream using the // LBL statement. The parameters required on the LBL job control statement to decatalog a generation file are the same as those used to catalog a generation file (3.2). After the file has been identified, it is decataloged by using the DECAT job control statement.

Review of DECAT format

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

GEN and ROL for generation files

We explained, under decataloging nongeneration files, all parameters in this statement with the exception of the GEN and ROL parameters. These two parameters are used only when decataloging generation files. The ROL parameter is discussed in 4.2 because it is used as a maintenance function of generation files in the catalog.

Decataloging all members (GEN)

The GEN parameter is the important specification in the removal of generation files from the catalog. It allows you to collectively decatalog all members of a generation file from the catalog.

Example: Decataloging all members

The following job control stream uses the GEN parameter. In this example, all members of the generation file PAYROLL are decataloged:

```
// JOB REMOVE
// LBL PAYROLL
// DECAT CATFIL,,,GEN
/&
```

Decataloging a member

If you want to decatalog only a single member of a generation file, do not use the GEN parameter of the DECAT job control statement. Instead, in the // LBL job control statement, specify the files exact identification – including its generation number. For example, // LBL PAYROLL03 or // LBL PAYROLL-1.

Example: Decataloging a single member

The following job control stream is used to decatalog a single member of a generation file:

```
// JOB REMOVE
// LBL PAYROLLØ3
// DECAT CATFIL
/&
```

3.5. SCRATCHING GENERATION FILE MEMBERS

Scratching a member

Scratching a member of a generation file is similar to scratching a nongeneration file (2.7). To delete a member of a generation file, you use the scratch parameter [,SCR] of the CAT or DECAT job control statement. Using the scratch parameter in the CAT statement will not cause your file to be immediately scratched; it will be scratched when it is decataloged. (It is not necessary to include the scratch parameter in the DECAT job control statement at that time.) When scratching is specified in the DECAT job control statement, the file identified is both decataloged and scratched in the same operation.

Example: Scratching a member

In the following job control stream, a specific generation of the PAYROLL file is both removed from the catalog file and scratched:

```
// JOB DELETE
// LBL PAYROLLØ2
// DECAT CATFIL,PSWD1,SCR
/&
```

Scratching for security

If your generation file is password protected and located on disk or diskette, be sure to scratch a member immediately after it is dropped from the catalog. This is important because once a member is dropped from the catalog, any passwords assigned to it are not applicable.

Scratching restrictions

NOTE:

You can scratch only one member of a generation file at a time. You can't scratch all generations of a file collectively. This applies whether scratching is specified from either a CAT or DECAT job control statement. You can, however, collectively decatalog all generations of a file by use of the GEN parameter. The GEN parameter removes only the catalog entries; it does not scratch the files. Therefore, the GEN parameter of the DECAT job control statement should not be used in conjunction with the SCR parameter:

4. Maintaining Generation Files

4.1. AN EXAMPLE – BUILDING AN INVENTORY GENERATION FILE

Introduction

In this section, we use examples to show how to build a generation file. Our discussion focuses on the differences between the job streams used for single-volume generation files and those used for multivolume generation files.

The following paragraphs also give you a picture of what occurs in the catalog as we proceed through the various steps contained in the examples.

Assume the generation file in our example represents an inventory file where the stock inventory is frequently updated. First, we place an entry for the file in the catalog. Then we build the file on disk using a tape as our input file, and show how to create and catalog the next three generations. Finally, we briefly discuss how to reference a file that has been dropped from the catalog.

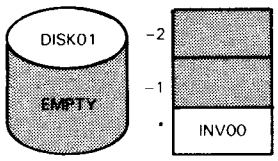
Setting up the catalog to maintain three generations

The inventory file is called INV (// LBL INV). When the file is cataloged, the file identifier is linked to the disk volume where the file resides. The maximum number of generations maintained in the catalog is three (GEN=03). This does not mean that you are limited to three generations of the INV file, but that the catalog reflects only the three most current generations of that file.

Examples: Cataloging the first generation

The first two job streams show the cataloging of the device assignment sets for the first generation file that has not yet been created:

Single Volume

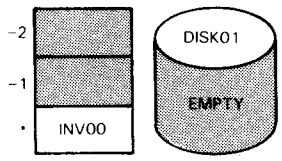


*Current

```
// JOB CATALOG
// DVC 60
// VOL DISK01
// LBL INV
// LFD NEWINV
// CAT NEWINV,,SCR,GEN=03
/&
```

Multiple Volume

```
// JOB CATALOG
// DVC 60
// VOL DISK01
// LBL INV
// LFD NEWINV
// CAT NEWINV,,SCR,GEN=03
/&
```

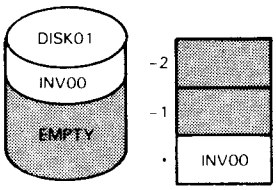


Notice that the job control stream used to catalog this generation file on a single volume is identical to the job stream used in a multivolume environment. You can see that a specific disk type is assigned. Because this is the initial generation of the inventory file, it is recorded as INV00 in the catalog.

Example: Creating and allocating the first generation

The next step in our example creates the inventory file we just cataloged, using input from tape.

Single Volume

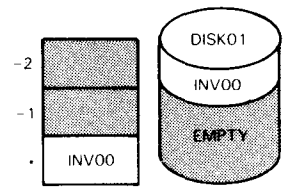


*Current

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

Multiple Volume

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

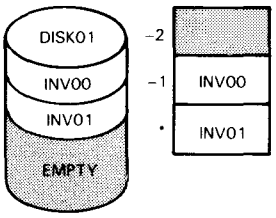


The complete device assignment sets are needed for the tape file. However, the DVC and VOL statements are not needed for the output inventory file because they are already contained in the catalog. The LFD statement is required since it differs from the LFD name used when originally cataloging the file (2.5). Also, the EXT statement is needed because we are creating a new file. As you can see, both single-volume and multivolume job streams are identical.

Examples: Creating, allocating, and cataloging the second generation, INV01

Now we are going to update the inventory file to create and catalog the second generation, INV01. Remember that the first generation (the result of the original cataloging) was INV00.

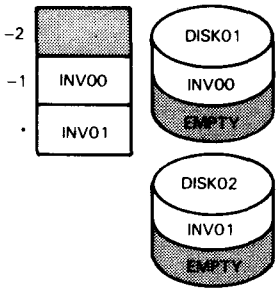
Single Volume




```
// JOB UPDATE
// LBL INV
// LFD INPUT
// LBL INV+1
// EXT,,,CYL,1
// LFD OUTPUT
// EXEC UPDATE1
/$
.
.  new records
.
/*
/&
```

Multiple Volume

```
// JOB UPDATE
// LBL INV
// LFD INPUT
// DVC 60
// VOL DISK02
// LBL INV+1
// EXT,,,CYL,1
// LFD OUTPUT
// EXEC UPDATE1
/$
.
.  new records
.
/*
/&
```



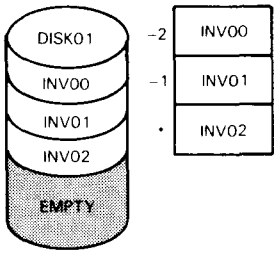
*Current

Again, DISK01 is allocated for single volume use, while DISK02 is allocated for the multiple volume environment. You can see that the two job streams are similar except for the inclusion of the DVC and VOL statements for multiple volumes.

Examples: Creating, allocating, and cataloging the third generation, INV02

We are now ready to create and catalog the third generation of the file, INV02.

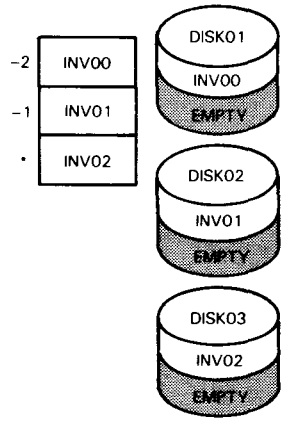
Single Volume




```
// JOB UPDATE
// LBL INV
// LFD INPUT
// LBL INV+1
// EXT,,,CYL,1
// LFD OUTPUT
// EXEC UPDATE1
/$
.
.  new records
.
/*
/&
```

Multiple Volume

```
// JOB UPDATE
// LBL INV
// LFD INPUT
 // DVC 60
// VOL DISK03
// LBL INV+1
// EXT,,,CYL,1
// LFD OUTPUT
// EXEC UPDATE1
/$
.
.  new records
.
/*
/&
```



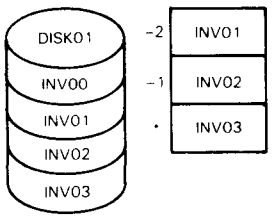
*Current

Because we specified three generations to be maintained in the catalog, the cataloging of the third generation completes the generation file sequence. In this update, DISK01 remains the volume assigned for the single volume use, while DISK03 is the third volume used in the multiple volume stream.

Exceeding the GEN limitation – creating and cataloging the fourth generation (INV03)

The cataloged generation file is now built to our specifications. Each time you exceed three generations of the INV file (as limited by the GEN parameter of the CAT job control statement), the oldest file entry in the catalog is automatically dropped, and the new entry is based on the device assignment of the one being dropped. Thus, the DVC and VOL statements are not required in the multiple volume environment to catalog the fourth and subsequent generations, as long as the device assignment of the entry being dropped is what you want to use. The following job streams are used to catalog the fourth (INV03) and subsequent generations of the inventory file:

Example: Creating, allocating, and cataloging INV03

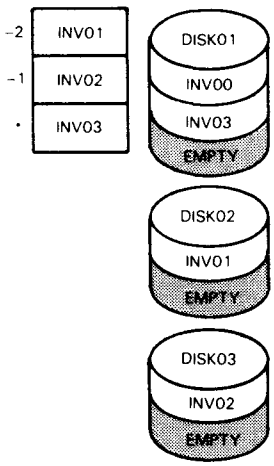


Single Volume

```
// JOB UPDATE
// LBL INV
// LFD INPUT
// LBL INV+1
// EXT,,,CYL,1
// LFD OUTPUT
// EXEC UPDATE1
/$
.
.  new records
.
/*
/&
```

Multiple Volume

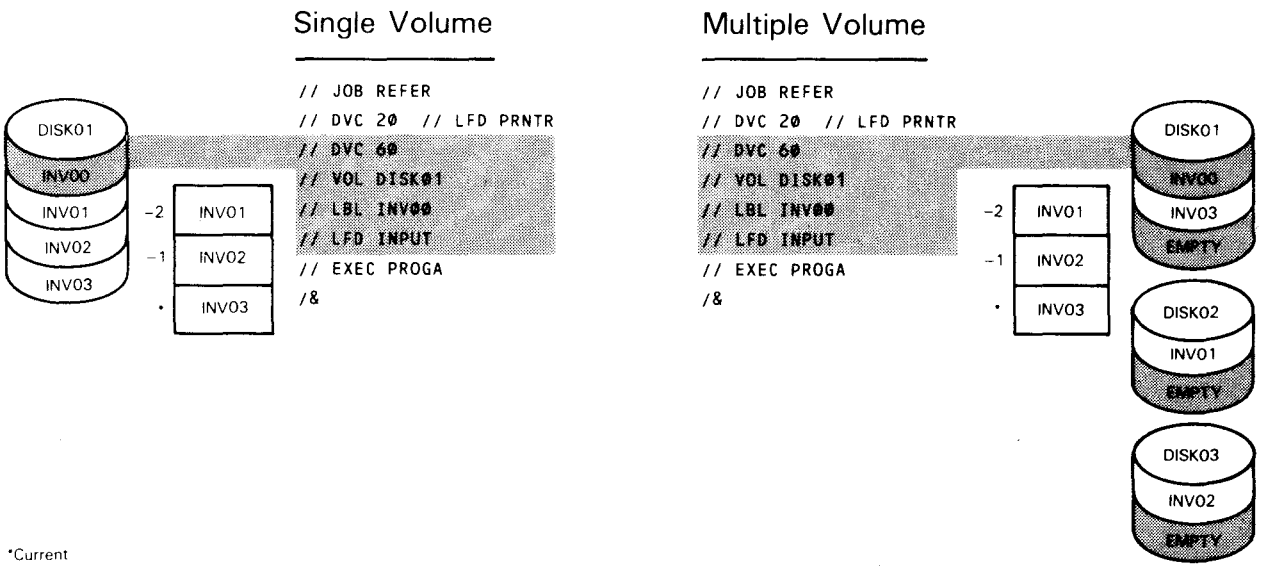
```
// JOB UPDATE
// LBL INV
// LFD INPUT
// LBL INV+1
// EXT,,,CYL,1
// LFD OUTPUT
// EXEC UPDATE1
/$
.
.  new records
.
/*
/&
```



*Current

Example: Referencing INV00

You can see that the two job streams are identical. Remember, a file that has been dropped from the catalog is no longer maintained by the catalog. Thus, the job control stream used to reference a file that has been dropped must include the complete file identifier (as recorded by the catalog) in the LBL job control statement. It must also include the DVC number, VOL number, and LFD statements. For example, when the fourth generation of the inventory file is cataloged, the first generation is dropped by the catalog. The following job streams are used to reference the first generation:



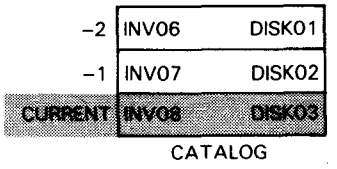
4.2. ROLLBACK – MAINTAINING THE INVENTORY GENERATION FILE

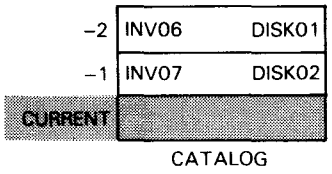
General

Rollback is an optional parameter in the DECAT job control statement. It is used to reset the status of member files maintained by the catalog when the current generation is decataloged. However, before we elaborate on the functions of rollback, we give an example showing what occurs when a current generation is decataloged without using rollback. In this manner, you can fully appreciate the benefits rollback provides.

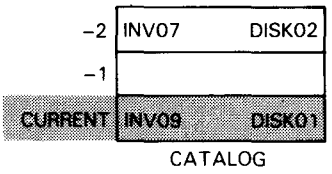
Example: Consequences of not using rollback

If the inventory file we used in the last section was maintained through several generations, with the oldest generation removed with each addition of a new generation, the catalog will look like this.

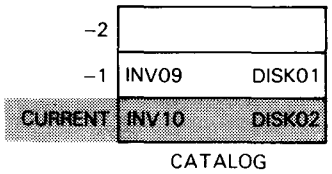




Suppose that the last update to the catalog contains an error, or the program creating it terminated in error, and you must decatalog the file for reprocessing. If you use the DECAT job control statement by itself to decatalog the file, the device assignment sets associated with that file are removed from the catalog, and the status of the remaining members is not automatically readjusted. Thus, the catalog would appear as shown here.



This is undesirable because an attempt to reference INV (the current generation) won't be successful; at this point, the catalog isn't maintaining a current generation. In addition, it results in the loss of DISK03 for the purpose of cataloging members of this generation file. For example, when the next update is cataloged, it is recorded as INV09, and the catalog appears as shown here.



When the next generation is cataloged, it's assigned to DISK02, and the catalog appears as shown here.

When the next generation is cataloged, it's assigned to DISK01 instead of DISK03. You can now see how the gap causes an inefficient use of resources as well as incorrect references. This can be avoided through the use of rollback.

Rollback parameter

Rollback is specified in the DECAT job control statement (ROL parameter).

DECAT format showing ROL option

The complete format for the DECAT job control statement is:

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

Rollback function

Rollback is used to decatalog the current member of a generation file when a generation file has a full complement of members. (If a generation file is set up to maintain three generations and all three are contained in the catalog file, a full complement exists). Using rollback will reset the status of the generation file to that which existed before the last generation was created.

Rollback

1. Moves the catalog entry for the current member of a generation file to the back of the catalog and makes it the oldest member of the generation file
2. Moves the entry for the next most recent member of the file into the current position
3. Moves all other members forward
4. Adjusts the status of each member (current–1, current–2, etc) of the generation file in the catalog

Example: Introduction to using rollback

Using the inventory file as an example, suppose INV08 contains an error and you must reprocess the file. What you want to do is get rid of INV08 and use INV07 as your current inventory file. If you removed only INV08 from the catalog, INV07 would not be in the current position. Instead, it would still be in the current –1 position. To keep the files in the catalog in order, you use the rollback specification.

Example: Job stream using rollback

Figure 4–1 shows the job control statements needed to roll back the current generation of the inventory file.

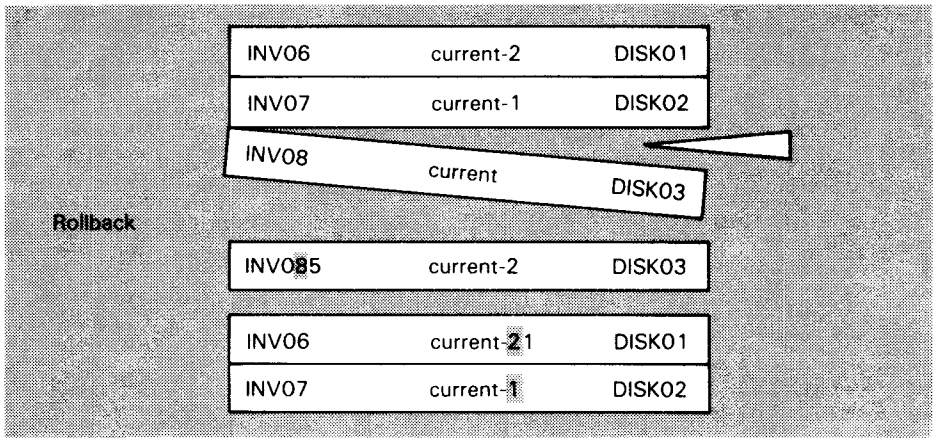
```
// JOB STATUS
// LBL INV
// LFD NEWINV
// DECAT NEWINV,,,ROL
/&
```

Figure 4–1. Job Control Stream Used to Roll Back the Current Member of a Generation File

The following illustration shows the catalog entries for the file before, during, and after the current generation (INV08) is rolled back:

Before rollback

INV06	current-2	DISK01
INV07	current-1	DISK02
INV08	current	DISK03



After rollback

INV05	current-2	DISK03
INV06	current-1	DISK01
INV07	current	DISK02

LEGEND:

XY means that status X is being changed to status Y.

What rollback accomplishes

In this case, rollback	
1	Moves the catalog entry for INV08 to the back of the catalog for that generation file
2	Moves the INV07 file to the current position
3	Moves all other entries forward for the generation file
4	Adjusts the status of each member of the generation file

When rollback is performed on INV08, the entry for that generation is redefined as INV05, making it the oldest generation in the catalog. INV05 takes its volume and device specifications from INV08 and is remaintained by the catalog. Later, when a new INV08 file is created containing the correct information, INV05 is dropped from the catalog and INV08 becomes the current generation.

Option: scratching while using rollback

Suppose that during the rollback operation, you also wanted to scratch the current generation from its volume. The DECAT job control statement in Figure 4–1 would be:

```
// DECAT NEWINV,,,SCR,ROL
```

Option: assigning a new VOL

Another option in rollback is to roll back the current generation and also give it a new volume serial number. To do this, you would include the DVC and VOL job control statements in the job control stream in Figure 4–1 where rollback is performed. For example:

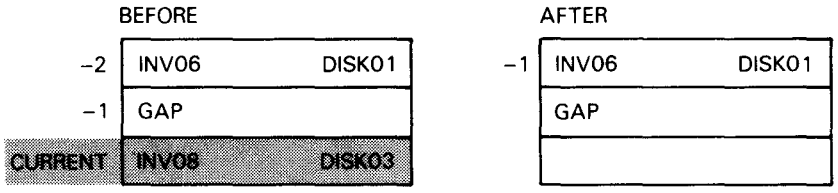
```
// DVC 60  // VOL X
```

Avoid partial rollback

As we have mentioned, rollback can be used only against the current generation when a generation file has a full complement of members. Attempting rollback on a generation file that does not contain a full complement of members will result in a partial rollback. When partial rollback occurs, the catalog does not transfer the DVC and VOL information for the file being decataloged to the oldest entry of the generation file in the catalog. As a result, when the next generation is cataloged, it will be assigned the device information from the most recent generation, not from the oldest entry as is the case with complete rollback. If a partial rollback occurs, you get a run processor warning message to let you know the status of your file.

Example: Partial rollback

The following illustration is an example of partial rollback. The generation file is maintained with three generations. However, only two generations currently reside in the catalog because one generation, INV07, was decataloged. The current generation (INV08) needs to be reprocessed, and rollback is performed. The catalog is shown before and after rollback:



You use the same job stream shown in Figure 4–1 to accomplish rollback. Generation 08 of the inventory file is removed from the catalog, but, because the generation file does not have a full complement of members, the DVC and VOL information from INV08 cannot be inserted as the oldest member of that file. Notice that the status of generation 06 is changed to current –1. With every partial rollback, the catalog entries are reduced by 1. You can see from this example that when the next generation is cataloged, it will be assigned to DISK01 rather than DISK03. This is because DISK03 is no longer maintained by the catalog, so the catalog assigns the next available volume. So, it is important that you perform rollback only when a full complement of members exists.

Rollback limitations

NOTE:

Rollback can be used only once against the current generation within a single job. However, current generations of different files can be rolled back within one job.

4.3. ESTABLISHING CATALOGING SEQUENCE

Eliminating job statements

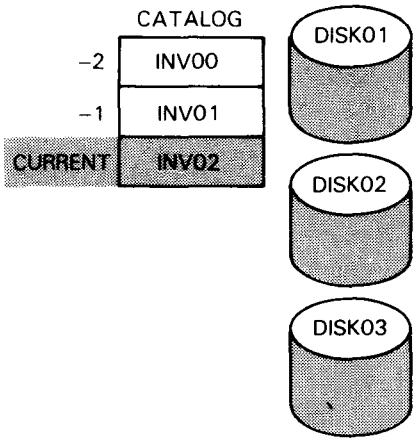
You can catalog all the device assignment sets for a generation file before any members have been created. You do this to establish a sequence of volumes that cataloging will follow in adding subsequent generations. Once established, you won't need to include the DVC and VOL job control statements in the job control streams used to create each member.

Two job streams required

Two job control streams are used to establish cataloging sequence: the first catalogs the first generation, and the second catalogs all subsequent generations. Using the inventory generation file again, you can catalog all three members of the generation file on two jobs, the first job stream catalogs the first generation, and the second job stream catalogs the remaining two generations (GEN=03).

Example: Establishing cataloging sequence for the inventory file

```
// JOB CATALOG
// DVC 60
// VOL DISK01
// LBL INV
// LFD NEWINV
// CAT NEWINV,,SCR,GEN=03
/&
// JOB CAT1
// DVC 61      // VOL DISK02
// LBL INV+1   // LFD NEWINV
// DVC 62      // VOL DISK03
// LBL INV+2   // LFD NEWINV
/&
```



Remember, if you use this method, the current entry in the catalog is INV02, which will be located on DISK03. Therefore, the name of the first generation you create is INV02, not INV00. We continue this example in the following paragraphs.

4.4. AN ALTERNATE METHOD TO CREATE AND CATALOG MEMBERS

Two job streams are used

So far, we have shown you how to use a single job stream to both catalog and create a member of a generation file. Now we'll show you an alternate method to catalog and create members of a generation file. This method uses two separate job streams: the first catalogs a new member, and the second creates the new member.

Advantage

Proper use of this method makes the use of rollback unnecessary after a program failure; that is, if your new generation is cataloged by the first job stream but fails to be created in the second, you can prevent cataloging when rerunning the program by eliminating the first job control stream. As a result, rollback will not be needed between each run to reset the status of the generation file because the status will not change since no cataloging has occurred. For example, if you use the single job stream method and your program fails while creating the new generation, you would need to decatalog the file using rollback before the program (job stream) is run again. This is because the single job stream catalogs a file each time it executes the program. If, on the other hand, your program fails while using this alternate method, you just remove the first job stream (since that generation is now cataloged), and rerun the second job stream for as many times as necessary.

When to use it

The alternate method is used after the normal procedures are followed to catalog and create the first generation. Or, if you use the procedures in 4.3, the current generation must already be created.

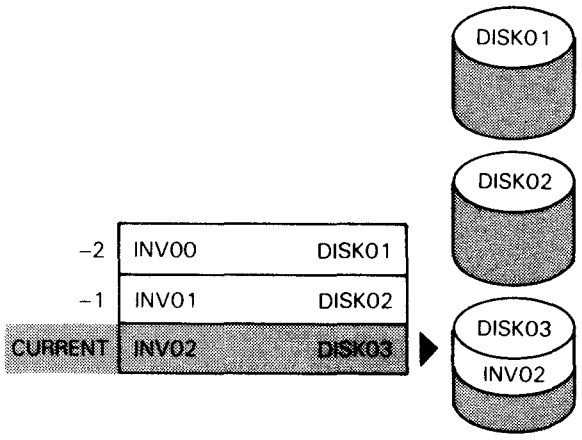
How files are identified

This method uses two job streams instead of one. The first job stream catalogs the new member, displacing the current file (created file) to the −1 position. The second job stream is used to create the new file. Because the file in the −1 position is needed to create the new generation, it is identified in the second job stream as // LBL file-identifier-1, and the new generation (the file being created) is identified as the current file in the catalog // LBL file-identifier.

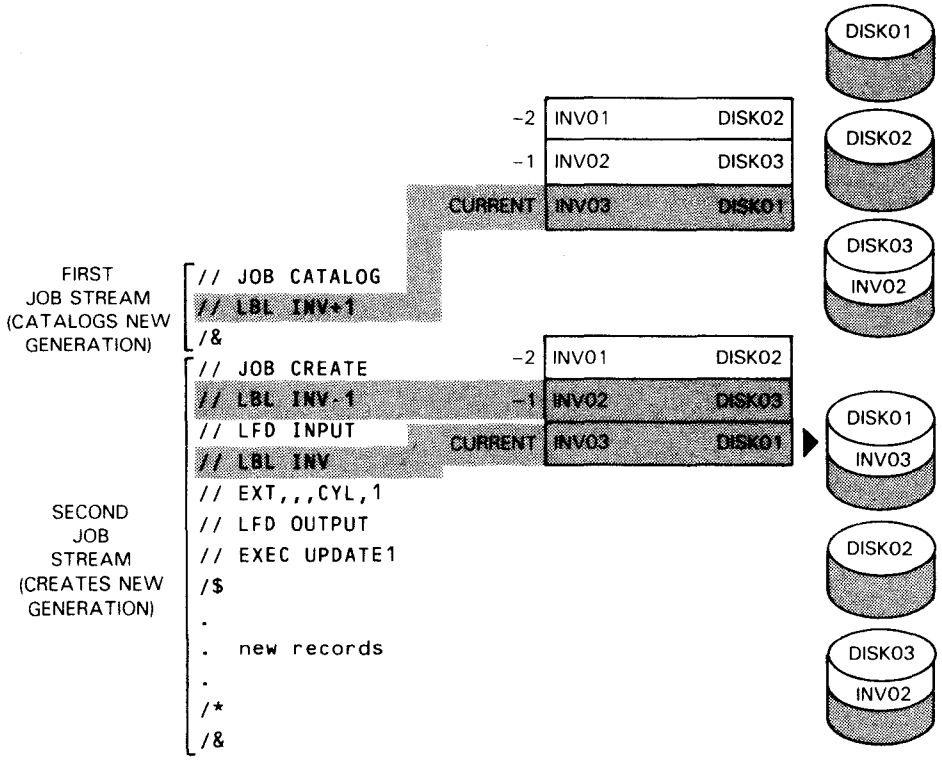
First create current generation

To show you how the alternate method is used, we will return to the sample job stream in 4.3. There we cataloged the device assignment sets needed to create member files before the files are created. Before using the alternate method, however, we must create the current generation, INV02.

```
// JOB CREATE
// DVC 60
// VOL INSAV
// LBL INV
// EXT,,,CYL,1
// LFD OUTPUT
// EXEC BUILD
/&
```



Now that the first generation is created, you can use the alternate method to catalog and create the second generation, INV03.



Explanation

These job streams can be used to catalog and create the second and subsequent members of the inventory file as long as the program (UPDATE1) runs error free. Suppose, though, that your program fails during the creation of INV03. If you used the single job stream method (as in 4.1) you would have to decatalog the entry for INV03 and readjust the status of the catalog using the rollback option each time the program fails. However, if the program fails while using the alternate method, you simply eliminate the first job stream and rerun the second job stream for as many times as necessary. Rollback is not required between each run because, without the first job stream, no incrementation occurs in the catalog. When your program successfully runs, the sequence of files in the catalog will have remained unchanged, and the name of the new generation will match the current file in the catalog, which in this case is INV03.

4.5. CHANGING DEVICE TYPES AND VOLUME SERIAL NUMBERS

MEM function

To replace a previously cataloged device type and volume serial number with a new device type and volume serial number, use the MEM parameter of the CAT job control statement.

Example: Using MEM

For example, using the inventory file again, we set up our catalog so it would maintain three members. Each is located on either DISK01, DISK02, or DISK03.

-2	INV06	DISK01
-1	INV07	DISK02
CURRENT	INV08	DISK03

Suppose we want to replace DISK01 with a diskette having a volume serial number of DSKTX. The following job control stream is used for this purpose:

Job stream using MEM

```
// JOB CHANGE
// DVC 130  // VOL DSKTX
// LBL INV-2
// LFD NEWINV
// CAT NEWINV,,,MEM
/&
```

Now the catalog file appears as shown here.

-2	INV06	DSKTX
-1	INV07	DISK02
CURRENT	INV08	DISK03

General rules

These rules apply when using the MEM parameter:

1. Use a full device assignment set (DVC–LFD) in the job control stream using MEM.
2. Do not use a DECAT job control statement and then a CAT job control statement with a MEM specification against the same generation in one job.

We have now covered all options in the CAT job control statement. The complete format of the CAT job control statement is:

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

5. The Catalog Manipulation Utility JC$CAT

5.1. WHAT JC$CAT DOES

Utility control statements direct JC$CAT

JC$CAT is the name of the catalog manipulation utility. It is executed in a job control stream that includes user-specified control statements. These control statements direct JC$CAT to automatically perform one or more complicated operations on the catalog. The control statements used to operate JC$CAT are discussed later in this text and are presented in summary form in Appendix A.

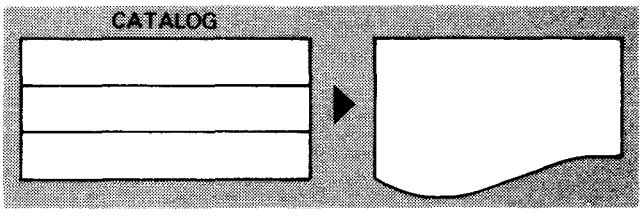
JC$CAT for system administrator

JC$CAT is intended for use by the system administrator, because, as you will see below, the operations performed by JC$CAT involve the security of the catalog and the files it contains.

Four Major Operations Performed by JC$CAT

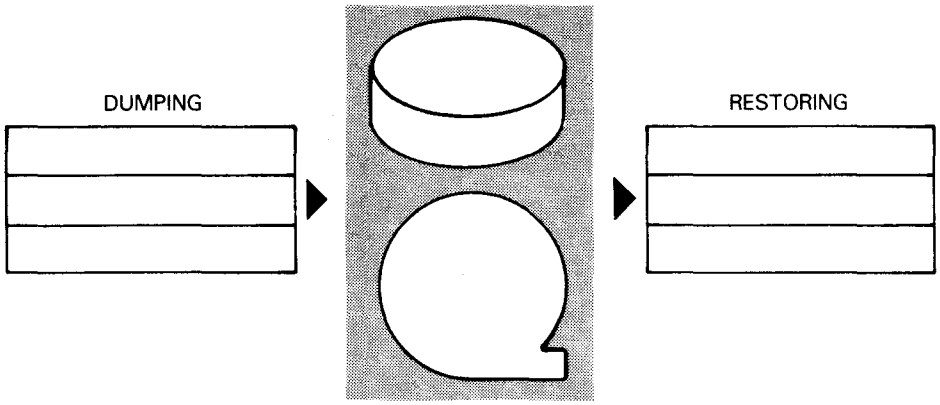
1 Printing the catalog

The print facility provided by JC$CAT offers many convenient aids to listing the contents of the catalog. Depending on your needs, you can print the contents of the entire catalog, all members of cataloged generation files, the current generation of cataloged generation files, or a specific file entry in the catalog.



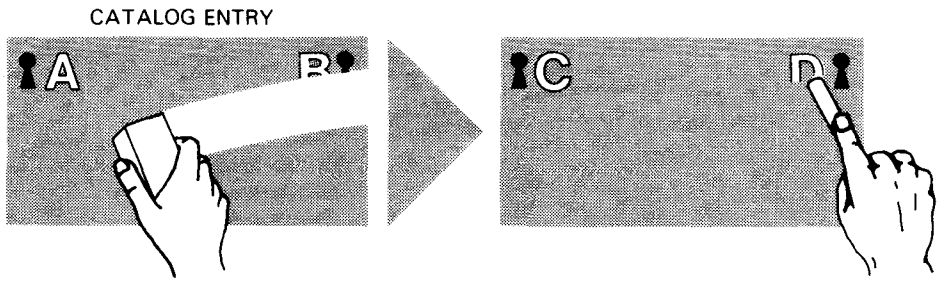
2 Dumping the catalog to another disk or tape file and restoring the catalog to SYSRES

You can use JC$CAT to copy (dump) the catalog to a backup tape or disk file if you want the security of having a second copy of the catalog. This copy can always be restored to your SYSRES volume if anything should happen to affect the integrity of the current version of the catalog. Dumping and restoring the catalog can also be a convenience feature. For example, when you install a new software release or update your operating system to a new release level, the copied catalog can be restored easily to the new SYSRES.



3 Changing read and write passwords assigned to individual cataloged files

You can use JC$CAT to change the read and write passwords assigned to cataloged files. This function is discussed (2.3) following the description of the assignment of read/write passwords.



▶▶▶

4. Assigning, changing, and voiding the catalog password

   If, in your evaluation, you find that file security is of major importance, you can use certain functions of JC$CAT to assign a password to the catalog. With this password assigned, you restrict access to the catalog, because only those programmers (or just the system administrator) who know the catalog password can catalog a file or access the catalog through JC$CAT. However, if your primary use of the catalog is as a convenience facility, your catalog can be set up without a password. Thus, anyone may have access to the catalog.

   Because a catalog password is an important consideration, especially when you are first setting up the catalog, it is discussed in Section 6.

5.2. JOB STREAM REQUIREMENTS FOR JC$CAT

Executing JC$CAT – required statements

As we just mentioned, JC$CAT operates through user-specified control statements. These control statements are placed in the job control stream that executes JC$CAT, and they will vary depending on the operation you want performed. However, any job control stream that executes JC$CAT must include:

1. the device assignment set for the printer;
2. the catalog name; and,
3. the FIL control statement.

Use printer for error messages

The device assignment set for the printer is required when using JC$CAT to output error messages.

The catalog name

The catalog name is identified on the // LBL job control statement. The file identifier for the system catalog (located on the system resident volume) is YCAT. If you make any copies of your catalog, the first six characters of their file identifiers must be YCAT, and their file identifiers cannot exceed 17 alphanumeric characters for tape, or 44 alphanumeric characters for disk.

FIL function

The file association statement (FIL) is a utility control statement that is always required when executing JC$CAT. It identifies the catalog name to JC$CAT.

FIL format

```
FIL {Dn}=filename-1[/password] [...,{Dn}=filename-n[/password]]
    {Tn}                       [   {Tn}                       ]
```

Dn or Tn parameter

The Dn or Tn parameter is the logical catalog file name (lcfn). It is specified by a type code (D when the catalog is on disk and T when it is on tape) and a file number in the range of 0 to 8 (depending on how many copies of the catalog exist, e.g., D0...D8 to T0...T8).

filename parameter

For the filename parameter, you specify the same logical file name that you used on the LFD job control statement. The FIL control statement equates the logical catalog file name (Dn or Tn) to the logical file name from the LFD job control statement. This identifies the control statements used by JC$CAT to the catalog device assignment set.

/password parameter

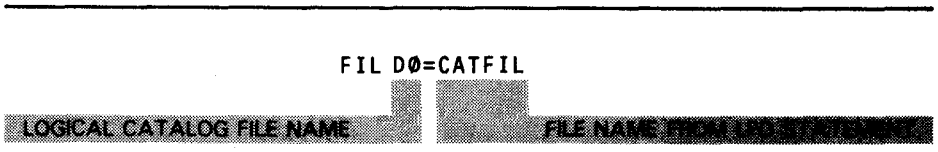
The catalog password is preceded by a slash, and it can be from one to six alphanumeric characters. The password parameter on the FIL control statement is not used if the catalog has no password, but it is used to specify an existing catalog password.

Specifying one or more catalogs

As you can see from the format of the FIL control statement, the parameters are repeated. This indicates that you can specify more than one catalog for each FIL control statement. This is useful when you want to perform JC$CAT operations on multiple copies of the catalog, because you can identify all the copies involved on one FIL control statement.

FIL example

A typical FIL control statement might look like this:



Executing JC$CAT

The following is an example intended to show you the general format of a job control stream used to execute JC$CAT. Each line is numbered and described below.

```
1. // JOB ------
2. // DVC 20  // LBL PRNTR
3. // DVC ---  // LBL $Y$CAT  // LFD ------
4. // EXEC JC$CAT
5. /$
6.       FIL {D0-D8}=-------/password
             {T0-T8}
7.
8. /*
9. /&
```

1. This job control statement indicates the beginning of the job.

2. This is the device assignment set for the printer, which is always required when executing JC$CAT.

3. This is the device assignment set for the catalog. The first six characters identifying the catalog on the // LBL job control statement must be YCAT, and the system catalog is always named YCAT. If you are performing a JC$CAT operation on more than one catalog (copies), you must also specify their device assignment sets.

4. This job control statement executes the JC$CAT utility.

5. This indicates the start of embedded data, which, in this case, is the utility control statements.

6. The FIL control statement is always required when executing JC$CAT.

7. One or more utility control statements are placed here.

8. This is the end-of-data job control statement.

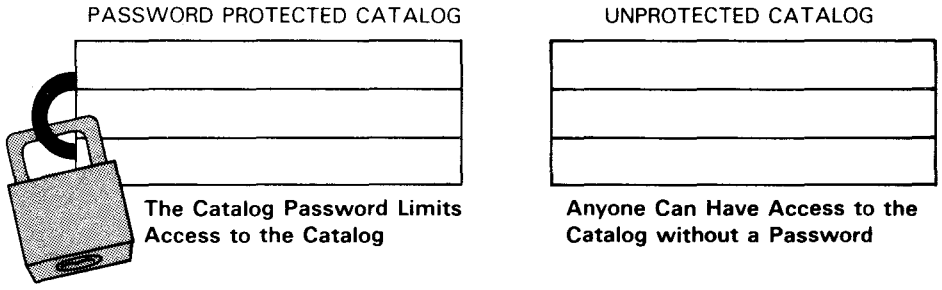
9. This job control statement signals the end of the job.

For a more detailed description of the job control statements used in this manual, see the job control user guide, UP-8865 (current version).

6. Protecting the Catalog with a Password

6.1. DETERMINING ITS USE

If you are a system administrator you must determine if your catalog will be used as a security or convenience facility. If you decide to use the catalog primarily as a convenience facility and don't want to restrict a person's ability to catalog and decatalog files or access the catalog through JC$CAT, you have no need to assign a catalog password. However, if you are going to use the catalog as a security facility and want to protect it against deliberate or accidental misuse, you can do so by establishing a catalog password.



Catalog password function

Establishing a catalog password gives you control over the catalog by ensuring that only you or people authorized by you will have access to the catalog: Only you or people knowing the catalog password will be able to catalog and decatalog files, print the contents of the catalog, or make copies of the catalog.

6.2. ASSIGNING A CATALOG PASSWORD

Use JC$CAT, FIL, and CHP

To establish a catalog password, you execute JC$CAT in a job control stream that includes the utility control statements FIL and CHP. We discussed JC$CAT and the FIL control statement in Section 5. You may recall that the FIL control statement identifies the catalog to JC$CAT and is always required when executing JC$CAT. To review the format and parameters of the FIL control statement, see 5.2.

CHP function

The FIL control statement is followed by the CHP control statement. The CHP control statement identifies the password that you want assigned to the catalog, and it associates the logical catalog file name (lcfn) of the FIL control statement with the catalog password.

CHP format

```
CHP Dn,{new-password }
       {NOPASSWORD  }
```

Dn and password parameters

The password consists of one to six alphanumeric characters, while the Dn parameter is the logical catalog file name that identifies the catalog.

Required parameters for assigning password

To assign a catalog password, you need the following parameters:

```
CHP Dn,new-password
```

Example: Assigning catalog password

The following job control stream executes JC$CAT to assign a catalog password:

```
// JOB CATPSWD
// DVC 20 // LBL PRNTR
// DVC RES // LBL $Y$CAT // LFD CATFIL
// EXEC JC$CAT
/$
       FIL  D0=CATFIL
       CHP  D0,catpwd
/*
/&
```

This job stream assigns a catalog password to the system catalog. The first device assignment set is for the printer. The second device assignment set identifies the system catalog, // LBL YCAT, which is located in the system resident volume, // DVC RES. In this example, the logical file name for our catalog is CATFIL, // LFD CATFIL. The logical catalog file name D0 is shown in the first parameter of the FIL control statement; it specifies the system catalog file. Notice that the logical file name in the second parameter of the FIL statement (CATFIL) is the name from the // LFD job control statement of the catalog device assignment set. The password we are assigning to the catalog is specified in the second parameter of the CHP control statement, catpwd.

NOTE:

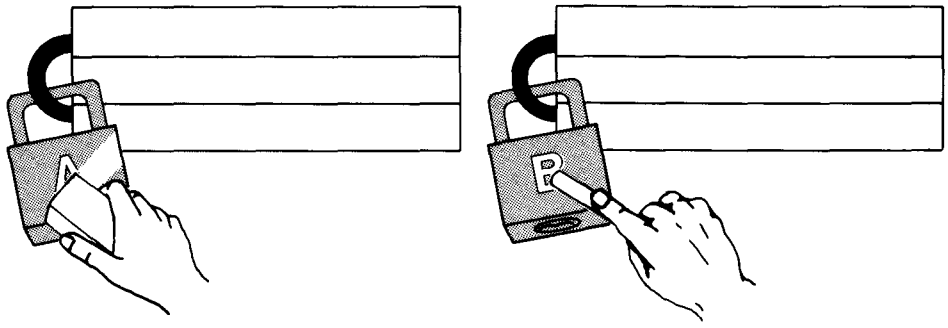
Password for disk only

A catalog password can be established or changed for a disk file only.

6.3. CHANGING OR VOIDING THE CATALOG PASSWORD

Changing catalog password

If you want to change your catalog password, identify the existing password on the FIL control statement, the new password you want to assign on the CHP control statement, and execute JC$CAT. The new password replaces the old one in the catalog.



Example: Changing catalog password

The following is a typical job control stream used to execute JC$CAT for the purpose of changing a catalog password:

```
// JOB CHGPSWD
// DVC 20    // LFD PRNTR
// DVC RES   // LBL $Y$CAT   // LFD CATFIL
// EXEC JC$CAT
/$
      FIL  D0=CATFIL/old-password
      CHP  D0,new-password
/*
/&
```

Voiding catalog password (NOPASSWORD)

If you want to remove an existing catalog password (without replacing it with a new one), use the FIL control statement identifying the catalog and its existing password, and follow this control statement with the CHP control statement using the NOPASSWORD parameter.

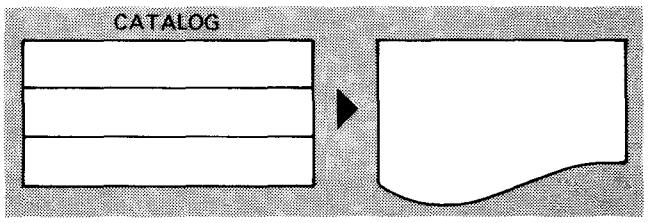
7. Other Functions of JC$CAT

7.1. INTRODUCTION

Remaining JC$CAT operations

We have discussed how the catalog manipulation utility JC$CAT is used to assign, change, and void a catalog password (Section 6), and how it is used to add, change, and void the read/write passwords for individual cataloged files (Section 2). In this section, we complete our discussion of JC$CAT by explaining how you use it to print the catalog, dump (copy) the catalog to tape or disk, restore the catalog to the system resident volume, and change or void passwords on multiple copies of the catalog.

7.2. PRINTING THE CATALOG



What is listed

JC$CAT can be used to print the catalog. The listing produced by the utility includes the file identification, LFD name, device type, number of volumes, vsn, number of extents, and volume sequence of each cataloged file. In addition, for generation files, current generation number, number of generations in the file, and number of generations retained are provided. (See Figure 7–1.)

Read/write passwords not listed

What is not listed in the printed output is the read/write passwords protecting the individual files.

Disk only

Only a catalog residing on disk can be printed.

Figure 7–1 is an example of a printed catalog.

```
*.*.*.*.*.*.*.*.*
1       C       1
*               *
1               1
*.*.*.*.*.*.*.*.*

TYPE = GEN.FILE
CURRENT GEN.NO.=20
NO. OF GENS. IN FILE=04
NO. of GENS TO KEEP=04 ** FILES SHOWN IN ORDER OF OLDEST TO NEWEST **
* * * C U R R E N T   G E N E R A T I O N - 0 3 * * *
    FILE IDENTIFICATION=C17
    LFD NAME=GF164
    DEV.  TYPE=8418 VOL-1 VSN=REL060 NO.VOLS=1 NO.EXTENTS=8 VOL SEQ.=1

* * * C U R R E N T   G E N E R A T I O N  -  0 2  * * *
    FILE IDENTIFICATION = C18
    LFD NAME = GF164
    DEV. TYPE=8418 VOL-1 VSN=REL060 NO.VOLS=1 NO.EXTENTS=8 VOL SEQ.=1
* * * C U R R E N T   G E N E R A T I O N  -  0 1  * * *
    FILE IDENTIFICATION=C19
    LFD NAME=GF164
    DEV. TYPE=8418 VOL-1 VSN=REL060 NO.VOLS=1 NO.EXTENTS=8 VOL SEQ.=1
* * * C U R R E N T   G E N E R A T I O N  -  0 0 * * *
    FILE IDENTIFICATION=C20
    LFD NAME=GF164
    DEV. TYPE=8418 VOL-1 VSN=REL060 NO.VOLS=1 NO.EXTENTS=8 VOL SEQ.=1
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*

*.*.*.*.*.*.*.*.*
1               1
*       B       1
1               1
*.*.*.*.*.*.*.*.*

TYPE=GEN.FILE
CURRENT GEN. NO.=02
NO. OF GENS.  IN FILE=03
NO. OF GENS TO KEEP=04 ** FILES SHOWN IN ORDER OF OLDEST TO NEWEST **
* * * C U R R E N T   G E N E R A T I O N  -  0 2  * * *
    FILE IDENTIFICATION=B00
    LFD NAME=GF164
    DEV.  TYPE=8418 VOL-1 VSN=REL052 NO.VOLS=1 NO.EXTENTS= VOL SEQ.=1
```

Figure 7–1. Sample Printout of the Catalog (Part 1 of 2)

```
* * *  C U R R E N T   G E N E R A T I O N  -  Ø 1   * * *
    FILE IDENTIFICATION=BØ1
    LFD NAME=GF164
    DEV.  TYPE=8418 VOL-1 VSN=RELØ52 NO.VOLS=1 NO.EXTENTS=8 VOL SEQ.=1
* * *  C U R R E N T   G E N E R A T I O N  -  Ø Ø   * * *
    FILE IDENTIFICATION=BØ2
    LFD NAME=GF164
    DEV TYPE=8418 VOL-1 VSN=RELØ52 NO.VOLS=1 NO.EXTENTS=8 VOL SEQ.=1
*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*.*
*.*.*.*.*.*.*.*.*
1               1
*       A       *
1               1
*.*.*.*.*.*.*.*.*
TYPE=GEN.FILE
CURRENT GEN. NO.=ØØ
NO. OF GENS. IN FILE=Ø1
NO. OF GENS TO KEEP=Ø4 ** FILES SHOWN IN ORDER OF OLDEST TO NEWEST **
```

Figure 7–1. Sample Printout of the Catalog (Part 2 of 2)

DSP function

To obtain a printed listing of the catalog, use the DSP control statement.

DSP format

```
DSP[.options] Dn[,file-id] [,{(nn) }]
                              {(-nn)}
```

Dn parameter

The only parameter required in this control statement is the 2-character logical catalog file name (D0...D8) that identifies to JC$CAT the catalog to be printed. If only the required parameters are used, JC$CAT will print all the file entries in the catalog.

FIL precedes DSP

In the statement sequence, the DSP control statement must follow the FIL control statement.

Printing all entries

The following job stream is used to execute JC$CAT for the purpose of listing all the entries in the system catalog. Note the YCAT on the LBL job control statement, the printer device assignment set, and the FIL control statement. These are required when using JC$CAT and are discussed in 5.2.

Example: Printing entire catalog

```
// JOB PRTCAT
// DVC 20   // LFD PRNTR
// DVC RES  // LBL $Y$CAT   // LFD CATFIL
// EXEC JC$CAT
/$
   FIL  D0=CATFIL/PASWD1
   DSP  D0
/*
/&
```

Options parameter for generation files

There are two options available for the option parameter in the DSP control statement. Both options are used for obtaining a list of generation files. The two options for generation files are:

C Lists the current generation of all cataloged generation files

G Lists all members of cataloged generation files

Examples: Printing current generation or entire generation file

`DSP.C D0` Lists the current generation of generation files in the catalog. Only generation files are processed when this option is specified.

`DSP.G D0` Lists all members of generation files in the catalog. Only generation files are processed when this option is specified.

Other DSP parameters for specific files

The remaining parameters in the DSP control statement are the file-id and (nn) parameters. You can use the file-id parameter by itself to print a specific file entry, or you may use it in combination with the (nn) and (–nn) parameter to define the printing of a specific generation file.

Examples: Printing specific files

```
DSP D0,PAYROLL
```

Prints the specific file, PAYROLL.

```
DSP D0,PAYROLL.ENGINEERING,(03)
```

Prints generation 03 for the PAYROLL.ENGINEERING file.

```
DSP D0,PAYROLL.ENGINEERING,(-03)
```

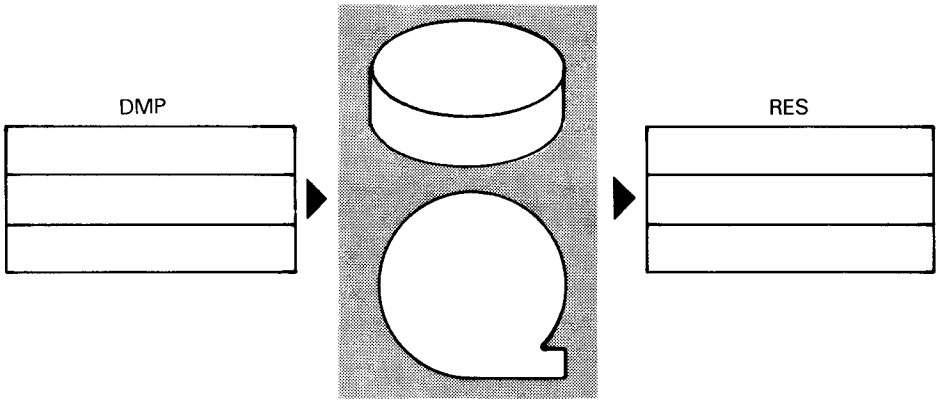
Prints the third generation that precedes the current generation of the PAYROLL.ENGINEERING file.

7.3. DUMPING/RESTORING THE CATALOG

Purpose of catalog copy

The catalog can be dumped from SYSRES to a backup magnetic tape or disk file, and the backup tape or disk file can be restored to YCAT on SYSRES. It is a good idea to copy your catalog so that:

- you have a backup tape or disk; and
- when you need to perform an initial program load (IPL) for a new SYSRES pack, you can easily restore your copied catalog file onto the new SYSRES volume.



The utility that performs the dumping and restoring functions consists of two separate areas: the DMP control statement that performs dumping, and the RES control statement that performs the restoring functions. When the catalog is either dumped or restored, no printed output is produced, but you can create a catalog listing by placing the DSP control statement for printing in the statement sequence.

DMP function

The DMP control statement copies (dumps) the system catalog residing on SYSRES to a tape or disk file. The dump control statement identifies the device type on which the catalog resides and the device type to which the catalog is written. The logical catalog file name (Dn or Tn) is used in place of the LFD name that it represents.

DMP format

```
DMP  {Dn,Dn}
     {Dn,Tn}
```

DMP parameters

The first parameter (Dn) is the input catalog identifier, and the second parameter (Dn or Tn) is the output identifier.

Examples: Copying the catalog from disk to tape, disk to disk

```
DMP D0,T0
```

This will dump the catalog on SYSRES (D0) to a tape file (T0).

```
DMP D0,D1
DSP D1
```

The first statement will dump the catalog from SYSRES to another disk. The second statement, which can precede the first statement, will give a printed output of the dump. Remember, you can print a catalog from disk only.

Examples: Making a catalog copy

In the following job stream, a password-protected catalog on SYSRES is dumped to a tape file. Remember, the copy on tape will not be password protected (6.2).

```
// JOB CATDMP
// DVC 20    // LFD PRNTR
// DVC RES   // LBL $Y$CAT   // LFD CATFIL
// DVC 90    // VOL TAPE01   // LBL $Y$CATBKUP
// LFD CATCPY
// EXEC JC$CAT
/$
      FIL  D0=CATFIL/PSWD1,T0=CATCPY
      DMP  D0,T0
/*
/&
```

RES function

The RES control statement copies (restores) the backup disk or tape file to the catalog file on SYSRES.

RES format

```
RES  {Dn,Dn}
     {Tn,Dn}
```

RES parameters

You need to use both parameters of the RES control statement. The first parameter is the device type code for the input catalog (catalog copy), and the second parameter is the device type code for the output file (YCAT on SYSRES).

Example: Restoring the catalog file with printed output

In the following job stream, a catalog on tape used as a backup is restored to a new SYSRES pack with password protection. Also, a printout of the catalog on SYSRES is obtained:

```
// JOB RESTORE
// DVC 20   // LFD PRNTR
// DVC 90   // VOL TAPE01   // LBL $Y$CATBKUP   // LFD CATCPY
// DVC RES  // LBL $Y$CAT   // LFD CATFIL
// EXEC JC$CAT
/$
      FIL  T0=CATCPY,D1=CATFIL
      CHP  D1,PSWD1
      RES  T0,D1
      DSP  D1
/*
/&
```

Include all necessary passwords

If the catalog and its copy were both on disk and password protected, (remember, a password can only be established and changed for a disk file), you would include both the passwords in the FIL control statement. For example, if the backup catalog on disk is restored to YCAT on SYSRES, the statement sequence might look like this:

```
FIL  D1=CATCPY/PSWD2,D0=CATFIL/PSWD1
RES  D1,D0
```

Password checking during DMP and RES

Notice that both the passwords are specified on the FIL control statement. Password checking occurs on output files in both the dump and restore functions. Before an existing catalog is overwritten with its copy, the password on the existing catalog is checked. If there is no match, a copy is not made. If there is a match, the entire output file is overwritten by the input file; the output file password is also overwritten. In the previous example, the restored catalog will be protected by PSWD2.

7.4. CHANGING/VOIDING PASSWORDS ON MULTIPLE COPIES OF THE CATALOG

CAT utility control statement

You can change or void passwords on multiple copies of the catalog by using the same procedure as for a single catalog (6.3). However, for your convenience, the catalog manipulation utility provides a control statement that is especially helpful when performing any function on multiple copies of the catalog. This statement is the CAT control statement, which should not be confused with the CAT job control statement. The CAT control statement is used when the catalog and its copies share a common password. The CAT control statement saves the time required to specify a catalog password for each copy of the catalog.

CAT function

CAT format

```
CAT  PASSWORD=password-name
```

Example: Using CAT to change shared passwords

The following is a typical example of a job control stream used to execute JC$CAT for changing passwords on multiple copies of the catalog:

```
// JOB CHGPSWDS
// DVC 20    // LFD PRNTR
// DVC RES   // LBL $Y$CAT  // LFD CATFIL
// DVC 60    // VOL DISK01  // LBL $Y$CAT  // LFD CATCPY
// DVC 60    // VOL DISK02  // LBL $Y$CAT  // LFD CATBUP
// EXEC JC$CAT
/$
      FIL  D0=CATFIL,D1=CATCPY,D2=CATBUP
      CAT  PASSWORD=KATLOG
      CHP  D0=PSWD1
      CHP  D1=PSWD2
      CHP  D2=PSWD3
/*
/&
```

Procedure for using CAT

If you decide to use the CAT control statement you must:

- declare the catalog and its copies (FIL); then
- specify the common password (CAT); and then
- specify the new password. (A separate CHP control statement is needed for each copy of the catalog.)

If, however, a copy of the catalog has its own individual password, then that password must be specified on the FIL control statement.

Voiding common passwords

When voiding a common password for multiple copies of the catalog, you can use the CAT control statement to specify that common password, and the NOPASSWORD parameter is used on the CHP control statements.

Importance of FIL and CAT sequence

One final point to keep in mind: the order of the FIL and CAT control statements in the statement sequence is important.

- If the CAT control statement is encountered after a FIL control statement, the password on the CAT control statement replaces any previously stated password on the FIL control statement.
- If the CAT control statement precedes the FIL control statement, the catalog password on the CAT control statement applies to only those FIL control statements without an explicit password.

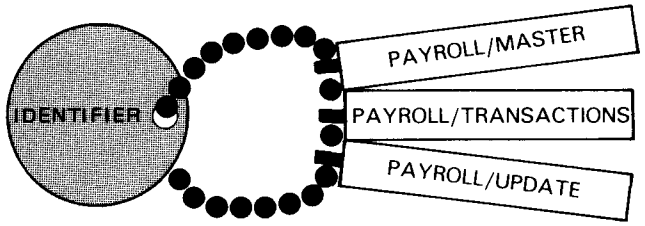
Example: FIL and CAT sequence

In the following statement sequence, you can see how the rules for password checking apply:

`FIL DØ=CATFIL/PSWDY` `CAT PASSWORD=PSWDX`	X applies because the CAT statement follows the FIL statement.
`FIL D1=CATCPY`	X applies because the CAT statement precedes the FIL statement, and the FIL statement contains no explicit password.
`FIL D2=CATBUP/PSWDZ`	Z applies because the CAT statement precedes the FIL statement and this FIL statement does contain a password.

8. Cataloging Related Files

8.1. USING QUALIFIERS TO GROUP RELATED FILES



Function of qualifiers

In this chapter we discuss an optional way to catalog files. It is based on the same procedures we discussed in Sections 2 and 3, but it enables you to group or link logically related files by means of a file label prefix called a qualifier. A qualifier indicates that a file is related, in some way, to other files having the same qualifier. You may use qualifiers to group files in any relationship you find is valid, such as subject, author, date of creation, and so on.

Advantages of qualifiers

These are the advantages of using qualifiers:

1. Qualifiers help you to quickly identify files with similar functions or uses.

2. When you want to print the catalog, you can print all the files specified by a qualifier. For example, if you want to list all the files with the qualifier PAYROLL, you specify the following in your job stream:

```
DSP D0,PAYROLL/
```

Printing the contents of the catalog is discussed in 7.2.

3 Files with the same qualifier can be decataloged in groups rather than one file at a time. (This is discussed in 8.3.)

8.2. JOB CONTROL STATEMENTS USED TO GROUP FILES

Assigning qualifiers with LBL or QUAL job control statements

Files are assigned qualifiers through a special form of the LBL job control statement, or by the QUAL job control statement.

Minimum LBL format

First, the LBL job control statement. So far, we have seen the format of the LBL job control statement as:

```
// LBL file identifier
```

To assign a qualifier, an expanded format of the LBL job control statement is used. It is shown below:

Special LBL format for assigning qualifier

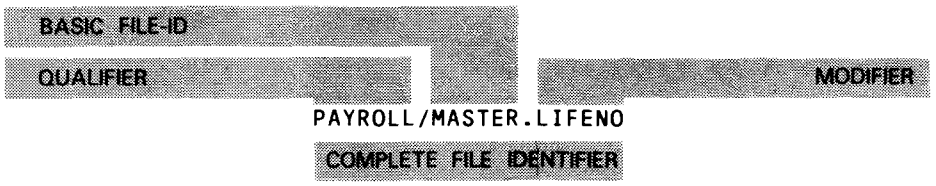
```
// LBL [qualifier/]basic file-id[.modifier-1...[.modifier-n]]
```

Examples: Files with a qualifier

Keeping this format in mind, suppose that you have three payroll files (master, transactions, and update.) These can be grouped together under the basic qualifier PAYROLL as follows: PAYROLL/MASTER, PAYROLL/TRANSACTIONS, PAYROLL/UPDATE. Here, MASTER, TRANSACTIONS, and UPDATE are the basic file identifiers. Though they are grouped together by a common qualifier, they remain three separate files.

Modifiers

You can also expand the basic file identifier by appending a modifier to it. An example is: MASTER.LIFENO, where LIFENO is the modifier.



In summary, you can identify a file by simply using the basic file-id, or you can expand the description of the file by using the basic file-id affixed with a qualifier and/or modifier.

Parameter restrictions

The total number of characters for file identification may be up to 17 for tape and diskette files and 44 for disk files (including the slash and periods). As you can see by the examples, you code a slash after the qualifier and use periods to separate modifiers. (Up to 21 modifiers are allowed.)

Assigning a qualifier with LBL

To catalog a group file with its qualifier, you simply include the complete file identifier in the LBL job control statement and follow the usual procedure for cataloging a file (2.1).

Example: Assigning a qualifier

Figure 8–1 shows a job control stream used to catalog a group file with its qualifier via the LBL job control statement.

```
// JOB CATALOG
// DVC 20  // LFD PRNTR
// DVC RES
// LBL PAYROLL/MASTER.LIFENO
// LFD INPUTA
// CAT INPUTA
/&
```

Figure 8–1. Job Control Stream Using the LBL Job Control Statement to Assign a Qualifier

QUAL function

The second method of using the qualifier to group related files involves the QUAL job control statement. The major function of the QUAL statement is to assign a qualifier to the file identifiers of every file being cataloged within your job stream. This makes it unnecessary to include the qualifier on the LBL job control statements associated with each file. Thus, the QUAL job control statement is particularly useful when you want to assign a qualifier common to several files being cataloged in the same job stream.

QUAL format

```
//[symbol] QUAL [qualifier]
```

QUAL parameters

The only parameter of this statement is the qualifier, which is a 1- to 8-character alphanumeric name. A slash (qualifier/) is automatically added to the parameter when specified, so there is no need for you to include it as part of your specification. The qualifier does not apply to a LBL job control statement with its own unique qualifier.

QUAL restrictions

NOTE:

The QUAL job control statement applies only to the job stream in which it is found.

Using QUAL

Similar to a job control stream used to catalog a file without a qualifier, the job control stream using the QUAL statement must include the device assignment set and the CAT job control statement.

Example: Using QUAL to assign a qualifier

Figure 8–2 shows a job control stream used to catalog a file identified with the qualifier, PAYROLL.

```
// JOB CATALOG
// QUAL PAYROLL
// DVC 6Ø  // VOL DISKØ1
// LBL MASTER
// LFD CATFIL
// CAT CATFIL
/&
```

New file identifier
PAYROLL/MASTER

Figure 8–2. Job Control Stream Using the QUAL Job Control Statement to Assign a Qualifier

Referencing a file having a qualifier

Remember, after you have cataloged a file using the QUAL control statement, it will be referenced by its complete file identifier on the LBL control statement.

Example: Referencing a file with a qualifier

The following job control stream is used to reference the file that was cataloged in Figure 8–2. Note the LBL statement.

```
// JOB REFER
// DVC 2Ø  // LFD PRNTR
// LBL PAYROLL/MASTER
// EXEC PROGA
/&
```

Example: Using QUAL to assign a qualifier to multiple files

The following job control stream is used to catalog two files identified with the qualifier, PAYROLL:

```
// JOB CATALOG
// QUAL PAYROLL
// DVC 60  // VOL DISK01
// LBL TRANSACTIONS.LIFENO
// LFD CATFIL
// CAT CATFIL              (New file-id: PAYROLL/TRANSACTIONS.LIFENO)
// DVC RES
// LBL UPDATE.LIFENO
// LFD INPUTA
// CAT INPUTA                (New file-id: PAYROLL/UPDATE.LIFENO)
```

Voiding qualifiers

In addition to assigning a qualifier to the file identifier, the QUAL job control statement can be used to void a qualifier. To void a qualifier, include the QUAL job control statement in your job control stream without specifying an operand.

Example: Voiding a qualifier

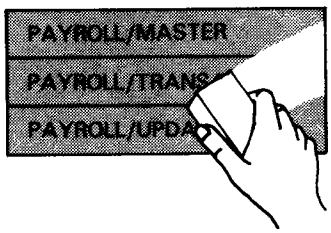
The following job stream removes the qualifier that was assigned to the file in Figure 8–2:

```
// JOB QUALVOID
// QUAL
// DVC 60   // VOL DISK01
// LBL MASTER
// LFD CATFIL
// CAT CATFIL
```

Old file-id: PAYROLL/MASTER
New file-id: MASTER

8.3. DECATALOGING GROUP FILES

The procedure used to decatalog files having the same qualifier takes advantage of an important benefit offered by qualifiers – easy removal of groups of file entries from the catalog. This is because all files with the same qualifier can be decataloged with one job stream.



Job stream requirements

To decatalog a set of files with the same qualifier, your job control stream must include:

- the qualifier (/) on the LBL job control statement; and,
- a full device assignment set since the LBL job control statement references an incomplete file identifier (PAYROLL/).

Example: Decataloging group files

The following job stream is used to decatalog a group of files with the same qualifier:

```
// JOB REMOVE
// DVC 6Ø
// VOL DISKØ1
// LBL PAYROLL/
// LFD INPUTA
// DECAT INPUTA,PSWD1
/&
```

In the above job stream, all the files with the qualifier PAYROLL are decataloged, provided they are located on DVC 60, VOL DISK01.

Restrictions

NOTE:

You can decatalog files in groups, but they cannot be scratched in groups; that is, each file with a qualifier must be scratched individually, specifying its complete file-id on the LBL job control statement.

Appendix A. Statement Summary

The job control statements for file cataloging and the control statements of the catalog manipulation utility (JC$CAT) are listed and described in Table A–1 and Table A–2, respectively.

Table A–1. Job Control Statements Applicable to File Cataloging

Control Statements	Description
//[symbol] CAT lfdname[,catpw][,SCR] [,{GEN=nn / MEM}]	Catalogs a file
//[symbol] DECAT lfdname[,catpw][,SCR] [,{GEN / ROL}]	Removes a file entry from the catalog
//[symbol] LBL [qualifier/]basic file-id [.modifier-1...[.modifier-n]] [{+n / -n / nn}] [(rpw/wpw)]	Identifies the file
//[symbol] QUAL [qualifier]	Assigns a qualifier to file identifiers in a job

Table A–2. JC$CAT Control Statements

Control Statements	Description
`CAT PASSWORD=password-name`	Specifies a common password for multiple copies of the catalog file
`CFP Dn,file-id {[,RDOLD=old-password],RDNEW={new-password | NOPASSWORD} | [,WROLD=old-password],WRNEW={new-password | NOPASSWORD}}`	Changes a read and/or write password on a cataloged file
`CHP Dn,{new-password | NOPASSWORD}`	Assigns, changes, and voids the catalog file password
`DMP {Dn,Dn | Dn,Tn}`	Dumps the catalog ifle to another disk or tape file
`DSP[.options] Dn[,file-id] [,{(nn) | (-nn)}]`	Prints the contents of the catalog file
`FIL {Dn | Tn}=filename-1[/password-1] [...,` `{Dn | Tn}=filename-n[/password-n]]`	Identifies the catalog file to JC$CAT
`RES {Dn,Dn | Tn,Dn}`	Restores the catalog file to SYSRES

Appendix B. Job Control Options to be Respecified when Processing Cataloged Files

The catalog file (YCAT) contains the device job assignment sets for your files. Some parameters of these job control statements are not remembered by the catalog file when you later reference the file. These parameters, if needed, must be specified in subsequent job control streams. The following job control statement parameters must be respecified.

Table B–1. Job Control Options to be Respecified

Job Control Statement	Parameter
DD	No parameters are cataloged.
DVC	IGNORE
EXT	No parameters are cataloged.
LFD	EXTEND INIT RELOAD PREP
VOL	PREP

Appendix C. Statement Conventions

The conventions used to present job control statements are:

The // job control statement

- The // for a job control statement must be the first characters on the statement record. (Columns 1 and 2 are normally used; however, they may start later in the record.) At least one blank column must separate the // from the operation code. For example:

```
    BLANK
      ▼
// LBL...
```

- At least one blank column must separate the operation code and the first parameter. For example:

```
      BLANK
        ▼
// LBL [qual/]
```

Using the comma

- When more than one parameter is used, a comma must be used to separate the parameters, with no intervening blanks. Everything after the first blank is considered to be a comment. For example:

```
                    COMMA
                      ▼
//[symbol] CAT lfdname,catpw
```

Continuation rules

- A continuation line is not considered to be a job control statement in itself, but rather an extension of a job control statement in a preceding record. The continuation rules are:

  1. Code a comma after the last parameter used on a record that is going to be continued (but do not place a comma after the very last parameter used in the statement). The comma, followed by a blank, indicates that there is a continuation of this statement.

  2. Begin the record containing the continued characters with a // as the first characters and leave at least one blank between the // and the parameter starting the continuation.

Lowercase letters and words

- Lowercase letters and words are generic terms representing a value that you must supply. For example:

```
[,catpw]
```

Capital letters

- Capital letters must be coded exactly as shown. For example:

```
[,SCR]
```

Braces

- Information contained within braces represents mandatory entries of which one must be chosen. For example:

$$\left\{\begin{matrix}Dn,Dn\\Dn,Tn\end{matrix}\right\}$$

Brackets

- Information contained within brackets represents optional entries that (depending upon program requirements) are included or omitted. Braces within brackets signify that one of the entries must be used if that bracketed parameter is to be used. For example:

```
[(rpw/wpw)]
```

$$\left[\left\{\begin{matrix}+n\\-n\\nn\end{matrix}\right\}\right]$$

Index

CUT

SPERRY

USER COMMENT SHEET

We will use your comments to improve subsequent editions.

NOTE: Please do not use this form as an order blank.

_____

(Document Title)

_____ _____ _____

(Document No.) *(Revision No.)* *(Update No.)*

Comments:

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

CUT

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 ADDRESSEE

SPERRY CORPORATION

ATTN.: SOFTWARE SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

FOLD

UNISYS

USER COMMENTS

We will use your comments to improve subsequent editions.

NOTE: Please do not use this form as an order blank.

_____

(Document Title)

_____ _____ _____

(Document No.) *(Revision No.)* *(Update Level)*

Comments:

From:

_____

(Name of User)

_____

(Business Address)

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

CUT

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 ADDRESSEE

Unisys Corporation
E/MSG Product Information Development
PO Box 500 C1-NE6
Blue Bell, PA 19422-9990

FOLD