Guide to VAX DEC/Code Management System

Order Number: AA–KL03D–TE

December 1989

This manual describes the concepts, commands, and features of the VAX DEC/Code Management System.

Revision/Update Information:	This document supersedes the *Guide to VAX DEC/Code Management System* (Order No. AA–KL03C–TE).
Operating System and Version:	VMS Version 5.0 or higher; VMS Version 5.3 or higher for the CMS DECwindows Interface.
Software Version:	VAX DEC/CMS Version 3.3

digital equipment corporation
maynard, massachusetts

First Printing, April 1987
Revised, April 1988
Revised, May 1989
Revised, December 1989

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

ZK5321

Contents

Preface

The VAX DEC/Code Management System (CMS) is an online library system that helps track software development and maintenance. This manual provides reference and conceptual information on how to use CMS on the VMS operating system.

Intended Audience

This guide is intended for all users of CMS, including managers, project programmers, writers, and others who may be responsible for maintaining CMS libraries.

This guide can be used by both experienced and novice users of CMS. You need not have a detailed understanding of the VMS operating system; however, some familiarity with the conventions of the Digital Command Language (DCL) is helpful.

Document Structure

This guide has two parts. The first part is task-oriented and consists of Chapters 1 through 10; it explains how to use CMS, gives information on CMS concepts, and explains syntax rules. The second part is a reference section; it contains the Command Dictionary and four appendixes.

- Chapter 1, Introduction to CMS, describes the basic concepts of CMS and presents a tutorial example to help you get started.
- Chapter 2, Using CMS with DECwindows, describes how to use the CMS DECwindows user interface.
- Chapter 3, Libraries, describes how to set up a CMS library and how to use library search lists.

- Chapter 4, Elements and Generations, explains the concepts of files in a CMS library.
- Chapter 5, Groups and Classes, explains how to organize files into groups and classes.
- Chapter 6, Variants and Merging, describes lines of descent, creating variant lines of descent, and how to merge files.
- Chapter 7, Security Features, describes the protection mechanisms that you can use in CMS.
- Chapter 8, Event Handling and Notification, describes how CMS handles events and the concept of notification when these events occur.
- Chapter 9, Library Maintenance, describes how you can maintain the validity and integrity of your CMS library.
- Chapter 10, Command Syntax, gives detailed information on CMS syntax and how to specify commands.
- The Command Dictionary contains detailed descriptions of each CMS command. The commands are listed in alphabetical order with the command name at the top of every page.
- Appendix A, Summary of CMS Interface Functional Mappings, provides a table displaying how each of the CMS interfaces are functionally mapped to each other.
- Appendix B, Error Messages, lists CMS diagnostic messages and includes a brief description of each message.
- Appendix C, CMS Library Storage Method, contains information on how libraries are stored.
- Appendix D, System Management Considerations, contains information about running CMS on the VMS operating system.

Associated Documents

- The *VAX DEC/Code Management System Callable Routines Reference Manual* contains information about using the CMS callable routines.
- The *VAX DEC/Code Management System Installation Guide* supplies the instructions for installing CMS on a VMS system.
- The *Using VAXset* manual provides information on using VAX Software Engineering Tools with other VMS facilities to extend programmer productivity.

onventions

The following conventions are used in this guide:

Conventions	Description
DELETE	A key name is shown enclosed to indicate that you press a key on the keyboard.
Ctrl/x	A sequence such as CTRL/x indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
KPn	A sequence such as KP1 indicates that you must first press and release the key labeled KP1, then press and release another key or a pointing device button.
. . .	In examples, a horizontal ellipsis indicates one of the following: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
( )	In format descriptions, parentheses indicate that, if you choose more than one option, you must enclose the choices in parentheses.
[ ]	In format descriptions, brackets indicate that whatever is enclosed is optional; you can select none, one, or all of the choices.
{}	In format descriptions, braces surround a required choice of options; you must choose one of the options listed.
user input	The hardcopy version of this manual has interactive examples that show user input in red letters and system responses or prompts in black letters. The online version differentiates user input from system responses or prompts by using a different font.
boldface text	Boldface words introduce attributes.

Conventions	**Description**
italic text	Italicized words introduce new terms.
UPPERCASE TEXT	Uppercase letters indicate the name of a command or routine. Lowercase words and letters used in examples indicate that you should substitute a word or value of your choice.
mouse	The term *mouse* is used to refer to any pointing device, such as a mouse, a puck, or a stylus.
MB1, MB2, MB3	MB1 indicates the left mouse button, MB2 indicates the middle mouse button, and MB3 indicates the right mouse button. (The buttons can be redefined by the user.)

Unless otherwise noted, all numeric values are represented in decimal notation.

Unless otherwise noted, you terminate a command by pressing the RETURN key.

Chapter 1

Introduction to CMS

The VAX DEC/Code Management System (CMS) is a library system for software development and maintenance. CMS stores files called elements in an online library, keeps track of changes made to these files, and monitors user access to the files.

This chapter provides the following information:

- An overview of CMS and information on how to get started
- An introduction to CMS concepts
- A sample CMS session
- A summary of CMS commands

1.1 Overview

During software development, programmers continually make changes to project files. CMS stores and monitors these files.

CMS allows you to store project files in a central library where they are available to all project members. Some of the tasks you can perform on these files are as follows:

- Store files (called elements) in a library
- Fetch elements, modify them, and test them in your own directory
- Control concurrent modifications to the same element
- Merge concurrent modifications to an element
- Create successive versions (called generations) of elements
- Compare two generations of an element within a library
- Organize related library elements into groups

- Define a set of generations of elements as a class to make up a base level or release version of a project
- Track which users are working on which elements from the library
- Maintain a historical account of element and library transactions

1.2 CMS Concepts

This section introduces basic CMS concepts.

1.2.1 Libraries, Elements, and Generations

CMS stores all the information it needs in a library. A CMS *library* is a VMS directory containing specially formatted files. It serves as a container or repository for various CMS entities (called *objects*).

An *element* is the basic structural unit in a CMS library; it consists of one file and all its versions. An element *generation* represents a specific version of that element. When you create an element and place it in a CMS library for the first time, CMS creates generation 1 of that element. Each time you reserve and then replace a generation of an element in the library, CMS creates a new generation of that element.

For information on libraries, see Chapter 3. For information on elements and generations, see Chapter 4.

1.2.2 Groups and Classes

A *group* is a set of elements (or other groups) that you can combine and manipulate as a unit. For example, you might create a group containing all the elements that process error messages.

A *class* is a set of particular generations of elements. You typically combine generations of elements into classes to represent progressive stages, or base levels, in the development of an entire system.

For information on groups and classes, see Chapter 5.

1.2.3 Reservations and Replacements

As changes are made to a file in the VMS file system, new versions of that file are created. Similarly, as an element is developed in CMS, new generations of that element are created. In addition to storing the element and its generations, CMS manages the development process by using reservations and replacements.

A *reservation* exists in the CMS library when you retrieve an element generation with the intent to modify it. The reservation ends and a replacement occurs when you return the modified contents to the library.

For information on reservations and replacements, see Chapter 4.

1.2.4 Review

You can mark an element generation for *review* to indicate that its contents should be reviewed by other users. After the review process is complete, the element generation can be marked as having been accepted or rejected.

For information on marking an element generation for review and the review process, see Section 4.5.4.

1.2.5 History and Remarks

All CMS commands that modify a library or its contents are recorded in the library *history*. You can display any part of the history by using the SHOW HISTORY command. All commands that are recorded allow you to enter a remark, which is recorded in the history along with the command. Remarks are useful in explaining library and element modifications.

For information on library history, see Chapter 4. For information on remarks, see Section 10.2.2.

1.2.6 Reference Copies

For easy reference, you can direct CMS to automatically store copies of the latest main-line generation of selected library elements in a separately designated directory, called a *reference copy directory*.

For information on reference copies, see Sections 3.1.4 and 4.5.3.

1.2.7 Lines of Descent and Variant Generations

The first generation of a newly created element is generation 1. Every time you reserve and replace a generation of that element, CMS numbers a new generation by adding 1 to the number of the reserved generation. This new generation is a descendant of the generation from which it was created. The *main line of descent* consists of generation 1 and its direct descendants.

A generation can have only one direct ancestor and one direct descendant, but it can also have a number of variant descendants. Those generations that are not on the direct line of descent of a generation are called *variant generations*. You specify variant generations by adding a letter, called the variant letter, and the number 1 to the parent generation. For example, generation 2E1 is a variant descendant of generation 2.

A variant generation and its direct descendants (for example, generations 2E1, 2E2, 2E3) form a variant line of descent. A variant generation may have variant descendants; for instance, generation 2E1W1 is a variant descendant of generation 2E1.

For information on lines of descent and variant generations, see Chapter 6.

1.2.8 Concurrent Reservations

You can create variant generations at any time, but default usage creates successive generations along the same line of descent. You must create a variant generation when the direct successor of a reserved generation already exists and you replace a concurrent reservation.

A *concurrent* reservation exists when an element generation has been reserved more than once by one or more users. In this case, only one of these reservations can be replaced on the direct line of descent; the rest of the reservations must be replaced as variant generations.

For information on concurrency, see Section 4.3.

1.2.9 Merging and Conflicts

You can use variant generations to maintain separate but related development of an element, or you may have generations that have undergone concurrent development.

If concurrent changes have been made to a generation, you can *merge* the changes from one line of descent and some variant line of descent into a single generation.

CMS resolves changes from two generations by comparing them to their common ancestor generation. If both generations change a region of their common ancestor in different ways, then this region is known as a conflict. Where the changes do not conflict, CMS includes the appropriate change; where the changes conflict, CMS includes the changes from both generations and flags the conflicting region. In either case, you should verify the resulting merged output for correctness; for instance, program source code should be compiled and executed to ensure that it is syntactically and logically correct. After verifying and making any necessary modifications, you can replace the merged reservation.

For information on merging and conflicts, see Chapter 6.

1.2.10 Security

The VMS operating system provides a security mechanism based on user identification codes (UIC) and access control lists (ACLs) to control access to files within the file system. Similarly, you can use *CMS ACLs* for controlling access to CMS objects through CMS operations. For you to successfully access an object in the CMS library, both the file system and the CMS internal security mechanism must allow you to do so.

For information on the VMS and CMS security mechanisms, see Chapter 7.

1.2.11 Events and Notification

You can use CMS ACLs to specify that a CMS object being accessed constitutes an event, and that some action should be taken when an event occurs. You can specify lists of people to be notified when certain events occur on objects in the CMS library. The default action performed is *notification* through the VMS Mail Utility (MAIL) to one or more users. CMS provides a default notification event handler; in addition, you can write event handlers of your own for CMS to use.

For information on events and notification, see Chapter 8.

1.3 Invoking CMS

You can invoke CMS in four ways:

- From DCL command level
- From CMS subsystem command level

- From a program that calls CMS routines directly (see the *VAX DEC/Code Management System Callable Routines Reference Manual*)
- From the DECwindows user interface

Enter CMS commands at the DCL command level prompt ($) by preceding them with the word CMS. After each command executes, control is returned to DCL level. For example:

```
$  CMS SHOW RESERVATIONS
     .
     .
     .
$
```

You can invoke CMS as a subsystem in the command-line interface in the following ways:

- By entering the CMS command at the DCL prompt
- By entering the CMS command with the /INTERFACE qualifier at the DCL prompt
- By entering the CMS command with the /INTERFACE=CHARACTER_CELL qualifier and keyword at the DCL prompt

For example, you can enter any one of the following commands to enter CMS command-line subsystem mode:

```
$  CMS
CMS>  SHOW RESERVATIONS
 . . .

$  CMS/INTERFACE
CMS>  SHOW RESERVATIONS
 . . .

$  CMS/INTERFACE=CHARACTER_CELL
CMS>  SHOW RESERVATIONS
 . . .
```

For information on entering the CMS DECwindows interface, see Section 2.1.

You should enter the CMS subsystem when you plan on entering a series of CMS commands. This avoids the overhead involved with invoking CMS multiple times.

To terminate the CMS session and return to DCL level, type EXIT or press CTRL/Z.

1.4 Getting Help

You can get information about CMS either at DCL level or at CMS subsystem level. At DCL level, the DCL command HELP CMS provides online help on CMS commands, qualifiers, and other topics. For example:

```
$ HELP CMS
```

To get help on a specific CMS command, such as the CREATE ELEMENT command, type the command after HELP CMS. For example:

```
$ HELP CMS CREATE ELEMENT
```

You can get help at the CMS subsystem level by typing either HELP, or HELP and the specific command. For example:

```
CMS> HELP CREATE ELEMENT
```

To get help from the DECwindows interface, see Section 2.2.

1.5 Sample Session

This section contains a tutorial example showing how to use basic CMS features. The numbers in the example match the explanations at the end of the example.

```
1 Username:  JONES
  Password:

2 $ DIRECTORY

  Directory DISKX:[JONES]

  CMDRMVGRO.BLI;1         CMDRMVGRO.SDML;1     DIFF_DESIGN.MEM;2
  INSTALL-VERSION.TXT;4   INTERNAL_CUST_SITES.COM;6
  LOGIN.COM;71            MAIL_FIL_KEY.COM;6
  NOTES$NOTEBOOK.NOTE;1                       V010-29_INSTALL.TXT;1
  V050-CALLABLE.LOG;2

  Total of 10 files.

3 $ CREATE/DIRECTORY [JONES.CMSLIB]
4 $ CMS

5 CMS> CREATE LIBRARY [JONES.CMSLIB]
  _Remark:  creating new library for my project
  %CMS-S-CREATED, CMS Library DISKX:[JONES.CMSLIB] created
  %CMS-I-LIBIS, library is DISKX:[JONES.CMSLIB]
  %CMS-S-LIBSET, library set
```

```
❻ CMS>  CREATE ELEMENT/KEEP *.*
  _Remark:  creating elements from default directory to new CMS lib
  %CMS-S-CREATED, element DISKX:[JONES.CMSLIB]CMDRMVGRO.BLI created
  %CMS-S-CREATED, element DISKX:[JONES.CMSLIB]CMDRMVGRO.SDML created
  %CMS-S-CREATED, element DISKX:[JONES.CMSLIB]DIFF_DESIGN.MEM created
  %CMS-S-CREATED, element DISKX:[JONES.CMSLIB]INSTALL-VERSION.TXT created
       .
       .
       .
  %CMS-S-CREATED, element DISKX:[JONES.CMSLIB]V010-29_INSTALL.TXT created
  %CMS-S-CREATED, element DISKX:[JONES.CMSLIB]V050-CALLABLE.LOG created
❼ CMS>  EXIT
  $  LOGOUT
       .
       .
       .
❽ $  CMS
❾ CMS>  SET LIBRARY [.CMSLIB]
  %CMS-I-LIBIS, library is DISKX:[JONES.CMSLIB]
  %CMS-S-LIBSET, library set
  -CMS-I-SUPERSEDE, library list superseded

❿ CMS>  SHOW ELEMENT

  Elements in DEC/CMS Library DISKX:[JONES.CMSLIB]

  CMDRMVGRO.BLI         "creating elements from default directory to new CMS lib"
  CMDRMVGRO.SDML        "creating elements from default directory to new CMS lib"
  DIFF_DESIGN.MEM       "creating elements from default directory to new CMS lib"
  INSTALL-VERSION.TXT   "creating elements from default directory to new CMS lib"
       .
       .
       .
  V010-29_INSTALL.TXT   "creating elements from default directory to new CMS lib"
  V050-CALLABLE.LOG     "creating elements from default directory to new CMS lib"

⓫ CMS>  CREATE GROUP
  _Group name:  USER_MANUAL
  _Remark:  creating group for the project user's manual
  %CMS-S-CREATED, group DISKX:[JONES.CMSLIB]USER_MANUAL created

⓬ CMS>  INSERT ELEMENT CMDRMVGRO.BLI,CMDRMVGRO.SDML USER_MANUAL
  _Remark:  inserting the command routine files into group USER_MANUAL
  %CMS-I-INSERTED, element DISKX:[JONES.CMSLIB]CMDRMVGRO.BLI inserted into
  DISKX:[JONES.CMSLIB]group USER_MANUAL
  %CMS-I-INSERTED, DISKX:[JONES.CMSLIB]element CMDRMVGRO.SDML inserted into
  DISKX:[JONES.CMSLIB]group USER_MANUAL
  %CMS-I-INSERTIONS, 2 insertions completed

⓭ CMS>  CREATE ELEMENT CMS$$GSR.TXT/INPUT=DISK$$XXX:[PROJECT.PUBLIC]
  _Remark:  also need the shareable image
  %CMS-S-CREATED, element DISKX:[JONES.CMSLIB]CMS$$GSR.TXT created

⓮ CMS>  INSERT ELEMENT CMS$$GSR.TXT USER_MANUAL
  _Remark:  inserting the shareable image into group USER_MANUAL
  %CMS-I-INSERTED, element DISKX:[JONES.CMSLIB]CMS$$GSR.TXT inserted into
  DISKX:[JONES.CMSLIB]group USER_MANUAL
  %CMS-I-INSERTIONS, 1 insertion completed
```

```
15 CMS>  CREATE CLASS BASELEVEL1
   _Remark:  creating class to contain files needed for base level 1
   %CMS-S-CREATED, class DISKX:[JONES.CMLSIB]BASELEVEL1 created

16 CMS>  RESERVE DIFF_DESIGN.MEM,USER_MANUAL "must add topics to these files"
   %CMS-I-RESERVED, generation 1 of element DISKX:[JONES.CMSLIB]CMDRMVGRO.BLI reserved
   %CMS-I-RESERVED, generation 1 of element DISKX:[JONES.CMSLIB]CMDRMVGRO.SDML reserved
   %CMS-I-RESERVED, generation 1 of element DISKX:[JONES.CMSLIB]CMS$$GSR.TXT reserved
   %CMS-I-RESERVED, generation 1 of element DISKX:[JONES.CMSLIB]DIFF_DESIGN reserved
   %CMS-I-RESERVATIONS, 4 elements reserved

17 CMS>  REPLACE CMS$$GSR.TXT "made two changes to table"
   %CMS-S-GENCREATED, generation 2 of element DISKX:[JONES.CMSLIB]CMS$$GSR.TXT
   created

18 CMS>  SHOW GENERATION

   Element generations in DEC/CMS Library DISKX:[JONES.CMSLIB]

   CMDRMVGRO.BLI      1   23-JAN-1988 17:45:46 JONES "creating elements from
             default directory to new CMS lib"

   CMDRMVGRO.SDML     1   23-JAN-1988 17:46:47 JONES "creating elements from
             default directory to new CMS lib"
   CMS$$GSR.TXT       2   23-JAN-1988 18:12:18 JONES "made two changes to
             table
        .
        .
        .
   V050-CALLABLE.LOG 1   23-JAN-1988 17:49:47 JONES "creating elements from
             default directory to new CMS lib"

19 CMS>  INSERT GENERATION CMS$$GSR.TXT/GEN=1,MAIL_FIL_KEY.COM BASELEVEL1
   _Remark:  these generations needed in class to build baselevel1
   %CMS-S-GENINSERTED, generation 1 of element DISKX:[JONES.CMSLIB]CMS$$GSR.TXT
   inserted into class DISKX:[JONES.CMSLIB]BASELEVEL1
   %CMS-S-GENINSERTED, generation 2 of element DISKX:[JONES.CMSLIB]MAIL_FIL_KEY.COM
   inserted into class DISKX:[JONES.CMSLIB]BASELEVEL1

20 CMS> SHOW RESERVATIONS

   Reservations in DEC/CMS Library DISKX:[JONES.CMSLIB]

   CMDRMVGRO.BLI
       (1)   JONES    1    24-JAN-1988 17:45:46 "need to add merging to these files"
   CMDRMVGRO.SDML
       (1)   JONES    1    24-JAN-1988 17:46:47 "need to add merging to these files"
   DIFF_DESIGN
       (1)   JONES    1    24-JAN-1988 17:48:29 "need to add merging to these files"

21 CMS>  SHOW GROUP USER_MANUAL/CONTENTS

   Groups in DEC/CMS Library DISKX:[JONES.CMSLIB]

   USER_MANUAL      "creating group for the project user's manual"
       CMDRMVGRO.BLI
       CMDRMVGRO.SDML
       CMS$$GSR.TXT
```

```
㉒ CMS>  EXIT
  $
```

❶ User Jones logs in.

❷ Jones displays the directory for the default directory DISKX:[JONES].

❸ Jones creates the subdirectory [.CMSLIB] from the default directory.

❹ Jones invokes the CMS image and enters the CMS subsystem.

❺ Jones creates the CMS library [.CMSLIB] with the CREATE LIBRARY command.

❻ Jones issues the CREATE ELEMENT command. In this case, all files from the main default directory are created as elements in the CMS library. The files are not deleted from Jones's default directory because the /KEEP qualifier was specified on the CREATE ELEMENT command.

❼ Jones exits from CMS and logs out.

❽ Jones later logs in and reenters CMS.

❾ Jones sets the library to [JONES.CMSLIB].

❿ Jones then displays all the elements in the library with the SHOW ELEMENT command.

⓫ Jones creates a group named USER_MANUAL.

⓬ Jones then inserts the two elements CMDRMVGRO.BLI and CMDRMVGRO.SDML into the group USER_MANUAL.

⓭ Jones decides that an element from the project directory is needed, and specifies the /INPUT qualifier on the CREATE ELEMENT command to indicate that the element is located in a different directory from the default directory. Because Jones did not specify /KEEP, the file will be deleted from the project directory.

⓮ Jones then inserts the element into the group USER_MANUAL.

⓯ Jones creates a class named BASELEVEL1 with the CREATE CLASS command.

⓰ Jones reserves the element DIFF_DESIGN.MEM and the group USER_MANUAL from the CMS library. CMS places the element DIFF_DESIGN.MEM and the contents (in this case, elements) of group USER_MANUAL in Jones's default directory. Jones can then modify these files as necessary.

⓱ Jones had previously reserved the element CMS$$GSR.TXT (which is part of the group USER_MANUAL), and made changes to that file. Jones replaces the element from the default directory [JONES] back into the CMS library [JONES.CMSLIB].

⓲ Jones issues the SHOW GENERATION command to display the last generation on the main line of descent for each element in the CMS library.

⓳ Jones then inserts generation 1 of the element CMS$$GSR.TXT and a generation of the element MAIL_FIL_KEY.COM into class BASELEVEL1. (If you do not specify the /GENERATION qualifier on an element, CMS uses the latest generation.)

⓴ Jones displays all current reservations.

㉑ Jones displays the contents of the group USER_MANUAL.

㉒ Jones exits from CMS.

1.6 Command Summary

Table 1–1 lists and briefly describes all CMS commands.

Table 1–1: CMS Command Summary

Command	Description
ACCEPT GENERATION	Changes the review status of one or more generations from pending to accepted and removes them from the review pending list.
ANNOTATE	Creates a listing file (element-name.ANN) that includes the element history and an annotated source listing.
CANCEL REVIEW	Changes the review status of one or more element generations from pending to none and removes them from the review pending list.
CONVERT LIBRARY	Converts libraries that were created with Version 2.n of CMS for use with Version 3.n of CMS.
COPY ELEMENT	Copies one or more existing library elements (including generation history and file attributes) to form one or more new elements.
CREATE CLASS	Establishes one or more classes. Once a class is established, any set of element generations can be placed in that class with the INSERT GENERATION command.

(continued on next page)

Table 1–1 (Cont.): CMS Command Summary

Command	Description
CREATE ELEMENT	Establishes one or more new elements in a CMS library by moving one or more files into the CMS library. By default, CMS deletes all copies of the input file after creating the element.
CREATE GROUP	Establishes one or more groups. Once a group is established, any set of elements or groups can be placed in that group with the INSERT ELEMENT or INSERT GROUP command.
CREATE LIBRARY	Creates one or more CMS libraries by loading one or more empty directories with CMS control structures.
DELETE CLASS	Deletes one or more classes from the library.
DELETE ELEMENT	Deletes one or more elements from the library.
DELETE GENERATION	Deletes one or more generations from one or more elements in the library.
DELETE GROUP	Deletes one or more groups from the library.
DELETE HISTORY	Deletes some or all of the library history.
DIFFERENCES	Compares the contents of two files and creates a listing file (filename.DIF) showing all the lines that differ. DIFFERENCES can also compare element generations in a CMS library, or a file to an element generation.
FETCH	Retrieves a copy of one or more specified element generations.
HELP	Provides online CMS help.
INSERT ELEMENT	Places one or more elements in one or more groups.
INSERT GENERATION	Places one or more element generations in one or more classes.
INSERT GROUP	Places one or more groups in another group or groups.
MARK GENERATION	Changes the review status of one or more generations to pending and adds them to the review pending list.

(continued on next page)

Table 1–1 (Cont.): CMS Command Summary

Command	Description
MODIFY CLASS	Changes the attributes of a class from those established with the CREATE CLASS command or with a previous MODIFY CLASS command.
MODIFY ELEMENT	Changes the attributes of one or more elements from those established with the CREATE ELEMENT command or with a previous MODIFY ELEMENT command.
MODIFY GENERATION	Changes the attributes of one or more generations from those established with the CREATE ELEMENT or REPLACE command or with a previous MODIFY GENERATION command.
MODIFY GROUP	Changes the attributes of one or more groups from those established with the CREATE GROUP command or with a previous MODIFY GROUP command.
MODIFY LIBRARY	Changes the attributes of the library from those established with the CREATE LIBRARY command or with a previous MODIFY LIBRARY command.
REJECT GENERATION	Changes the review status of one or more generations from pending to rejected and removes them from the review pending list.
REMARK	Enters a remark in the library history.
REMOVE ELEMENT	Removes one or more elements from one or more groups.
REMOVE GENERATION	Removes one or more generations from one or more classes.
REMOVE GROUP	Removes one or more groups from another group or groups.
REPLACE	Returns the most recent version of one or more reserved generations to the library, thus creating a new generation of each element. The reservation ends, and CMS deletes all versions of the input file.
RESERVE	Delivers a copy of one or more generations and marks them as reserved.

(continued on next page)

Table 1–1 (Cont.): CMS Command Summary

Command	Description
RETRIEVE ARCHIVE	Delivers a copy of one or more generations from one or more archive files created with the DELETE GENERATION/ARCHIVE command.
REVIEW GENERATION	Associates a review comment with one or more generations that are currently under review.
SET ACL	Manipulates access control lists on various objects in the CMS library.
SET LIBRARY	Identifies one or more existing CMS libraries so that subsequent CMS commands refer to the specified library or libraries.
SET NOLIBRARY	Removes one or more libraries from the current library search list.
SHOW ACL	Displays the access control list associated with one or more specified objects.
SHOW ARCHIVE	Displays information about the contents of one or more archive files created with the DELETE GENERATION/ARCHIVE command.
SHOW CLASS	Displays one or more established classes.
SHOW ELEMENT	Displays information about one or more elements.
SHOW GENERATION	Displays a listing of one or more established generations.
SHOW GROUP	Displays a listing of one or more established groups.
SHOW HISTORY	Displays a chronological listing of all CMS transactions that have affected the library.
SHOW LIBRARY	Displays the current library directory specification or list of library directory specifications.
SHOW RESERVATIONS	Displays a listing of all current reservations and concurrent replacements.

(continued on next page)

Table 1–1 (Cont.): CMS Command Summary

Command	Description
SHOW REVIEWS_PENDING	Displays a listing of generations that currently have reviews pending, and also displays any associated review remarks.
SHOW VERSION	Displays the version number of your CMS system.
UNRESERVE	Cancels an existing reservation.
VERIFY	Performs a series of consistency checks on your CMS library to confirm that all elements are present and stored properly.

Chapter 2

Using CMS with DECwindows

This chapter describes how you use CMS with the DECwindows interface. It describes how to invoke CMS in the DECwindows environment, how to get help, how to display information, and shows a sample session.

Before continuing with this chapter, you should be familiar with the basic DECwindows concepts described in the *VMS DECwindows User's Guide*. The *VMS DECwindows User's Guide* describes the DECwindows user interface, how to begin a session, how to interact with the session manager, how to use and manage windows, how to use the mouse to choose objects, and how to run a DECwindows application.

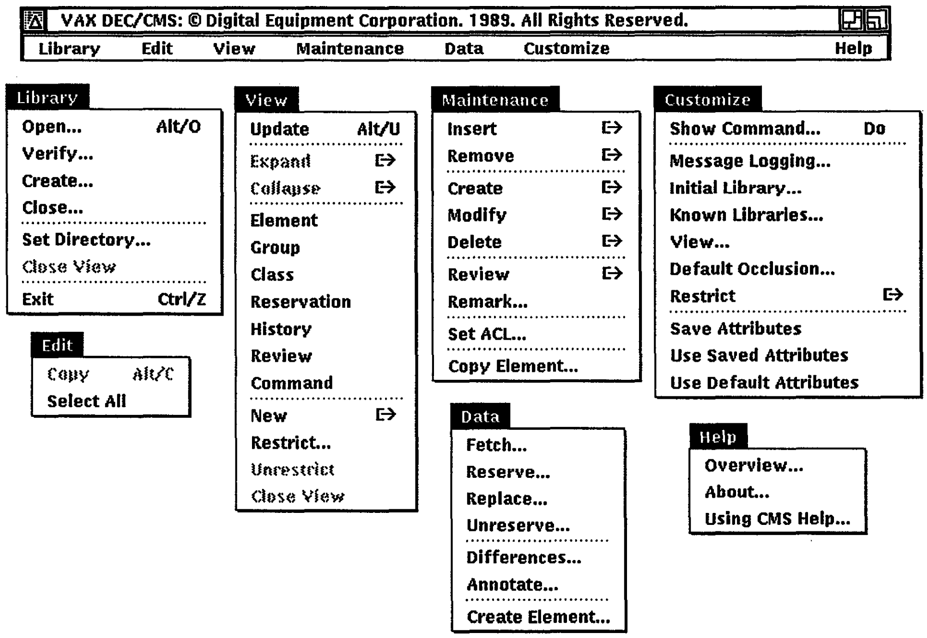
2.1 Invoking CMS

To invoke the CMS DECwindows interface, enter the following command:

```
$  CMS/INTERFACE=DECWINDOWS
```

Figure 2–1 shows the menu bar and the individual menus you can select from the menu bar.

Figure 2–1: CMS DECwindows Title Bar and Menus



In Figure 2–1, the menus are separated from the menu bar to show you all the CMS main menus at once; you pull down one main menu at a time.

2.2 Getting Help

You obtain help in the DECwindows environment by pulling down the Help menu. Help provides brief information about screen objects, concepts, and tasks that you can perform in CMS.

The CMS DECwindows interface has online help that provides complete information on all screen objects, including scroll bars, icons, menus, dialog boxes, text fields, buttons, and functions. The online help is context-sensitive. To get online help, follow these steps:

1. Position the pointer on the desired object.
2. Press and hold the HELP key while you press MB1.

3. Release both keys.

A Help window opens to display information about the object.

2.3 Displaying CMS Information in DECwindows

You display and obtain information about CMS objects through views. Views replace the CMS SHOW commands.

In the DECwindows environment, CMS provides the following types of views:

- Element
- Group
- Class
- Reservation
- History
- Review
- Command

When you invoke the CMS DECwindows interface for the first time, CMS displays an Element View. This is a view of all elements in your current library. However, if you have opened multiple libraries, CMS displays each library name. To obtain a different view, follow these steps:

1. Pull down the View menu.
2. Choose the desired view.

CMS displays the appropriate view for the type you choose; for instance, if you choose a group view, CMS displays the names of all the groups in the library. However, if you have more than one library open, CMS displays only each library name. You must then expand each library into the groups it contains.

Displaying More Than One View

A single view can display only one type of information at a time; however, you can display multiple view windows. To obtain multiple view windows, follow these steps:

1. Pull down the View menu.
2. Choose the New menu item; the New submenu appears.

3. Choose the desired view.

CMS displays an additional window with the view you choose.

You can display any number of views that you want; each view is independent of other views. By using CMS views, you can quickly and easily choose objects on which you want to perform functions.

Restricting Views

You can restrict views to display objects meeting certain criteria. For instance, you could restrict a Reservations View to display only reservations made by a particular user by following these steps:

1. Pull down the View menu.
2. Choose the Reservations menu item.
3. Pull down the View menu.
4. Choose the Restrict... menu item.

A dialog box appears, enabling you to specify the user name for which CMS should display reservations.

Customizing Your Initial View

CMS enables you to customize your CMS session by specifying which view you want displayed on startup. Do this by following these steps:

1. Pull down the Customize menu.
2. Choose the View... menu item.
3. Choose the desired view.
4. Pull down the Customize menu.
5. Choose the Save Attributes menu item.

You can also obtain information about CMS objects by expanding them. See Section 2.3.1 for more information.

2.3.1 Expanding and Collapsing CMS Objects

The CMS DECwindows interface provides the following ways to expand and choose objects:

- Double-click on an object to expand it.

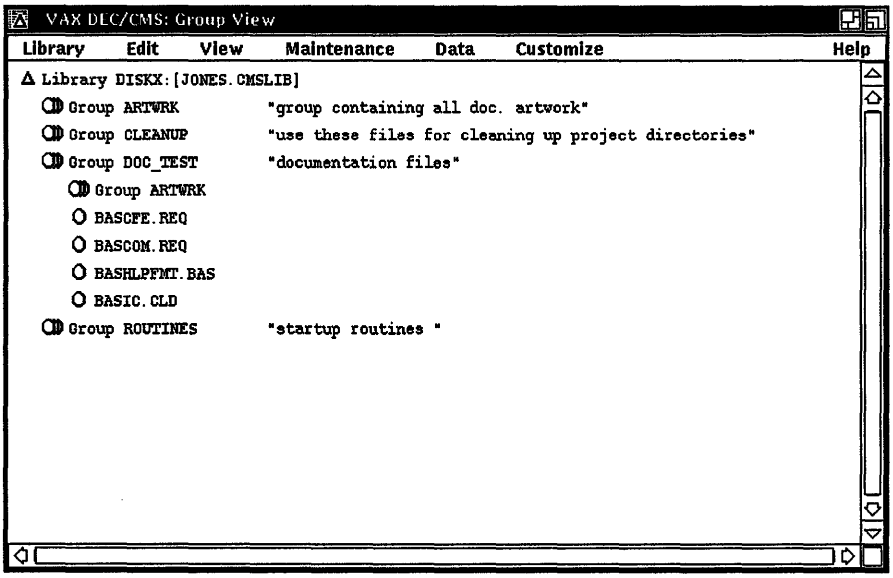
- Choose a menu item, then specify the name of the object in the associated dialog box. Or, first click on an object and then choose a menu item and provide information about it in the associated dialog box.
- Click on an object, then press MB2 to obtain a pop-up menu.

The following sections describe these methods.

2.3.1.1 Double Clicking

Figure 2–2 shows the group DOC_TEST expanded to show its children.

Figure 2–2: Expanding a Group



```
Δ Library DISKX:[JONES.CMSLIB]
   Group ARTWRK        "group containing all doc. artwork"
   Group CLEANUP       "use these files for cleaning up project directories"
   Group DOC_TEST      "documentation files"
      Group ARTWRK
      BASCFE.REQ
      BASCOM.REQ
      BASHLPFMT.BAS
      BASIC.CLD
   Group ROUTINES      "startup routines "
```

To expand this group using double clicking, follow these steps:

1. Pull down the View menu.
2. Choose the Group menu item.
3. Double click on the desired group (in this example, group DOC_TEST).

Group DOC_TEST expands into its components, including elements and any other groups that are contained in group DOC_TEST. Double clicking on the element BASCOM.REQ expands it into its generations.

NOTE

> If an item is expanded fully, double clicking collapses the information into the previous level of information.

You can also expand an object by choosing a function. For example, to expand the group DOC_TEST, follow these steps:

1. Click on the desired object (in this example, group DOC_TEST).
2. Pull down the View menu.
3. Choose the Expand menu item; the Expand submenu appears.
4. Choose the Children submenu item.

Section 2.3.1.2 contains more information about choosing a function.

2.3.1.2 Choosing a Function

Most of the functions performed on CMS objects are grouped into two menus: Maintenance and Data. The Maintenance menu contains all the functions that are completely internal to CMS libraries, while the Data menu contains all the functions that involve data transfers between CMS libraries and external files.

To choose an object and perform a specific operation, use one of the following methods:

- Click on an object, then choose a menu item and provide information about the object in the associated dialog box. For example, to reserve an element, follow these steps:
  1. Click on an element.
  2. Pull down the Data menu.
  3. Choose the Reserve... menu item.

  A dialog box appears, with the name of the element you have chosen in the Selected list box. You can then enter additional information about the element and the reserve function, and click on the OK button.
- Choose a menu item, then specify the name of the object in the associated dialog box. For example, to reserve an element, follow these steps:
  1. Pull down the Data menu.

2. Click on the Reserve... menu item.
3. Click on the Element field in the Reserve... dialog box.
4. Fill in the Element field with the name of the element you want to reserve.

You can then enter additional information about the element and the reserve function, and click on the OK button.

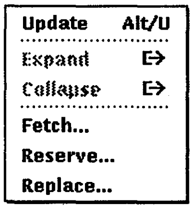
.3.1.3 Using the Pop-Up Menu

CMS provides a pop-up menu enabling you to quickly access some of the most commonly used CMS functions. You can use the pop-up menu with any CMS object that can be used in those functions.

To get the pop-up menu, press and hold MB2. Or, to first choose an object for the operation, click on the object, then press and hold MB2 to get the pop-up menu.

Figure 2–3 shows the pop-up menu.

Figure 2–3: CMS Pop-Up Menu



.4 CMS Command Correspondence

Most command-line interface CMS commands have a corresponding menu path in the DECwindows interface; however, there is not a complete one-to-one correspondence. Because you execute commands in DECwindows by selecting objects on your screen instead of typing command lines, certain CMS functions are not applicable in the DECwindows environment.

The following list describes those CMS commands that are not included in the CMS DECwindows interface:

- CONVERT LIBRARY

- RETRIEVE ARCHIVE
- SHOW ARCHIVE

You can invoke the CMS command line from the DECwindows interface by entering command mode. Do this by following these steps:

1. Pull down the Customize menu.
2. Choose the Show Command... menu item.

A dialog box appears, containing an output window and the CMS command-line prompt. Enter CMS command-line interface commands at the CMS prompt (CMS>). CMS displays the resulting command output in the output window (see Figure 2–10).

2.5 Sample CMS DECwindows Session

This section shows examples that use CMS with DECwindows.

NOTE

Many of the figures in this section show dialog boxes from which you initiate tasks. These figures also show the menu and menu item from which the dialog box is invoked.

2.5.1 Creating a CMS Library

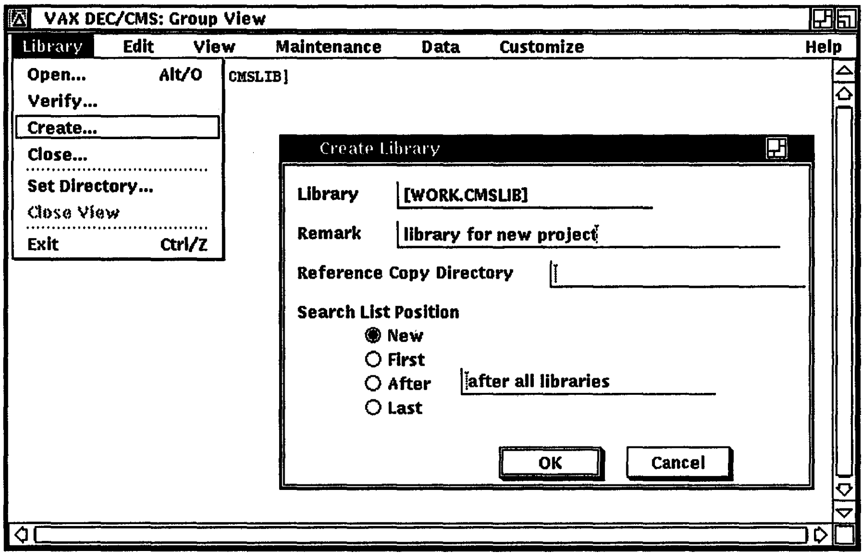
Figure 2–4 shows how to create a new CMS library. Do this by following these steps:

1. Pull down the Library menu.
2. Choose the Create... menu item.
3. Specify the directory name in the subsequent dialog box.

To specify an existing library, follow these steps:

1. Pull down the Library menu.
2. Choose the Open... menu item.
3. Specify the library name in the subsequent dialog box.

Figure 2–4: Creating a New CMS Library

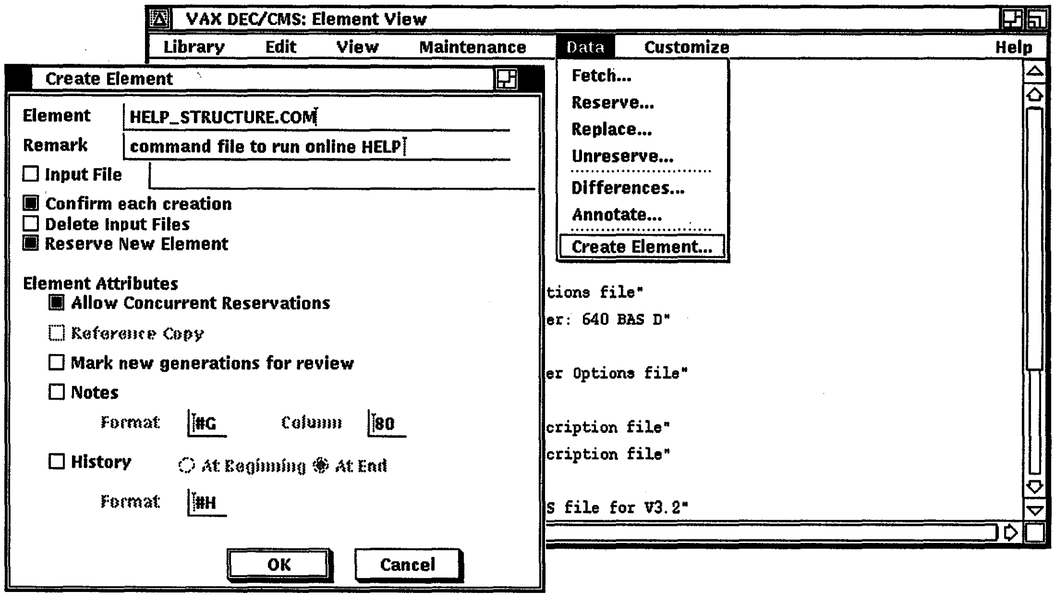


Library [WORK.CMSLIB]
Remark library for new project
Reference Copy Directory
Search List Position
[✓ New]
First
After after all libraries
Last

2.5.2 Creating an Element

Figure 2–5 shows how to create the element HELP_STRUCTURE.COM. The Create Element... submenu is pulled down from the Data menu, and the resulting dialog box is popped up. The user has filled in the Element and Remark field with an element name and remark, respectively.

Figure 2–5: Creating an Element


VAX DEC/CMS: Element View

Library Edit View Maintenance Data Customize Help

Fetch...
Reserve...
Replace...
Unreserve...
Differences...
Annotate...
Create Element...

Create Element

Element: HELP_STRUCTURE.COM
Remark: command file to run online HELP
☐ Input File
■ Confirm each creation
☐ Delete Input Files
■ Reserve New Element

Element Attributes
■ Allow Concurrent Reservations
☐ Reference Copy
☐ Mark new generations for review
☐ Notes
Format: #G Column: 80
☐ History ○ At Beginning ● At End
Format: #H

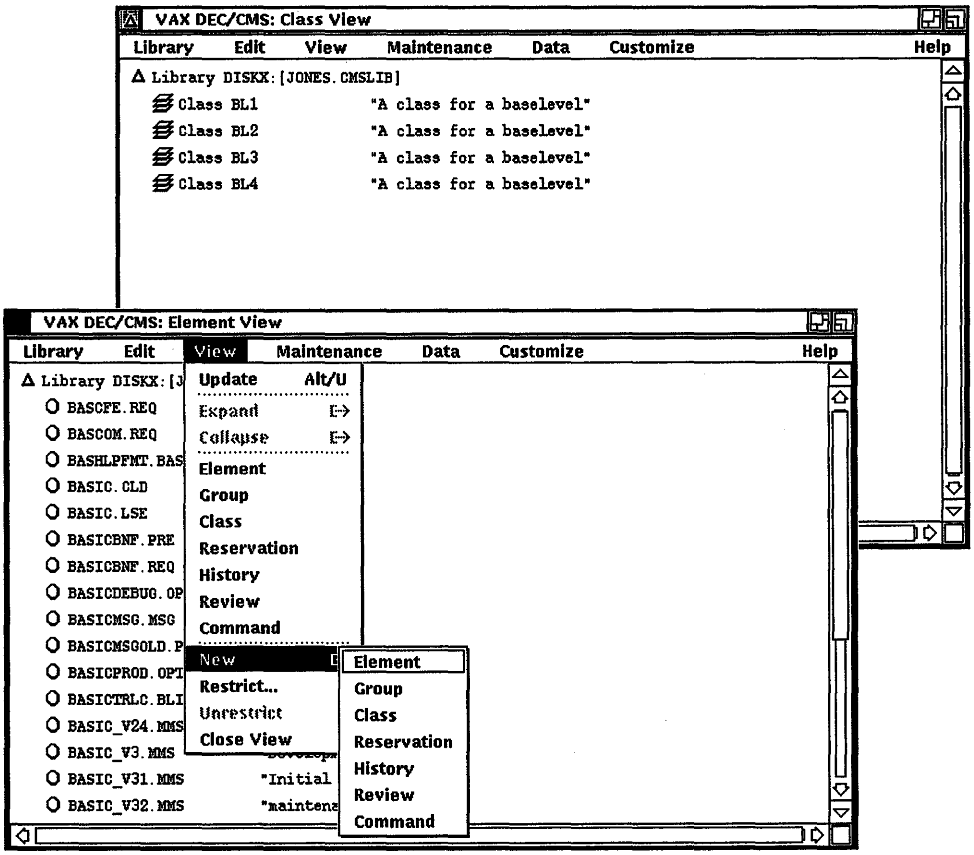
OK Cancel

```
tions file"
er: 640 BAS D"
er Options file"
cription file"
cription file"
S file for V3.2"
```

2.5.3 Displaying Two Views

In Figure 2–6, the library class structure is shown in the first view; the View menu is pulled down, and a new element view is chosen. The second view contains the element view. The element BASCOM.REQ has been expanded (by double clicking) to show its generations.

Figure 2–6: Displaying Two Views



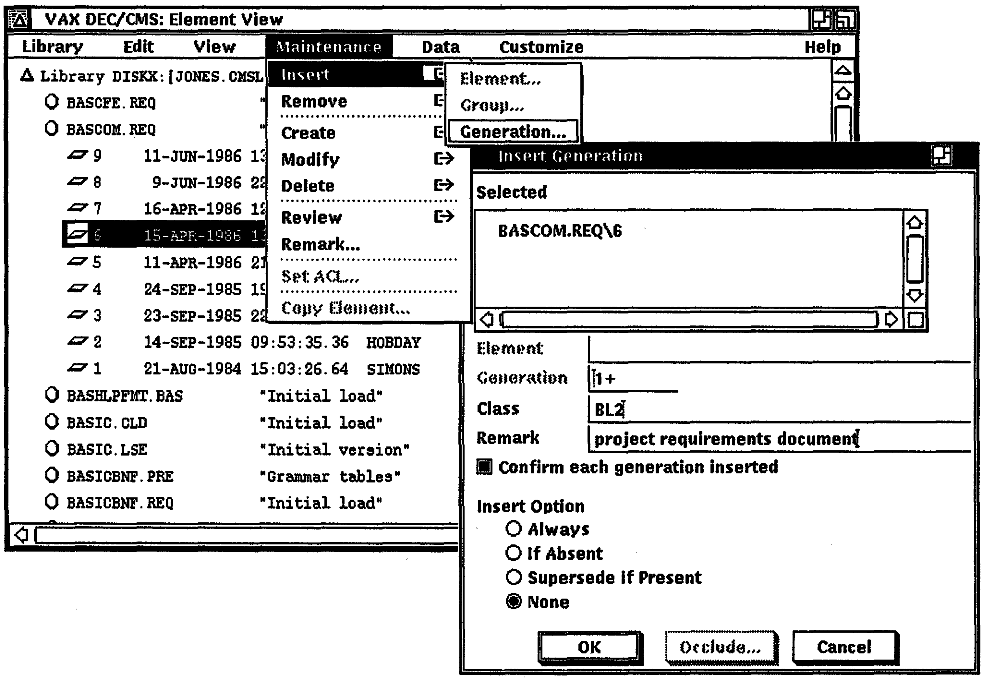
```
Δ Library DISKX:[JONES.CMSLIB]
   Class BL1            "A class for a baselevel"
   Class BL2            "A class for a baselevel"
   Class BL3            "A class for a baselevel"
   Class BL4            "A class for a baselevel"
```

```
Δ Library DISKX:[J
   O BASCFE.REQ
   O BASCOM.REQ
   O BASHLPFMT.BAS
   O BASIC.CLD
   O BASIC.LSE
   O BASICBNF.PRE
   O BASICBNF.REQ
   O BASICDEBUG.OP
   O BASICMSG.MSG
   O BASICMSGOLD.P
   O BASICPROD.OPT
   O BASICTRLC.BLI
   O BASIC_V24.MMS
   O BASIC_V3.MMS
   O BASIC_V31.MMS       "Initial
   O BASIC_V32.MMS       "maintena
```

2.5.4 Inserting a Generation into a Class

In Figure 2–7, generation 6 of the element BASCOM.REQ has been chosen for the Insert Generation function. The Selected list box confirms that generation 6 of BASCOM.REQ is being inserted into class BL2.

Figure 2–7: Inserting a Generation into a Class



```
Δ Library DISKX:[JONES.CMSL
  O BASCFE.REQ                "
  O BASCOM.REQ                "
      9   11-JUN-1986 1
      8    9-JUN-1986 2
      7   16-APR-1986 1
      6   15-APR-1986 1
      5   11-APR-1986 2
      4   24-SEP-1985 1
      3   23-SEP-1985 2
      2   14-SEP-1985 09:53:35.36 HOBDAY
      1   21-AUG-1984 15:03:26.64 SIMONS
  O BASHLPFMT.BAS             "Initial load"
  O BASIC.CLD                 "Initial load"
  O BASIC.LSE                 "Initial version"
  O BASICBNF.PRE              "Grammar tables"
  O BASICBNF.REQ              "Initial load"
```

Selected
BASCOM.REQ\6

Element	
Generation	1+
Class	BL2
Remark	project requirements document

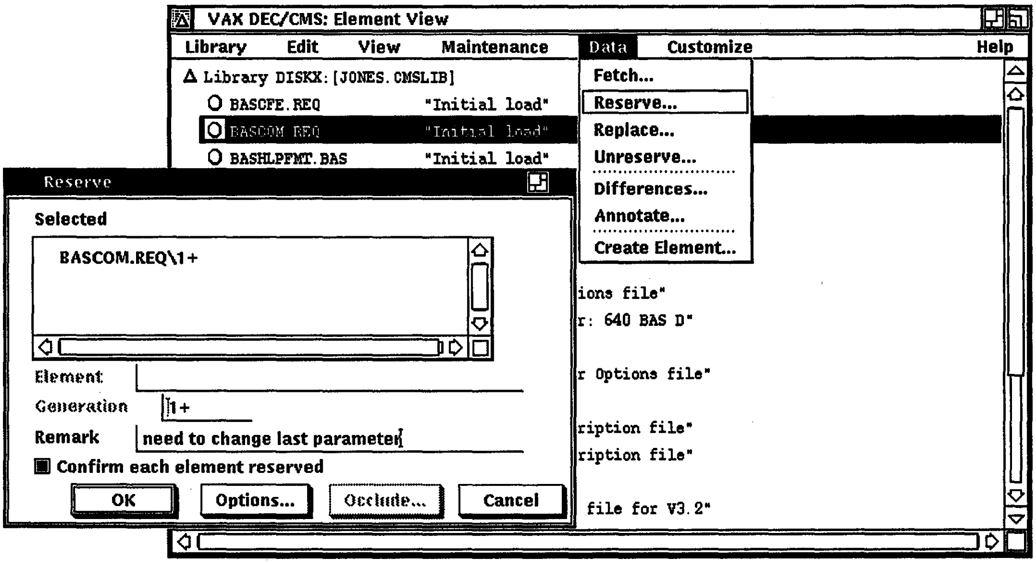
■ Confirm each generation inserted

Insert Option
- ○ Always
- ○ If Absent
- ○ Supersede if Present
- ● None

2.5.5 Reserving an Element

In Figure 2–8, the latest generation of the element BASCOM.REQ is reserved. Because BASCOM.REQ has already been chosen, the Reserve dialog box shows BASCOM.REQ in the Selected list box (the \1+ following the element generation name indicates the latest generation, and is included by default).

Figure 2–8: Reserving an Element



```
Library DISKX:[JONES.CMSLIB]
  BASCFE.REQ        "Initial load"
  BASCOM.REQ        "Initial load"
  BASHLPFMT.BAS     "Initial load"
```

```
ions file"
r: 640 BAS D"
r Options file"
ription file"
ription file"
file for V3.2"
```

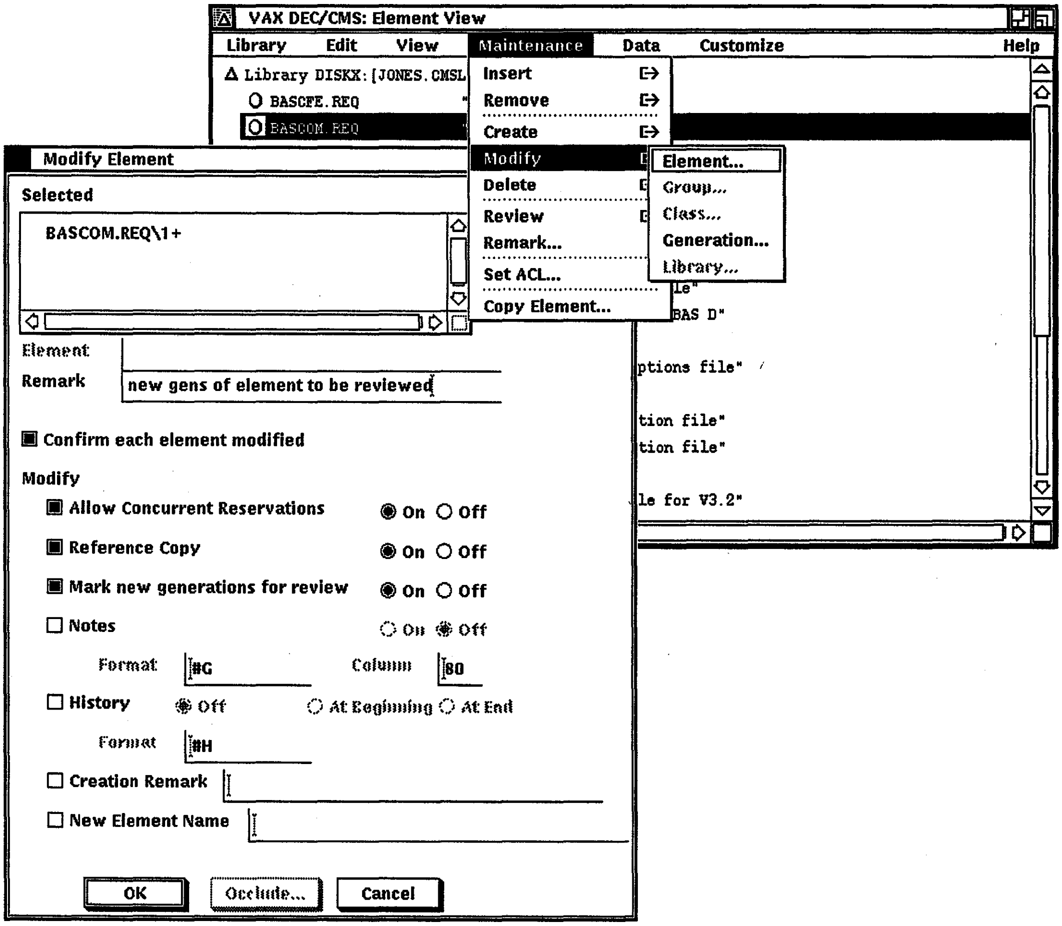
Selected: BASCOM.REQ\1+

Element:
Generation: 1+
Remark: need to change last parameter
■ Confirm each element reserved

2.5.6 Modifying an Element

In Figure 2–9, the element BASCOM.REQ is chosen for the modify transaction. Because BASCOM.REQ has already been chosen, the Modify Element dialog box shows BASCOM.REQ in the Selected list box (the \1+ following the element generation name indicates the latest generation, and is included by default).

Figure 2–9: Modifying an Element



Library DISKX:[JONES.CMSL
BASCFE.REQ
BASCOM.REQ

Selected
BASCOM.REQ\1+

Element
Remark: new gens of element to be reviewed

■ Confirm each element modified

Modify
■ Allow Concurrent Reservations [✓ On] Off
■ Reference Copy [✓ On] Off
■ Mark new generations for review [✓ On] Off
☐ Notes On [✓ Off]
Format #G Column 80
☐ History [✓ Off] At Beginning At End
Format #H
☐ Creation Remark
☐ New Element Name

2.6 Customizing Your CMS DECwindows Interface

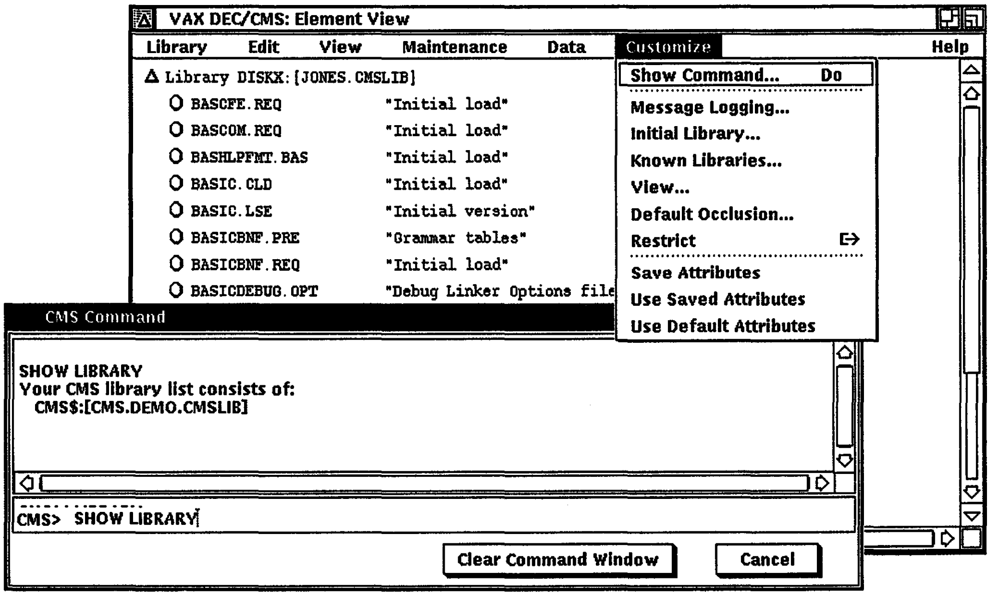
The CMS DECwindows interface enables you to conveniently customize many options, including:

- Message logging options
- The initial library to open each time you enter CMS

- A library (or libraries) that are most commonly used, which you can specify once, then conveniently open by choosing them from a list
- The default view to be displayed each time you enter CMS
- The default occlusion
- Default restrictions for each view type

Figure 2–10 shows the command mode dialog box. The command SHOW LIBRARY has been entered at the CMS command line prompt (CMS>), and the resulting information is displayed in the output box.

Figure 2–10: Command Mode



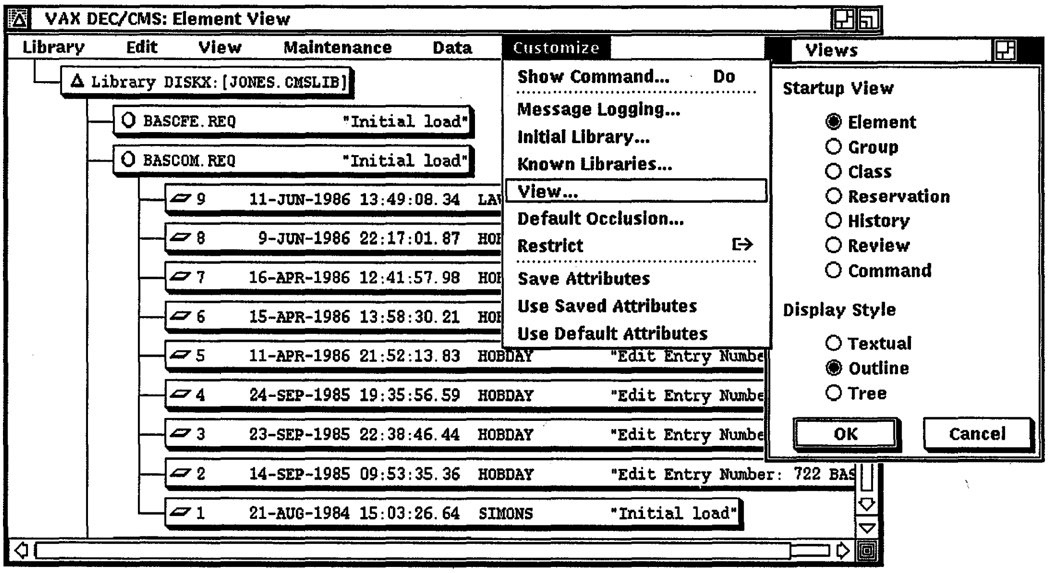
```
Library DISKX:[JONES.CMSLIB]
   BASCFE.REQ          "Initial load"
   BASCOM.REQ          "Initial load"
   BASHLPFMT.BAS       "Initial load"
   BASIC.CLD           "Initial load"
   BASIC.LSE           "Initial version"
   BASICBNF.PRE        "Grammar tables"
   BASICBNF.REQ        "Initial load"
   BASICDEBUG.OPT      "Debug Linker Options file
```

```
SHOW LIBRARY
Your CMS library list consists of:
  CMS$:[CMS.DEMO.CMSLIB]
```

```
CMS> SHOW LIBRARY
```

In Figure 2–11, the view display has been customized to show the library structure in outline form. The element BASCOM.REQ has been expanded to show its generations.

Figure 2–11: Outline View


VAX DEC/CMS: Element View

```
Library DISKX:[JONES.CMSLIB]
  BASCFE.REQ          "Initial load"
  BASCOM.REQ          "Initial load"
    9   11-JUN-1986 13:49:08.34  LA
    8    9-JUN-1986 22:17:01.87  HO
    7   16-APR-1986 12:41:57.98  HO
    6   15-APR-1986 13:58:30.21  HO
    5   11-APR-1986 21:52:13.83  HOBDAY   "Edit Entry Numbe
    4   24-SEP-1985 19:35:56.59  HOBDAY   "Edit Entry Numbe
    3   23-SEP-1985 22:38:46.44  HOBDAY   "Edit Entry Numbe
    2   14-SEP-1985 09:53:35.36  HOBDAY   "Edit Entry Number: 722 BAS
    1   21-AUG-1984 15:03:26.64  SIMONS   "Initial load"
```

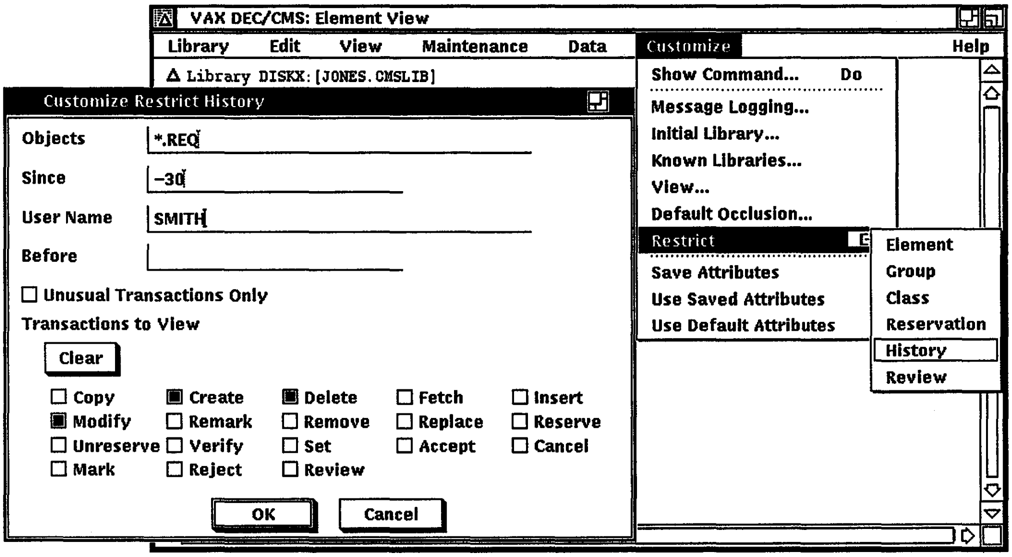
Views

Startup View: [✓ Element] Group Class Reservation History Review Command

Display Style: Textual [✓ Outline] Tree

In Figure 2–12, the Customize Restrict History dialog box is shown. The user has specified that the history view contain only elements with the file type .REQ that have been modified, created, or deleted in the last 30 days by user SMITH.

Figure 2–12: Restricting History



Objects	*.REQ
Since	-30
User Name	SMITH
Before	

☐ Unusual Transactions Only

Transactions to View

☐ Copy	■ Create	■ Delete	☐ Fetch	☐ Insert
■ Modify	☐ Remark	☐ Remove	☐ Replace	☐ Reserve
☐ Unreserve	☐ Verify	☐ Set	☐ Accept	☐ Cancel
☐ Mark	☐ Reject	☐ Review		

Chapter 3

Libraries

A CMS library consists of a set of defined objects that can be operated on by CMS commands. A CMS library resides in a directory that has been initialized for use solely by CMS.

This chapter discusses how to create and use CMS libraries and how to control occlusion of CMS objects, and describes library locking.

3.1 Creating Libraries

This section describes how to create a CMS library. First, you must create a directory to contain the library; then you create the library, and create elements in it.

You can also optionally create a reference copy directory. A reference copy directory is a directory used for storing copies of the latest generation on the main line of descent for specified elements in a CMS library. See Section 3.1.4 for more information.

3.1.1 Creating the Directory

You create a directory to contain your CMS library by using the DCL command CREATE/DIRECTORY. The format of the command is as follows:

CREATE/DIRECTORY directory-specification

For example:

```
$  CREATE/DIRECTORY [PROJECT.CMSLIB]
```

The name PROJECT identifies the first-level directory. This command creates the empty subdirectory [.CMSLIB] within the directory [PROJECT]. For more information on the CREATE/DIRECTORY command, see the *VMS DCL Dictionary*.

A directory specification can refer to either a first-level directory or a subdirectory. To create a first-level directory, you must have write access to the master file directory (MFD) on the volume on which you are creating the directory. Normally, on a system volume, only users with a system user identification code (UIC) or the SYSPRV or BYPASS user privilege are allowed write access to the MFD to create a first-level directory. To create a subdirectory, you must have write access to the next higher directory level. For more information on directory specifications, see Chapter 10.

NOTE

You should not place any version limit on a CMS library; CMS automatically purges and deletes unused files within a library. A library must have a file retention count of at least 2 to allow error recovery in case of system failure.

VMS limits directory trees to a depth of eight. Because CMS may create subdirectories, you should not create a library in an eighth-level directory.

If you want to place access control lists (ACLs) on the library directory, you should do so before you create the library, so that files created during library creation are assigned the correct protection. See Chapter 7 for more information on ACLs.

3.1.2 Creating the Library

You create a CMS library with the CREATE LIBRARY command. The CREATE LIBRARY command creates CMS control files in the specified directory. The directory must exist and must be empty. Once you create a library in a directory, CMS uses that directory to locate and store files. Note that your default directory cannot be a CMS library. After you create a library with the CREATE LIBRARY command, all further CMS commands refer to this library until the end of the terminal session, until you specify a different library with the SET LIBRARY command, or until you deassign the library list with the SET NOLIBRARY command.

The following command initializes a library in the empty directory [PROJECT.CMSLIB]:

```
$ CMS CREATE LIBRARY [PROJECT.CMSLIB]
_Remark:  test procedure library
%CMS-S-CREATED, CMS library DISKX:[PROJECT.CMSLIB] created
%CMS-I-LIBIS, Library is DISKX:[PROJECT.CMSLIB]
%CMS-S-LIBSET, CMS library set
```

CAUTION

Once the library is created, you should access it only with CMS commands. If a library has been accessed by means other than CMS, it may result in unrecoverable library corruption. Files that have been placed into the library directory by means other than CMS may be deleted by CMS when the library is verified and repaired (see Chapter 9).

You can create more than one library with the CREATE LIBRARY command by specifying a list of directory specifications separated by commas. For more information, see Section 3.2.

The CREATE LIBRARY command also allows you to optionally specify a directory to be used for maintaining reference copies of library elements. For information about using reference copy directories, see Section 3.1.4.

3.1.3 Creating Elements in the Library

You store a file in a CMS library with the CREATE ELEMENT command. CREATE ELEMENT uses the input file you provide to create the first version of an element. This first version represents generation 1 of the element. An element represents all of the versions of a particular file as it is developed. Every element in the CMS library must have a unique name.

The following is an example of the CREATE ELEMENT command:

```
$ CMS CREATE ELEMENT OUTPUT.FOR "ascii output format routines"
%CMS-S-CREATED, element [PROJECT.CMSLIB]OUTPUT.FOR created
```

This command creates the element named OUTPUT.FOR. Generation 1 of element OUTPUT.FOR now exists in the library.

The file specified in the CREATE ELEMENT command must be present in your current default directory (unless you specify a different location by using the /INPUT qualifier). CMS deletes all copies of that file from the default or specified directory after creating the new element. You can override this default by specifying the /KEEP or /RESERVE qualifier on the

CREATE ELEMENT command. The contents of the file used to create the element become generation 1 of that element.

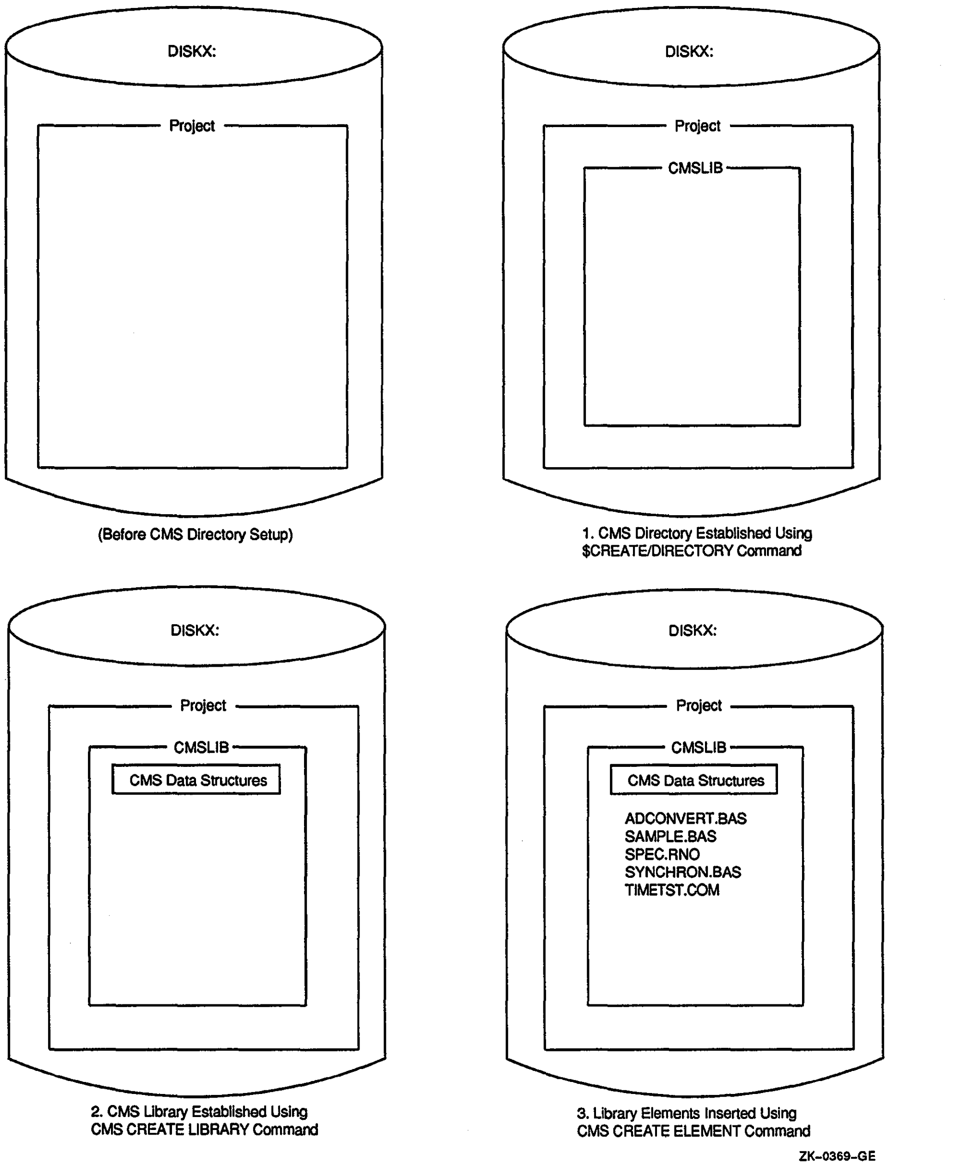
CMS can store and operate on nontext files; however, CMS cannot store directory files.

There is no explicit limit on the number of elements (or groups or classes) that can exist in a library. However, there may be limits imposed by your system configuration including system, process, disk space, and virtual memory limitations.

To create an element in the library, you must have read access to the file from which you are creating the element.

Figure 3–1 shows the process of establishing a library and creating elements in it. See Chapter 4 for more information about elements.

Figure 3–1: Building a CMS Library


DISKX:
Project
(Before CMS Directory Setup)
DISKX:
Project
CMSLIB
1. CMS Directory Established Using $CREATE/DIRECTORY Command
DISKX:
Project
CMSLIB
CMS Data Structures
2. CMS Library Established Using CMS CREATE LIBRARY Command
DISKX:
Project
CMSLIB
CMS Data Structures
ADCONVERT.BAS
SAMPLE.BAS
SPEC.RNO
SYNCHRON.BAS
TIMETST.COM
3. Library Elements Inserted Using CMS CREATE ELEMENT Command
ZK-0369-GE

3.1.4 Creating a Reference Copy Directory

A *reference copy directory* is a directory in which CMS maintains a copy of the latest generation on the main line of descent of each element.

The reference copy directory cannot be a CMS library, nor can it be a subdirectory of a CMS library directory. Although CMS allows different libraries to be assigned the same reference copy directory, it is strongly recommended that you assign each CMS library its own unique reference copy directory.

To establish a reference copy directory, first create a directory (see Section 3.1.1), and then use the /REFERENCE_COPY qualifier with the CREATE LIBRARY or MODIFY LIBRARY command. The /REFERENCE_COPY qualifier directs CMS to store the name of this directory in the library, creating a permanent association between the CMS library and this directory (unless you enter a MODIFY LIBRARY/NOREFERENCE_COPY command, which removes this association). For example:

```
$ CREATE/DIRECTORY [PROJECT.CMSLIB]
$ CREATE/DIRECTORY [PROJECT.REFCOPY]
$ CMS CREATE LIBRARY [PROJECT.CMSLIB]/REFERENCE_COPY=[PROJECT.REFCOPY]
_Remark:  Master library with reference copies
%CMS-S-CREATED, library DISKX:[PROJECT.CMSLIB] created
```

In this example, the first CREATE/DIRECTORY command creates the CMS library directory [PROJECT.CMSLIB]; the second CREATE/DIRECTORY command creates the reference copy directory [PROJECT.REFCOPY]. The CREATE LIBRARY command initializes a CMS library in the [PROJECT.CMSLIB] directory and creates a permanent association between the CMS library and the [PROJECT.REFCOPY] directory.

Once a reference copy directory is established for a library, CMS maintains reference copy files in that directory. Every time you create a new main-line generation of an element (by using CREATE ELEMENT or REPLACE), CMS updates the reference copy of that element. Existing elements in the library will not have the **reference copy** attribute set. Use the /REFERENCE_COPY qualifier on the MODIFY ELEMENT command to enable the **reference copy** attribute on those elements for which reference copies are to be maintained. See Section 4.5.3 for more information.

The following commands assign an existing CMS library a reference copy directory and create reference copies for existing elements:

```
$  CREATE/DIRECTORY [PROJECT.REFCOPY]
$  CMS
CMS>  SET LIBRARY [PROJECT.CMSLIB]
CMS>  MODIFY LIBRARY/REFERENCE_COPY=[PROJECT.REFCOPY]
_Remark:  Establish reference copy directory
CMS>  MODIFY ELEMENT/REFERENCE_COPY *.* "enable reference copy"
```

The MODIFY LIBRARY command establishes the directory [PROJECT.REFCOPY as the reference copy directory for the current CMS library [PROJECT.CMSLIB]. The MODIFY ELEMENT command changes the **reference copy** attribute for all currently existing elements and creates reference copies for them. Use the SHOW LIBRARY/FULL command to display the directory specification of a reference copy directory.

If you do not want some elements to have reference copies, modify those elements with the /NOREFERENCE_COPY qualifier on the MODIFY ELEMENT command. See the descriptions of the CREATE ELEMENT and MODIFY ELEMENT commands in the Command Dictionary for more information.

CMS does not create reference copies for any variant generations. CMS maintains a reference copy only of the latest generation on the main line of descent of each element.

3.2 Using Libraries

When you invoke CMS, you must explicitly set up a library environment to tell CMS which library (or libraries) you want to use. This sets a library search list. You do this by either creating a new library or libraries (with the CREATE LIBRARY command), or by selecting an existing library or libraries (with the SET LIBRARY command).

A CMS library search list is a list of one or more libraries. When CMS operates on multiple libraries, it accesses them in the order that you specified when you set the library list. If you invoke CMS and do not establish at least one library in the library search list, you receive an error indicating that your library environment is undefined and your library search list is empty.

Libraries in the library search list do not need to be related; each library is complete and self-contained. You can specify libraries in any order, but a library can appear only once in the library search list. You can specify a maximum of 128 libraries in a library search list.

After you establish a library search list, that list remains in effect for all further CMS commands until you modify it with the CREATE LIBRARY command or SET [NO]LIBRARY command, or log out.

3.2.1 Setting Libraries

You set one or more libraries by entering the SET LIBRARY command, or the CREATE LIBRARY command, which performs an implicit SET LIBRARY operation. The SET LIBRARY command defines a DCL logical name, CMS$LIB, which points to the library or libraries you have selected. After you have selected a library or libraries, all CMS commands you enter refer to the CMS$LIB library list. The library list exists until you enter another SET LIBRARY, SET NOLIBRARY, or CREATE LIBRARY command, or log out.

You can enter one or more library directory names as a parameter to the SET LIBRARY command. For example, to set your library to [PROJECT.CMSLIB], enter the following command:

```
$  CMS SET LIBRARY [PROJECT.CMSLIB]
%CMS-I-LIBIS, library is DISKX:[PROJECT.CMSLIB]
%CMS-S-LIBSET, library set
%CMS-I-SUPERSEDE, library list superseded
```

This command sets (or resets) the library search list to contain only the library [PROJECT.CMSLIB].

To set your library search list to contain more than one library, you would specify multiple libraries separated by commas. For example:

```
CMS>  SET LIBRARY [PROJECT1.CMSLIB],[PROJECT3.CMSLIB]
%CMS-I-LIBIS, library is DISKX:[PROJECT1.CMLSIB]
%CMS-I-LIBINSLIS, library DISKX:[PROJECT3.CMSLIB] inserted at end of library
list
%CMS-S-LIBSET, library set
```

This command sets (or resets) the library search list to contain only the two libraries [PROJECT1.CMSLIB] and [PROJECT3.CMSLIB].

If you try to set your library to a directory that has not been initialized by CREATE LIBRARY, CMS$LIB becomes undefined and CMS issues a warning message.

3.2.2 Modifying Library Lists

To add libraries to an established library search list, use the /BEFORE and /AFTER qualifiers on the CREATE LIBRARY or SET LIBRARY command to control the placement of the new libraries in the existing library search list. For example:

```
CMS>  CREATE LIBRARY [PROJECT1.CMSLIB],[PROJECT3.CMSLIB]
CMS>  SET LIBRARY [PROJECT2.CMSLIB]/BEFORE=[PROJECT3.CMSLIB]
```

In this example, the CREATE LIBRARY command establishes the library search list consisting of the two libraries [PROJECT1.CMSLIB] and [PROJECT3.CMSLIB]. The SET LIBRARY command inserts the library [PROJECT2.CMSLIB] in the library search list. The library search list now consists of three libraries: [PROJECT1.CMSLIB], [PROJECT2.CMSLIB], and [PROJECT3.CMSLIB], in that order.

If you specify either the /BEFORE or the /AFTER qualifier without a value, the new library (or libraries) are added to the existing search list either before or after the entire library list, respectively.

If you do not specify either qualifier, a new library search list is created, replacing the entire existing library search list.

To remove one or more libraries from the existing search list, use the SET NOLIBRARY command. The SET NOLIBRARY command accepts one or more library directory specifications, which are then removed from the list, leaving the rest of the list intact. If you do not specify a library directory, every library from the entire library search list is removed, and the library search list becomes undefined. See the Command Dictionary for more information on the SET NOLIBRARY command.

3.3 Controlling Occlusion in Multiple Libraries

CMS operates on your library search list by searching through the library (or libraries) in the list. If you have more than one library in the search list, CMS searches the libraries one at a time in the order they appear in the search list, until a specified object is found. Once the object is found, CMS performs the specified operation on the object, and by default does not continue to search for the object in any of the remaining libraries.

Objects with the same name can exist in more than one library. When an object exists in more than one library of a library search list, CMS processes only the first occurrence of the specified object and ignores any later instances of that object in subsequent libraries. This behavior is *occlusion*; that is, the first instance of the object occludes any subsequent instances of that object. For example:

```
CMS> SET LIBRARY [BOOK.CMSLIB],[EXAMPLES.CMSLIB],[TEMP.CMSLIB]
CMS> FETCH TESTBAS.SDML "fetch first instance"
```

In this example, CMS searches the library list, starting with the library [BOOK.CMSLIB], then [EXAMPLES.CMSLIB], then [TEMP.CMSLIB]. When CMS locates the element TESTBAS.SDML, it fetches the element from the first library in the list in which it finds it. For example, if the element TESTBAS.SDML existed in [EXAMPLES.CMSLIB] and in [TEMP.CMSLIB], CMS would fetch the element only from [EXAMPLES.CMSLIB], because that element would occlude the element in [TEMP.CMSLIB].

You control occlusion with the /OCCLUDE qualifier. The /OCCLUDE qualifier has the following format:

/OCCLUDE[=options,...]

You can specify the following options with this qualifier:

- [NO]CLASS—controls occlusion for classes
- [NO]ELEMENT—controls occlusion for elements
- [NO]GROUP—controls occlusion for groups
- [NO]OTHER—controls occlusion for library attributes, history, commands, the class list, the element list, and the group list
- ALL
- NONE

You can specify either ALL or NONE, or one or more of the remaining keywords in any combination. If you do not specify a keyword on the /OCCLUDE qualifier, the default is /OCCLUDE=ALL. The ALL keyword enables occlusion for all four object types; the NONE keyword disables occlusion for all four object types. To disable occlusion for a specific object, use the /OCCLUDE qualifier with a negated keyword. For example:

```
$  CMS SET LIBRARY [WORK.CMSLIB],[PROJECT.CMSLIB]
%CMS-I-LIBIS, library is DISKX:[WORK.CMSLIB]
%CMS-I-LIBINSLIS, library DISKX:[PROJECT.CMSLIB] inserted at end of library
list
%CMS-S-LIBSET, library set
%CMS-I-SUPERSEDE, library list superseded

$  CMS FETCH SAMPLE.PAS/OCCLUDE=NOELEMENT "fetch all instances"
%CMS-S-FETCHED, generation 1 of element DISKX:[WORK.CMSLIB]SAMPLE.PAS fetched
%CMS-I-FILEXISTS, file already exists, DISKX:[WORK]SAMPLE.PAS;2 created
%CMS-S-FETCHED, generation 1 of element DISKX:[PROJECT.CMSLIB]SAMPLE.PAS fetched
```

This command produces two copies of SAMPLE.PAS—the latest generation from library [WORK.CMSLIB], and the latest generation from library [PROJECT.CMSLIB].

Note that first CMS fetches the first instance of the file SAMPLE.PAS and then fetches the second instance, which creates a newer version of SAMPLE.PAS.

3.3.1 Occlusion of Multiple Object Types in a Command

Many CMS commands allow you to specify more than one object type on a command line. For instance, you can specify an element specification consisting of a list of element names and group names, as in the following example:

```
$  CMS FETCH CODE,BUILD.COM "fetch all code and the build procedure"
```

In this example, the group CODE represents all program source modules, and the element BUILD.COM is the build procedure to compile and link the program.

You can also specify a command in which objects of one type are inserted into or removed from objects of a different type. For example:

```
$  CMS INSERT ELEMENT MAIN.BAS CODE "insert main module into CODE group"
```

This command inserts the element MAIN.BAS into the group CODE.

When you specify multiple object types in a CMS command, CMS simultaneously performs occlusion on all applicable objects. Specifically, in the preceding example, CMS simultaneously performs occlusion on the elements and the groups, and ignores occlusion for other object types.

A special case occurs when you use a group name as an element specification. In this case, the elements in the group occlude subsequent instances of those elements (if element occlusion is enabled). In such cases, CMS performs element occlusion even if the specification contained only group names.

The following two examples show the difference between using a group name as an element specification and using a comma-separated list of the same element names that are in the group. In these examples, the default directory is [WORK], the current library search list is set to [WORK.CMSLIB],[PROJECT.CMSLIB]; the group SAMPLES is in [PROJECT.CMSLIB] and contains the elements SAMPLE.PAS and SAMPLE.DAT. The library [WORK.CMSLIB] does not contain any groups, but does contain the same elements. The examples correspond with Figure 3–2.

```
CMS> FETCH SAMPLES
_Remark:  fetch 1st instance of element generations from group SAMPLES
%CMS-I-FETCHED, generation 2 of element DISKX:[PROJECT.CMSLIB]SAMPLE.DAT fetched
%CMS-I-FETCHED, generation 2 of element DISKX:[PROJECT.CMSLIB]SAMPLE.PAS fetched
%CMS-I-FETCHES, 2 elements fetched
```

```
CMS>  FETCH/OCCLUDE=NOELEMENT SAMPLES
_Remark:  fetch all instances of element generations from group SAMPLES
%CMS-I-FETCHED, generation 2 of element DISKX:[PROJECT.CMSLIB]SAMPLE.DAT fetched
%CMS-I-FETCHED, generation 2 of element DISKX:[PROJECT.CMSLIB]SAMPLE.PAS fetched
%CMS-I-FETCHES, 2 elements fetched
```

In this case, the group SAMPLES is used as the element specification. Note that the /OCCLUDE qualifier has no effect; two elements are fetched in each case. Although the two elements SAMPLE.DAT and SAMPLE.PAS exist in the first library [WORK.CMSLIB], they are not fetched because CMS looks for the elements in group SAMPLES, which is in the second library [PROJECT.CMSLIB]. Since there are no further occurrences of SAMPLES, the two elements in the second library are fetched.

```
CMS>  FETCH SAMPLE.DAT,SAMPLE.PAS
_Remark:  fetch first instance of sample elements
%CMS-I-FETCHED, generation 1 of element DISKX:[WORK.CMSLIB]SAMPLE.DAT fetched
%CMS-I-FETCHED, generation 1 of element DISKX:[WORK.CMSLIB]SAMPLE.PAS fetched
%CMS-I-FETCHES, 2 elements fetched

CMS>  FETCH/OCCLUDE=NOELEMENT SAMPLE.DAT,SAMPLE.PAS
_Remark:  fetch all instances of sample elements
%CMS-I-FILEXISTS, file already exists, DISKX:[WORK]SAMPLE.DAT;2 created
%CMS-I-FETCHED, generation 1 of element DISKX:[WORK.CMSLIB]SAMPLE.DAT fetched
%CMS-I-FILEXISTS, file already exists, DISKX:[WORK]SAMPLE.DAT;2 created
%CMS-I-FETCHED, generation 1 of element DISKX:[WORK.CMSLIB]SAMPLE.PAS fetched
%CMS-I-FILEXISTS, file already exists, DISKX:[WORK]SAMPLE.DAT;3 created
%CMS-I-FETCHED, generation 1 of element DISKX:[PROJECT.CMSLIB]SAMPLE.DAT fetched
%CMS-I-FILEXISTS, file already exists, DISKX:[WORK]SAMPLE.PAS;3 created
%CMS-I-FETCHED, generation 1 of element DISKX:[PROJECT.CMSLIB]SAMPLE.PAS fetched
%CMS-I-FETCHES, 4 elements fetched
```

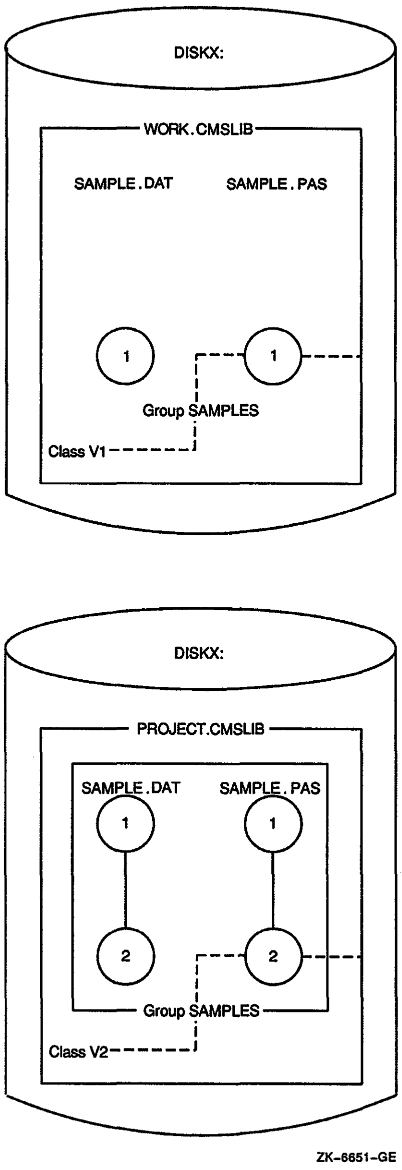
In the first command, CMS assumes /OCCLUDE=ALL, and fetches only the first instance of each of the elements SAMPLE.DAT and SAMPLE.PAS. In the second command, because the /OCCLUDE=NOELEMENT qualifier was specified, CMS fetches all occurrences of each element from both libraries.

Thus, an element specification consisting of a group containing a set of elements, and an element specification consisting of a list of the same set of elements, are not equivalent. In the first case, CMS locates the first instance of the group. Previous instances of the elements in the group may exist in an earlier library, but are not selected because they are not located in the library with the specified group. In the second case, the first instance of each of the specified elements is found, regardless of which library they may be in. In fact, they may be found in libraries in the list prior to the library in which the group is found.

3.3.2 Examples

The following examples show how you can control occlusion on various CMS objects. For the following examples, assume the library is set to [WORK.CMSLIB],[PROJECT.CMSLIB]. The library [WORK.CMSLIB] contains the two elements SAMPLE.DAT and SAMPLE.PAS with generation 1 of the element SAMPLE.PAS inserted into class V1. The library [PROJECT.CMSLIB] contains the two elements SAMPLE.DAT and SAMPLE.PAS. These two elements are inserted into group SAMPLES. Generation 2 of element SAMPLE.PAS is inserted into class V2. Figure 3–2 matches the following examples.

Figure 3–2: Library Occlusion



1.
```
$ CMS FETCH SAMPLE.PAS "fetch the first instance"
%CMS-S-FETCHED, generation 1 of element DISKX:[WORK.CMSLIB]SAMPLE.PAS fetched
%CMS-S-FETCHES, 1 element fetched
```

In this example, CMS assumes the default value of the /OCCLUDE qualifier (/OCCLUDE=ALL), and fetches only the first instance of SAMPLE.PAS.

2.
```
$ CMS FETCH/OCCLUDE=NOELEMENT SAMPLE.PAS "fetch all instances"
%CMS-I-FILEXISTS, file already exists, DISKX:[WORK]SAMPLE.PAS;2 created
%CMS-S-FETCHED, generation 1 of element DISKX:[WORK.CMSLIB]SAMPLE.PAS fetched
%CMS-I-FILEXISTS, file already exists, DISKX:[WORK]SAMPLE.PAS;3 created
%CMS-S-FETCHED, generation 2 of element DISKX:[PROJECT.CMSLIB]SAMPLE.PAS fetched
%CMS-S-FETCHES, 2 elements fetched
```

In this example, because the /OCCLUDE qualifier is specified with the negated keyword NOELEMENT, CMS retrieves all (both) instances of SAMPLE.PAS. Note that the second instance of SAMPLE.PAS is fetched into the next higher version of the output file, which is then placed into your default directory.

3.
```
$ CMS FETCH SAMPLE.PAS/GENERATION=V1 "default occlusion"
%CMS-S-FETCHED, generation 1 of element DISKX:[WORK.CMSLIB]SAMPLE.PAS fetched
%CMS-S-FETCHES, 1 element fetched

$ CMS FETCH SAMPLE.DAT/GENERATION=V1 "SAMPLE.DAT not in class V1"
%CMS-E-NOFETCH, error fetching element DISKX:[WORK.CMSLIB]SAMPLE.DAT
-CMS-E-GENNOTFOUND, generation V1 of DISKX:[WORK.CMSLIB]SAMPLE.DAT not found
%CMS-E-ERRFETCHES, 0 elements fetched and 1 error occurred

$ CMS FETCH SAMPLES/GENERATION=V1
_Remark:     SAMPLES group not in 1st library where class V1 is located
%CMS-E-NOFETCH, error fetching element SAMPLES
-CMS-E-NOTFOUND, Group SAMPLES not found
%CMS-E-ERRFETCHES, 0 elements fetched and 1 error occurred

$ CMS FETCH SAMPLE.PAS/GENERATION=V2 "element found but not class"
%CMS-E-NOFETCH, error fetching element DISKX:[WORK.CMSLIB]SAMPLE.PAS
-CMS-E-GENNOTFOUND, generation V2 of DISKX:[WORK.CMSLIB]SAMPLE.PAS not found
-CMS-E-NOTFOUND, Class DISKX:[WORK.CMSLIB]V2 not found
%CMS-E-ERRFETCHES, 0 elements fetched and 1 error occurred

$ CMS FETCH/OCCLUDE=NOELEMENT SAMPLE.PAS/GENERATION=V2
_Remark:  element found but not class
%CMS-E-NOFETCH, error fetching element DISKX:[WORK.CMSLIB]SAMPLE.PAS
-CMS-E-GENNOTFOUND, generation V2 of DISKX:[WORK.CMSLIB]SAMPLE.PAS not found
-CMS-E-NOTFOUND, Class DISKX:[WORK.CMSLIB]V2 not found
%CMS-S-FETCHED, generation 2 of element DISKX:[PROJECT.CMSLIB]SAMPLE.PAS fetched
%CMS-E-ERRFETCHES, 1 element fetched and 1 error occurred)
```

This example shows occlusion when multiple object types are present. Classes V1 and V2 exist in the first and second libraries, respectively.

Note that in the last case, an error diagnostic is generated when the first instance of SAMPLE.PAS is found and the class V2 is not found; then when both the element and the class are found, the specified generation is successfully fetched.

4.
```
$ CMS VERIFY SAMPLE.DAT/OCCLUDE=NOELEMENT
%CMS-I-VERCLS, class list verified
%CMS-I-VERCMD, command list verified
%CMS-I-VERELE, element list verified
%CMS-I-VERGRP, group list verified
%CMS-I-VERRES, reservation list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERARC, archive control block verified
%CMS-I-VER2, internal contiguous space verified
%CMS-I-VERCON, control file verified
%CMS-I-VEREDF, element DISKX:[WORK.CMSLIB]SAMPLE.DAT verified
%CMS-I-VEREDFS, element data files verified
%CMS-S-VERIFIED, library DISKX:[WORK.CMSLIB] verified
%CMS-I-VERCLS, class list verified
%CMS-I-VERCMD, command list verified
%CMS-I-VERELE, element list verified
%CMS-I-VERGRP, group list verified
%CMS-I-VERRES, reservation list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERARC, archive control block verified
%CMS-I-VER2, internal contiguous space verified
%CMS-I-VERCON, control file verified
%CMS-I-VEREDF, element DISKX:[PROJECT.CMSLIB]SAMPLE.DAT verified
%CMS-I-VEREDFS, element data files verified
%CMS-S-VERIFIED, library DISKX:[PROJECT.CMSLIB] verified
```

In this example, because the /OCCLUDE qualifier is specified with the negated keyword NOELEMENT, CMS verifies both libraries and both instances of the element SAMPLE.DAT.

3.4 Library Locking

CMS allows multiple read operations in the library at the same time. Read operations are operations that do not change any information in the library; for example, ANNOTATE, SHOW commands, SET LIBRARY, FETCH without a remark, and DIFFERENCES. Read operations allow users to use any combination of these commands in the library without interfering with each other in any way.

CMS controls concurrent access to the library by using the VMS locking mechanism. The locking mechanism does not allow multiple write or multiple read and write operations in the library at the same time. Write operations are operations that change information in the library; for example, CREATE, FETCH with a remark, INSERT, MODIFY, REMOVE, SET ACL, RESERVE, and REPLACE.

When CMS has locked the library during a write operation, any access attempts made by other users are not allowed until the write operation is complete.

If the library remains locked for an extended period, CMS periodically issues messages informing you that the library is still in use. Your command is processed as soon as the lock preventing your library access is released.

When you enter a command, CMS attempts to lock the library only for the appropriate type of access for that command. If no other locks prevent your lock from being granted, you gain immediate access to the library, and your command is processed. If the library is locked for an access type incompatible with that required by your command (for example, a user entered a SHOW GENERATION command that locks the library for read access, and you enter a REPLACE command that locks the library for write access), CMS informs you that the library is locked and issues the following error message:

```
%CMS-I-INUSE, library library-specification is in use, please wait
```

CMS processes read and write attempts on the library in order. For example, assume that user A has currently entered a command causing a library lock for read access (for example, a SHOW GENERATION command). User B enters a command requiring write access (for example, a REPLACE command), thus causing CMS to lock out user B. User C then enters a command requiring read access (for example an ANNOTATE command). CMS will not process user C's command until user B's command has been processed, even though the current library lock (user A's read lock) allows user C's command to gain access to the library. This prevents a chain of compatible lock requests from locking an incompatible lock request out of the library for a prolonged period of time.

If your command cannot gain access to the library after 15 minutes, the waiting loop expires and CMS issues a message requesting that you reattempt the command again later. You can use CTRL/C at any point to abort the command.

Chapter 4
Elements and Generations

A CMS library is a collection of files, which represent elements and element generations. An *element* is the basic structural unit in a library. An element consists of one file and all its versions, called *generations*. This chapter discusses elements and their generations in detail.

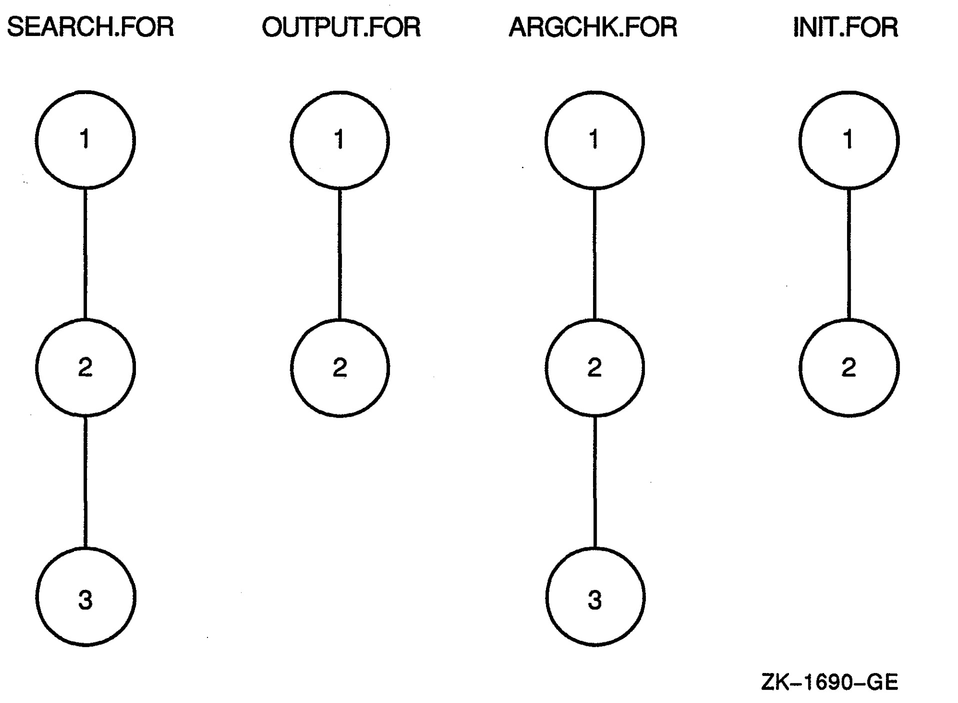
4.1 The Relationship Between Elements and Generations

When you place a file in a CMS library for the first time, CMS uses that file to create an element; that file becomes generation 1 of that element. An element generation represents a specific version of an element. Each time you reserve and replace an element in the library, CMS creates a new generation. CMS can store multiple generations of an element.

CMS assigns a permanent generation number to each new generation. This number is unique for each generation of a particular element. A CMS generation number is not the same as a version number that VMS assigns to files; file version numbers have no significance to CMS.

Figure 4–1 shows four elements and their generations in a simple CMS library.

Figure 4–1: Elements and Their Generations



This library contains three generations of the element SEARCH.FOR. The first generation was created with the CREATE ELEMENT command. Then, a generation of the element was reserved from and replaced into the library twice, creating generations 2 and 3. Similarly, the library contains two generations of OUTPUT.FOR, three generations of ARGCHK.FOR, and two generations of INIT.FOR. CMS stores the entire text of the first generation of an element. Then, in successive generations, CMS stores only the lines that change from one generation to the next (see Section 4.4 and Appendix C for more information).

The following example shows how to reserve a generation of an element named INIT.FOR and replace it in the CMS library, thereby creating a new generation:

```
$ CMS RESERVE INIT.FOR "change block header offset"
%CMS-S-RESERVED, generation 2 of element DISKX:[PROJECT.CMSLIB]INIT.FOR reserved
   .
   .
   .
$ CMS REPLACE INIT.FOR "header offset and additional free space added"
%CMS-S-GENCREATED, generation 3 of DISKX:[PROJECT.CMSLIB]element INIT.FOR created
```

The RESERVE command retrieves the latest main-line generation of element INIT.FOR, which is generation 2. The file is created in your current default directory and generation 2 is marked as reserved. The REPLACE command returns the contents of the file to the CMS library and assigns the next number in sequence to the new generation. Because generation 2 was reserved, the replacement transaction creates generation 3. See Section 4.2.3 and Section 4.2.4 for more information on the RESERVE and REPLACE commands, respectively.

4.2 Manipulating Elements and Generations

The following sections describe how to create, fetch, reserve, replace, monitor, display, and delete elements and generations in a CMS library.

4.2.1 Creating Elements and Generations

You create an element with the CREATE ELEMENT command. Each time you reserve and replace a generation of an element, you create a new generation of that element (see Section 4.2.4).

In the CREATE ELEMENT command, you specify the name and type of the file that is to become the name of the element. Within a library, all element names must be unique. The file-name component cannot be 00CMS because that name is reserved for CMS. Specify the file with the following syntax:

filename.type

For example:

```
$ CMS CREATE ELEMENT INIT.FOR "initialization routines"
%CMS-S-CREATED, element DISKX:[PROJECT.CMSLIB]INIT.FOR created
```

This command creates an element named INIT.FOR from the file INIT.FOR. CMS searches for the file named INIT.FOR in your default directory; use the /INPUT qualifier on the CREATE ELEMENT command to specify a different location, a different file name, or both.

When an element is created, CMS deletes all versions of the file in your default directory used to create the new element. Use the /KEEP or /RESERVE qualifier to prevent CMS from deleting any files.

4.2.2 Fetching an Element Generation

The FETCH command copies the contents of an element generation into a file. Unlike the RESERVE command, FETCH does not mark an element generation as reserved, and you cannot replace a fetched copy in the library. You can fetch a copy of a generation of an element whether or not the element is reserved.

For example, to retrieve a copy of the latest main-line generation of an element named TIMTST.COM, type the following command:

```
$ CMS FETCH TIMTST.COM
_Remark:  Testing storage blocks
%CMS-S-FETCHED, generation 2 of element DISKX:[PROJECT.CMSLIB]TIMTST.COM fetched
```

CMS retrieves the latest main-line generation of the element TIMTST.COM (generation 2). To retrieve an earlier generation (or variant generation; see Chapter 6), specify the /GENERATION qualifier on the FETCH command.

CMS creates a file with the same name as the fetched element, and places this file in your current default directory; use the /OUTPUT qualifier on the FETCH command to specify a different file name, a different location, or both.

4.2.3 Reserving an Element Generation

After creating an element in a CMS library, use the RESERVE command to retrieve a copy of a generation of the element to make changes to it. When you reserve a generation, CMS retrieves a copy of the generation of the element and marks that generation as reserved in the CMS library. You can then edit, compile, test, and debug the file as necessary. Most of your work with the CMS library consists of reserving library element generations for work and replacing the modified files back into the library as new generations.

When you reserve a generation of an element, CMS creates a file with the same name as the element, and places this file in your current default directory; use the /OUTPUT qualifier on the RESERVE command to specify a different location, a different file name, or both.

CMS prompts you to enter a remark when you reserve an element generation. You should use the remark to explain why you are reserving the generation; the remarks provide a permanent record of your work.

For example, to reserve the element SYNCHRON.BAS, use the following command:

```
$  CMS RESERVE SYNCHRON.BAS
_Remark:  losing sample from one data line
%CMS-S-RESERVED, generation 2 of element DISKX:[PROJECT.CMSLIB]SYNCHRON.BAS reserved
```

CMS copies the element generation into a file that is created in your current default directory. CMS marks the generation with your reservation and records the transaction in the library transaction history.

CMS retrieves the latest main-line generation of the element SYNCHRON.BAS (generation 2). To reserve an earlier generation (or a variant generation; see Chapter 6), specify the /GENERATION qualifier on the RESERVE command. For example:

```
$  CMS RESERVE SYNCHRON.BAS/GENERATION=1
_Remark:  Commenting data line sampling code
%CMS-S-RESERVED, generation 1 of element DISX:[PROJECT.CMSLIB]SYNCHRON.BAS reserved
```

A copy of the first generation of the element is then placed in the current default directory.

CMS allows you to concurrently reserve more than one generation of the same element, or the same generation more than once. If any generation of an element is already reserved (by you or another person) CMS issues a message about the reservation already in effect. You then have the option to proceed with your reservation or to quit. If you choose to proceed with the reservation, the element is considered to have concurrent reservations. While you have an element generation reserved, any user who reserves or attempts to reserve a generation of the same element receives a CMS message indicating that you have reserved the element generation. See Section 4.3.2 for more information on concurrent reservations.

If you reserve an element generation and then decide not to modify it, you can cancel your reservation with the UNRESERVE command. CMS records the cancellation in the library history. CMS does not modify the library element and does not create a new generation of the element when you cancel a reservation. The UNRESERVE command is useful if you reserve a wrong element, or if you do not want your modifications to become part of the element. For example, to unreserve a generation of the element named SYNCHRON.BAS, type the following:

```
$  CMS UNRESERVE SYNCHRON.BAS
_Remark:  element not applicable--wrong file
%CMS-S-UNRESERVED, element DISKX:[PROJECT.CMSLIB]SYNCHRON.BAS unreserved
```

Normally, CMS allows you to unreserve only your own reservation. However if you hold BYPASS privilege or if an access control entry (ACE) on the element grants you BYPASS access, you can cancel any reservation of that element held by another user. The cancellation of the reservation is then logged in the history file under your name. See Chapter 7 for more information.

4.2.4 Replacing an Element Generation

After modifying a reserved generation, use the REPLACE command to replace the latest version of the modified file into the library. CMS then deletes all copies of that file from your directory (unless you specify the /KEEP or /RESERVE qualifier), assigns a new CMS generation number to the newly created element generation, terminates your reservation, and records the transaction. For example:

```
$ CMS RESERVE SPEC.RNO
_Remark:  Misspelling in Reliability section
%CMS-S-RESERVED, generation 3 of element DISKX:[PROJECT.CMSLIB]SPEC.RNO reserved
     .
     .
     .
$ CMS REPLACE SPEC.RNO
_Remark:  Reliability section typo fixed
%CMS-S-GENCREATED, generation 4 of element DISKX:[PROJECT.CMSLIB]SPEC.RNO created
```

In this example, the current generation (generation 3) is reserved to correct a typographical error, and then replaced.

To avoid creating a new generation if the input file has no changes from the reserved generation, use the /IF_CHANGED qualifier on the REPLACE command. See the description of the REPLACE command in the Command Dictionary for more information.

Normally, CMS allows you to replace only your own reservation. However, if you hold BYPASS privilege or if an access control entry (ACE) on the element grants you BYPASS access, you can replace any reservation of that element held by another user. This mechanism allows you to designate a single person who is responsible for reviewing and entering all changed reservations into the library, for example. See Chapters 7 and 8 for more information on ACEs.

CMS allows you to concurrently reserve more than one generation of the same element, or the same generation more than once. When you replace the generations that are concurrently reserved by you, you must specify the /GENERATION or /IDENTIFICATION_NUMBER qualifier on the REPLACE command. See Section 4.3.3 for information on replacing concurrent reservations.

4.2.5 Monitoring Element Changes

You can monitor changes made to elements by using the notification access control entry (ACE) or the **review** attribute.

The notification ACE allows you to specify a list of people to be notified when particular events occur in a CMS library. See Chapter 8 for more information on CMS notification.

The **review** attribute allows you to specify that newly created element generations are to be placed on a review pending list. You can then associate review remarks with a generation under review. To assign the **review** attribute, use the /REVIEW qualifier on either the CREATE ELEMENT or the MODIFY ELEMENT command. The **review** attribute specifies that any new generations of that element are marked as pending review. You can also mark a specific generation for review by using the MARK GENERATION command. To determine which generations have reviews pending, use the SHOW REVIEWS_PENDING command.

When you review a generation, you can accept or reject the generation, cancel the review, or enter review comments. See Section 4.5.4 and the descriptions of the following commands in the Command Dictionary for more information on review:

```
ACCEPT GENERATION
CANCEL REVIEW
MARK GENERATION
REJECT GENERATION
REVIEW GENERATION
SHOW REVIEWS_PENDING
```

4.2.6 Displaying Information About Elements and Generations

You can view information about elements and generations in a CMS library with the SHOW commands. The SHOW ELEMENT command displays information about some or all of the elements in the current library. For example:

```
$  CMS SHOW ELEMENT
Elements in CMS Library DISKX:[PROJECT.CMSLIB]

ADCONVERT.BAS "analog to digital conversion routines"
ERRMSG.TXT    "initial load"
SAMPLE.BAS    "Sampling module"
SPEC.RNO      "ADS functional specification"
SYNCHRON.BAS  "Synchronization routines"
TIMTST.COM    "Command procedure for tests"
```

In this case, SHOW ELEMENT displays an alphabetical list of all the elements in the project library [PROJECT.CMSLIB], along with their remarks. You can also use the SHOW ELEMENT/MEMBER command to display the element name, creation remark, and the name of any groups to which the element belongs.

If you need information about a specific generation of an element, use the SHOW GENERATION command. If you omit a generation number, CMS assumes the last generation on the main line of descent. If you include a generation number, specify it with the /GENERATION qualifier. For example:

```
$  CMS SHOW GENERATION SYNCHRON.BAS/GENERATION=3

Element generations in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

SYNCHRON.BAS    3    26-JUN-1988 09:44:12 KELLEY "a/d conversion integrated"
```

This command displays the characteristics of generation 3 of the element SYNCHRON.BAS.

To discover which generation of an element is in a particular class, use the SHOW GENERATION command and specify the class name as the generation expression. For example:

```
$  CMS SHOW GENERATION/GENERATION=BASELEVEL1 SYNCHRON.BAS

Element generations in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

SYNCHRON.BAS    3    26-JUN-1988 09:44:12 KELLEY "a/d conversion integrated"
```

This command displays the generation of element SYNCHRON.BAS that is in class BASELEVEL1.

Whenever you create, reserve, or replace a library element, CMS stores information about the transaction in the library's history file. The SHOW HISTORY command allows you to review a chronological list of all library transactions. For example:

```
$  CMS SHOW HISTORY

History of CMS Library DISKX:[PROJECT.CMSLIB]

 2-MAY-1988 14:22:16 WHIPPLE CREATE LIBRARY DISKX:[PROJECT.CMSLIB] "a/d data
       sampling library"
 2-MAY-1988 14:26:47 MARTIN CREATE ELEMENT SPEC.RNO "ADS functional
       specification"
 8-JUN-1988 12:09:02 WHIPPLE CREATE ELEMENT ADCONVERT.BAS "analog to digital
       conversion routines"
 8-JUN-1988 12:25:41 WHIPPLE CREATE ELEMENT SAMPLE.BAS "Sampling module"
 8-JUN-1988 12:29:24 HENRY CREATE ELEMENT SYNCHRON.BAS "Synchronization
       routines"
 8-JUN-1988 14:01:36 HENRY CREATE ELEMENT TIMTST.COM "Command procedure for
       tests"
 9-JUN-1988 14:47:40 DAVIS RESERVE SYNCHRON.BAS(1) "losing sample from one
       data line"
```

CMS does not record transactions that do not alter the library. CMS logs FETCH transactions only if you supply a remark.

You can use the SHOW HISTORY command with the /UNUSUAL qualifier to report any abnormal library transactions that occurred, such as two reservations in effect for the same element at the same time.

The SHOW RESERVATIONS command lists element generations that are currently reserved (or by identification number, if the element is concurrently reserved), who reserved each generation, when it was reserved, and why it was reserved. For example:

```
$  CMS SHOW RESERVATIONS

Reservations in CMS Library DISKX:[PROJECT.CMSLIB]

SAMPLE.BAS
  (1)  JIMK     1      30-JUN-1988 11:19:29 "add code for more data lines"
SYNCHRON.BAS
  (1)  KELLEY   3      18-JUN-1988 09:42:03 "integrate a/d conversion"
```

You can use the SHOW GROUP/CONTENTS command to display the contents of a group. For example:

```
$  CMS SHOW GROUP/CONTENTS TIME_TST

Groups in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

TIME_TST           "comparison testing prototype group"
    SYNCHRON.BAS
    TIMTST.COM
```

This command lists the elements contained in group TIME_TST.

4.2.7 Deleting Generations

To delete one or more generations of an element from the library, use the DELETE GENERATION command. This command is useful if you replaced a wrong version of a file into the CMS library, or if you want to remove old generations of elements. This command permanently removes information about a generation from the corresponding element in the library. If the latest main-line generation is deleted, the next latest main-line generation is placed into the reference copy directory. Deleting a generation does not remove changes from subsequent generations that were originally made in the deleted generation and thus exist in subsequent generations.

Deleting unneeded generations allows operations that access generations (for example, FETCH, REPLACE, and RESERVE) to complete faster because the number of generations to be searched is reduced (see Section 9.3.2 for more information).

When you delete an element generation, you can optionally use the /ARCHIVE qualifier to direct CMS to create an archive file. When you specify /ARCHIVE, CMS creates a file containing all the information from the deleted generation, and places it in your default directory. See the description of DELETE GENERATION/ARCHIVE in the Command Dictionary and Section 9.3.2 for more information.

4.3 Concurrency

This section describes how CMS organizes concurrent changes to library elements and how to resolve conflicting changes to those elements. A concurrent change occurs when two or more people work on an element at the same time and make separate changes to the element.

If you cannot avoid making a concurrent reservation, be aware that some additional effort is involved when you replace concurrent reservations. The following sections describe how to reserve a generation of an element that has prior reservations, and replace the reservation into the library.

4.3.1 Concurrent Access

CMS allows you to control concurrent access to an element by using the **concurrent** attribute. You define the **concurrent** attribute by specifying the /[NO]CONCURRENT qualifier.

You can prohibit concurrent access by specifying the /NOCONCURRENT qualifier on the CREATE ELEMENT command for a new element, or by using the MODIFY ELEMENT command to change the attribute of an existing element. You cannot modify concurrent access to an element while a generation of the element is reserved. When you prohibit concurrent access to an element, only one reservation of the element is allowed at a time until you use the MODIFY ELEMENT/CONCURRENT command to allow concurrent access.

You can temporarily prohibit concurrent access for the duration of a reservation by specifying the /NOCONCURRENT qualifier on the RESERVE command. If you reserve a generation of an element in this way, you must replace it or cancel the reservation (with the UNRESERVE command) before you or anyone else can reserve any generation of the element.

4.3.2 Concurrent Reservations

If a generation of the element you want to reserve is already reserved and concurrent access is not prohibited, CMS accepts your RESERVE command and the remark you enter with it, but warns you that an element generation is currently reserved and by whom, and prompts you for confirmation before proceeding. If you continue with the reservation, the element is then marked as being concurrently reserved, and it retains that status until all reservations of the element are ended. For example:

```
$ CMS RESERVE BASTEST.GNC
_Remark:  reserving for final production
Element BASTEST.GNC currently reserved by:
     (1)    DAVIS      3     28-JAN-1988  09:27:46  "for testing"
Proceed?  [Y/N] (N):
```

If you type NO or press RETURN, CMS does not execute the reserve transaction. If you type YES, CMS places a copy of the generation in your current default directory, marks the element as concurrently reserved, and records the reservation transaction in the library history. CMS records the transaction as an unusual occurrence. For information about unusual occurrences, see Chapter 9.

CMS allows multiple reservations by a single user; that is, you can reserve more than one generation of the same element, and you can also reserve the same generation more than once. CMS assigns a unique identification number to each reserved generation. The identification number appears first on each line. Use the SHOW RESERVATIONS command to determine the identification number of each reservation. You must use the /IDENTIFICATION_NUMBER qualifier to replace a concurrent reservation (see Section 4.3.3).

4.3.3 Concurrent Replacements

When a concurrent reservation ends (when you replace the element generation with the REPLACE command), it is called a *concurrent replacement.* When you replace an element that another user has concurrently reserved, CMS reports that a prior concurrent reservation was made and specifies who the second reserver was, even if the second reserver has already replaced the element. For example:

```
$  CMS REPLACE BASTEST.GNC
_Remark:  replacing after completing edits
Concurrent replacements
      (1)    DAVIS      2     28-JAN-1988  09:27:46  "for testing"
Proceed?  [Y/N] (N):
```

If you type NO or press RETURN after the Proceed? prompt, CMS does not execute the replacement transaction. If you type YES, CMS proceeds with the command and records the transaction as an unusual occurrence. For information about unusual occurrences, see Chapter 9.

At least one reserver must replace a concurrent reservation as a variant generation. You replace the concurrent reservation as a variant generation by specifying the /VARIANT qualifier on the REPLACE command. This begins a variant line of descent. Either user can then merge the variant generation back into the original line so that both sets of program modifications appear in one generation, or the variant line of descent can be continued (see Chapter 6 for more information).

CMS allows you to concurrently reserve a specific generation more than once. When you replace the generations that are concurrently reserved by you, you must specify which reservation is to be replaced. You can do this with either the /GENERATION qualifier or the /IDENTIFICATION_NUMBER qualifier on the REPLACE command.

You can use /GENERATION as long as the concurrent reservations are not on the same generation. If you have more than one concurrent reservation for the same generation, you must identify the specific reservation to be replaced. Each reservation is assigned an identification number. Use the SHOW RESERVATIONS command to determine the identification number of each reservation. The identification number appears first on each line. If you use the /IDENTIFICATION_NUMBER qualifier, you do not need to also use the /GENERATION qualifier. For example:

```
$  CMS REPLACE BASTEST.GNC/IDENTIFICATION_NUMBER=2
_Remark:  replacing after completing edits
Element BASTEST.PAS currently reserved by:
      (1)    DAVIS      3     28-JAN-1988  09:27:46  "for testing"
Proceed?  [Y/N] (N):
```

In this example, the /IDENTIFICATION_NUMBER qualifier specifies that the second reserved generation be replaced into the CMS library. CMS reports other existing reservations you hold for that element (in this case, the first reserved generation), and then prompts you to proceed.

You must also use the /GENERATION or /IDENTIFICATION_NUMBER qualifier if you are replacing another user's reservation. See the description of the REPLACE command in the Command Dictionary and Section 7.3 for more information.

4.4 Delta Files

For each element stored in the library, CMS maintains a *delta file*—a single file containing a representation of the contents of all of the generations of that element.

In addition to the actual data, the delta file contains control records. Control records tell CMS which data records are valid for which specific generations of the element. When you retrieve a generation of an element, CMS includes records that are valid and excludes records that are not valid for that generation.

One of the effects of the delta file method of storing information is that retrieval times are consistent within a given element. For example, it takes a similar amount of time to fetch generation 100 of an element or generation 1 of that same element. Another effect is that generation deletion does not necessarily produce a significantly smaller delta file, because records that are valid in a generation being deleted may also be valid (and, in fact, are likely to be valid) in later or earlier generations that are not being deleted.

See Appendix C for more information on how CMS stores library information.

4.5 Element Attributes

The CREATE ELEMENT and MODIFY ELEMENT commands allow you to specify the following element attributes:

- The **concurrent** attribute controls whether concurrent reservations of an element are allowed (See Section 4.3.2 for more information).
- The **history**, **notes**, and **position** attributes allow you to manipulate the format of historical information that is associated with an element.
- The **reference copy** attribute directs CMS to maintain a reference copy of an element.

- The **review** attribute directs CMS to mark newly created generations of an element as pending review.

You can use the SHOW ELEMENT/FULL command to display the current settings of these attributes (see the Command Dictionary for more information).

4.5.1 The History Attribute

When the **history** attribute is defined for an element, CMS includes the element generation history in the output file when you retrieve a generation of an element from the library with the FETCH or RESERVE command. This history is a list of the transactions that created each generation of the element. Each transaction record consists of the generation number, user, date, time, and remark associated with the generation.

Use the /HISTORY qualifier to define the **history** attribute for an element. You can either establish the **history** attribute when the element is created with the CREATE ELEMENT command, or change the **history** attribute of an existing element with the MODIFY ELEMENT command. You can cancel the **history** attribute by using the /NOHISTORY qualifier on the MODIFY ELEMENT command. You can also specify the /[NO]HISTORY qualifier on the FETCH and RESERVE commands to temporarily override the element's **history** attribute.

The format of the /HISTORY qualifier is as follows:

/HISTORY="string" /NOHISTORY

"string"
Specifies the format of each line of the element history. The string must contain exactly one occurrence of the history format parameter, can contain only printing ASCII characters and the space and tab characters, and must begin and end with a quotation mark ("). The history format parameter consists of a number sign (#) followed by an uppercase or lowercase letter H or B. For example:

```
"!#H"
"/*#b*/"
```

The exclamation point (!) and the slash-asterisk characters (/* */) indicate comments.

Use the letter B to direct CMS to include the history at the beginning of the file and H to include the history at the end of the file. The history text is inserted into the string wherever the #H or #B history format parameter occurs. To include a number sign (#) in the string, type it twice (##). To include a quotation mark in the string, type it twice (""). If the file contains source code, you must include comment indicators or delimiters applicable to your source code in the string so that the program can be compiled or assembled. The history is then treated as a comment.

NOTE

Because of Record Management Services (RMS) record storage restrictions, CMS cannot include history text in files with fixed-length records. If you try to fetch or reserve a generation of an element that has history enabled and the generation has fixed-length records, you receive the following message:

```
%CMS-I-NOHISTNOTES, history and notes will not be included in output file
```

The history includes a line for each generation of an element. Each line consists of the text contained in the quoted string, with #H or #B replaced by the creation information for that generation. The history region is delimited by the following line:

```
DEC/CMS REPLACEMENT HISTORY, Element element-name
```

This line allows the REPLACE command to distinguish the history from the rest of the file when it is returned to the library. CMS does not consider history text to be part of the file. Instead, the history is added to the file when it is retrieved from the library and removed when the file is replaced into the library. The generation numbers of a retrieved generation and its ancestors are marked with an asterisk (*).

Do not insert or modify text in the history section while editing a file in your directory. CMS expects only history lines between the two header lines. The REPLACE command reports an error if it finds any other text where the history should be, and the command is not executed. You must then delete the extra text with an editor and reenter the REPLACE command.

The following command establishes the **history** attribute for a file that contains a Pascal program:

```
$  CMS MODIFY ELEMENT SEMANTICS.PAS/HISTORY="{#H}" "est. history attribute"
```

When the default generation of SEMANTICS.PAS is retrieved with the FETCH or RESERVE command, the history at the end of the program looks like this:

```
{ DEC/CMS REPLACEMENT HISTORY, Element SEMANTICS.PAS }
{ *6    28-JUL-1988 10:00:54 EDGAR   "formal parameter list support added"}
{ *5    24-JUL-1988 16:10:14 DAVIS   "actual parameter list support added"}
{ *4    20-JUL-1988 12:22:07 MARTIN  "preliminary work on routine calls done"}
{ 3C1   17-JUL-1988 12:15:45 JEFF    "error checking on CASE statement works"}
{ *3    11-JUL-1988 11:57:18 MALER   "CASE statement support"}
{ *2     7-JUL-1988 11:56:05 HENRY   "FOR loop support done"}
{ *1     9-JUN-1988 18:11:25 BARRETT "semantic analysis module"}
{ DEC/CMS REPLACEMENT HISTORY, Element SEMANTICS.PAS }
```

The braces ({}) indicate comments in Pascal. Because the braces surround the lines of the history, the history lines are ignored by the Pascal compiler. The history is delimited with the header line. Each existing generation of the element is listed. The generation numbers of the specified generation and its ancestors are marked with an asterisk. Generation 6 was retrieved; therefore, that generation and its ancestors are marked with an asterisk. Generation 3C1 is not an ancestor because it is on a variant line of descent.

NOTE

Some language processors do not accept a file that has data after the formal end of the program. If you use #H in the definition of the **history** attribute for an element, the element file may not be compatible with these processors. If this occurs, you can specify the /NOHISTORY qualifier with the RESERVE and FETCH commands. When you use this qualifier, CMS does not include the history in the file that is placed in your directory. Also, because CMS wraps history lines at 132 characters, you can use the /NOHISTORY qualifier with history lines that are longer than 132 if your file is to be used by a processor or compiler that does not accept 132-character lines.

See Section 4.5.5 for an example of using the **history** attribute.

4.5.2 The Notes and Position Attributes

When the **notes** and **position** attributes are defined for an element, CMS appends notes to lines of the file when you retrieve a generation of an element from the library with the RESERVE or FETCH command. A note appears on every line that has been modified since generation 1 as close to the position specified by the **position** attribute as possible. Notes can be one or both of the following:

- Generation numbers indicating the latest generation in which the line was inserted or modified
- ASCII text contained in the quoted string parameter of the /NOTES qualifier

You use the /NOTES qualifier to define the **notes** attribute, and the /POSITION qualifier to define the column in which the note should start. You can establish these attributes when the element is created with the CREATE ELEMENT command, or you can change the attributes of an existing element with the MODIFY ELEMENT command. Any element that has the **notes** attribute must have the **position** attribute and vice versa. Use the /NONOTES qualifier on the MODIFY ELEMENT command to cancel both attributes.

You can also specify the /[NO]NOTES and /POSITION qualifiers on the FETCH and RESERVE commands to temporarily override the element's **notes** and **position** attributes.

The format of the /NOTES qualifier is as follows:

```
/NOTES="string" /NONOTES
```

"string"
Specifies the format of the notes. The string can contain only ASCII characters; it must begin and end with a quotation mark ("). The notes string cannot exceed 100 characters. The string can optionally contain one occurrence of the notes format parameter. The notes format parameter consists of a number sign (#) followed by an uppercase or lowercase letter G. For example:

```
"!#G"
"/*#g*/"
```

The exclamation point (!) and the slash-asterisk characters (/* */) indicate comments.

To include a number sign (#) in the string, type it twice (##). To include a quotation mark in the string, type it twice (""). If the file contains source code, you must include comment indicators applicable to your source code in the string so that the program can be compiled or assembled. The notes are then treated as comments. See Section 4.5.5 for an example.

A note for a line consists of the text contained in the quoted string with the notes parameter replaced by the number of the generation in which the line was inserted or most recently modified.

NOTE

Because of Record Management Services (RMS) record storage restrictions, CMS cannot include notes text in files with fixed-length records. If you attempt to fetch or reserve a generation of an element that has notes enabled and the generation has fixed-length records, you receive the following message:

```
%CMS-I-NOHISTNOTES, history and notes will not be included in output file
```

A note for a line appears at the position specified by the /POSITION qualifier. The /POSITION qualifier is required when /NOTES is specified.

The format of the /POSITION qualifier is as follows:

/POSITION=n

n
Specifies the character position at which the notes are to appear on the line. The position value must be an integer in the range 1 to 511.

The note is placed to the right of the text of the line. If the length of the line is less than n, the note begins at position n. If the length of the line is greater than or equal to n, the note begins at the next tab stop after the end of the text of the line. (Tab stops are at position 9 and at every eight characters thereafter.)

CMS does not consider notes to be part of the element generation. Instead, notes are added to the file when it is retrieved from the library and removed when the file is replaced into the library. If, while editing the file, you add text after the note or within the note, CMS does not recognize it as a note and therefore replaces it as part of the generation. If you add text that looks like a note, CMS interprets it as a note and removes it before replacing the file.

See Section 4.5.5 for an example of using the notes and position attributes.

4.5.3 The Reference Copy Attribute

An element reference copy is a copy of the latest main-line generation of an element. CMS maintains reference copies of the latest generations of selected library elements in a nonlibrary directory.

If you have established a reference copy directory for a library, each newly created element is automatically set with the /REFERENCE_COPY qualifier. New elements inherit the **reference copy** attribute from the library setting.

When the **reference copy** attribute is enabled for an element, CMS creates a reference copy by fetching a copy of the latest main-line generation into the reference copy directory. If, for any reason, the reference copy directory cannot be updated, CMS does not create the new generation.

You can use the /REFERENCE_COPY qualifier to define the **reference copy** attribute for a single element. You can establish the **reference copy** attribute when the element is created with the CREATE ELEMENT command, or you can change the **reference copy** attribute of an existing element with the MODIFY ELEMENT command. You can prevent CMS from creating a reference copy by specifying the /NOREFERENCE_COPY qualifier with the CREATE ELEMENT or MODIFY ELEMENT command.

The format of the /[NO]REFERENCE_COPY qualifier is as follows:

```
/REFERENCE_COPY /NOREFERENCE_COPY
```

See Section 3.1.4 and the /[NO]REFERENCE_COPY qualifier in the Command Dictionary for more information on reference copies.

4.5.4 The Review Attribute

When the **review** attribute is enabled for an element, CMS places any newly created generations of that element on the review pending list, and marks them for review. You can associate review remarks with a generation under review by using the REVIEW GENERATION command (see the Command Dictionary).

You use the /REVIEW qualifier to define the **review** attribute for an element. You can establish the **review** attribute when the element is created with the CREATE ELEMENT command, or you can change the **review** attribute of an existing element with the MODIFY ELEMENT command. You can cancel the **review** attribute by using the /NOREVIEW qualifier on the MODIFY ELEMENT command.

The format of the /[NO]REVIEW qualifier is as follows:

```
/REVIEW /NOREVIEW
```

To determine what generations are under review, use the SHOW REVIEWS_PENDING command, which also shows any review comments. Once a generation is under review, a user trying to retrieve that generation with a FETCH command is informed that a review is pending. If you retrieve the generation with the RESERVE command, you are informed that a review is pending and are prompted for confirmation to continue. The messages are issued until the generation's review status is resolved. A generation with a review pending cannot be deleted.

You can resolve a generation's review status in one of three ways: accept the generation with the ACCEPT GENERATION command, cancel the review with the CANCEL REVIEW command, or reject the generation with the REJECT GENERATION command. If you accept the generation or cancel the review, CMS halts review-related messages and confirmations on subsequent reservation attempts. If you reject the generation, CMS issues a message indicating that the generation was reviewed and rejected. The generation is still accessible so that the problems in it that caused the rejection can be corrected.

A generation created from a generation that currently has a review pending or that was previously rejected is automatically marked for review, regardless of the setting of the element's **review** attribute.

You can also use the MARK GENERATION and REVIEW GENERATION commands to mark a generation for review and to review the generation. For more information, see the descriptions of these commands in the Command Dictionary.

See Section 4.5.5 for an example of using the **review** attribute.

4.5.5 Examples of Using Element Attributes

The following example shows how to create the element RESV.REQ with the **history**, **notes**, and **position** attributes:

```
$ CMS CREATE ELEMENT RESV.REQ/HISTORY="!#H"/NOTES="!#G"/POSITION=80
_Remark:  Require file for multiple reservations
%CMS-S-CREATED, element DISKX:[PROJECT.CMSLIB]RESV.REQ created
```

This command creates an element called RESV.REQ. The element contains data structures written in the BLISS programming language. The /HISTORY qualifier specifies that history is to be appended to the file when it is retrieved from the library. Each line of the history is preceded by an exclamation point (!), which indicates a comment in the BLISS language. The /NOTES and /POSITION qualifiers specify that generation numbers are to be embedded in the lines of the file at position 80. The generation numbers are preceded by an exclamation point (!).

The history and notes are embedded in the file RESV.REQ when it is retrieved with the RESERVE or FETCH command. Alternatively, you can specify /NONOTES or /NOHISTORY with the FETCH or RESERVE command to direct CMS to omit the notes or history in the file.

In the following example, five generations of the file RESV.REQ exist and the element is retrieved with the FETCH command:

```
$  CMS FETCH RESV.REQ
_Remark:  take a look at history and notes specifications
%CMS-S-FETCHED, generation 5 of element DISKX:[PROJECT.CMSLIB]RESV.REQ fetched
```

The FETCH command retrieves generation 5 of the element. The file that is delivered to the user's directory is shown in Example 4–1.

Example 4–1: An Element with History and Notes Attributes

```
! RESV.REQ - Reservation control information values and structures. ❶

LITERAL
        REPL = TRUE,                ❶     !Replace flag
        NOTREPL = FALSE,            ❶     !NOT replace flag
        RES_MAX = 150,              ❶     !Maximum number of entries in list  ❷ !3
        TXT_MAX = 15000;            ❶     !Working buffer size limit (assuming❷ !3
                                    ❶     !an average reservation line             ❷ !2
                                    ❶     !length of 132 characters)          ❷ !2

MACRO
        ENT_SIZ = 4 %;              ❶     !Number of entries in field

FIELD
    RES_FLD =
        SET
        LINK_ADR = [0,0,%BPVAL,0],                                            ❷ !5
        STG_ADR = [1,0,%BPVAL,0],                                             ❷ !5
        STG_SIZ = [2,0,%BPVAL/2,0],
        REM_FLG = [2,%BPVAL/2,%BPVAL/2,0],                                    ❷ !4
        REP_MKR = [3,0,%BPVAL/2,0],
        CUR_RES = [3,%BPVAL/2,%BPVAL/2,0]
        TES;

! RESV.REQ Last Line  ❶
! DEC/CMS REPLACEMENT HISTORY, Element RESV.REQ  ❷
!*5 13-AUG-1988 10:49:12 MARTIN "transportability fixes"  ❷
!*4 30-JUL-1988 20:45:13 DAVIS "fix field boundary error"  ❷
!*3 15-JUL-1988 17:27:42 REYB "lower buffer size and reservation max"  ❷
!*2 30-JUN-1988 07:25:24 KIRK "change work buffer size and reservation max"  ❷
!*1 31-MAY-1988 12:01:17 DAVIS "Require file for multiple reservations"  ❷
! DEC/CMS REPLACEMENT HISTORY, Element RESV.REQ  ❷
```

❶ Indicates comments existing in the file

❷ Indicates comments supplied by CMS

The numbers (preceded by an exclamation point) to the right of the code denote the generation in which the lines were inserted or most recently modified. These notes start at position 80, except for the third note (generation 2). Because the line exceeds position 80, the third note begins at the next tab stop after the end of the line. Lines that have not been changed since generation 1 have no notes. The history starts and ends with the following title:

```
! DEC/CMS REPLACEMENT HISTORY, Element RESV.REQ.
```

The history shows the transactions that created each generation of the element.

If the /NONOTES and /NOHISTORY qualifiers had been specified on the FETCH command, the retrieved file would not have contained the embedded notes and history as shown in Example 4–1.

Example 4–2 shows the process of marking generations for review, displaying the list of generations in the library that are on the review pending list, rejecting a generation, and reserving a generation that has been rejected.

Example 4–2: Example of Using the Review Attribute

```
CMS> MARK GENERATION BASCHAP*.SDML
_Remark:  need to review chapters for BASIC manual
%CMS-I-MARKED, generation 1 of element DISKX:[PROJECT.CMSLIB]BASCHAP1.SDML
marked for review
%CMS-I-MARKED, generation 1 of element DISKX:[PROJECT.CMSLIB]BASCHAP2.SDML
marked for review
%CMS-I-MARKED, generation 1 of element DISKX:[PROJECT.CMSLIB]BASCHAP3.SDML
marked for review
%CMS-I-MODIFICATIONS, 3 modifications completed

CMS> SHOW REVIEWS_PENDING

Reviews pending in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

BASCHAP1.SDML
    DAVIS      1      28-JAN-1988 15:48:25 "creating Chapter 1 INTRO"

BASCHAP2.SDML
    DAVIS      1      28-JAN-1988 15:48:29 "creating Chapter 2 SYNTAX"

BASCHAP3.SDML
    DAVIS      1      28-JAN-1988 15:48:32 "creating Chapter 3 NEW FEATURES"

CMS> FETCH BASCHAP3.SDML
_Remark:  new features still applicable?
Generation 1 of element BASCHAP3.SDML has a review pending
%CMS-S-FETCHED, generation 1 of element DISKX:[PROJECT.CMSLIB]BASCHAP3.SDML fetched
     .
     .
     .
CMS> REJECT GENERATION BASCHAP3.SDML "new features made into separate
section, not an entire chapter"
%CMS-S-REJECTED, generation 1 of element DISKX:[PROJECT.CMSLIB]BASCHAP3.SDML rejected

CMS> RESERVE BASCHAP3.SDML "need to pull section"
Generation 1 of element BASCHAP3.SDML has been rejected
Proceed?  [Y/N]  (N):  YES
%CMS-S-RESERVED, generation 1 of element DISKX:[PROJECT.CMSLIB]BASCHAP3.SDML reserved
```

Chapter 5

Groups and Classes

This chapter describes how to create and use groups and classes.

5.1 Overview

Groups and classes are mechanisms that you can use to organize a CMS library. Both groups and classes are typically used in a library; although each mechanism creates a different library structure, both can be used in the same library without conflict.

5.1.1 Groups

A *group* is a collection of elements or other groups, or a combination of both. You combine one or more elements into a group that you can then manipulate as a single unit. For example, you might create a group that contains all the files that process error messages, a group that contains all the chapters and appendixes in a book, or a group that contains the modules needed to build a part of a database.

Even if an element is in a group, you can still manipulate the element as an object that is separate from the group. A group can also belong to one or more other groups. The only restriction is that a group cannot be a member of itself, that is, it cannot directly or indirectly be a subgroup of itself.

5.1.2 Classes

A *class* is a set of *specific generations* of elements that can be manipulated as a unit. A class can hold only one generation of any element.

You use classes to represent the state of development of a system or set of elements at a particular time or stage. You can think of a class as a picture taken of a library at a particular time. For example, you might create a class named FIRST_DRAFT that contains only those generations of elements that were used in producing the first draft of a manual.

Typically, you create a class to contain generations of all the components of a software system for a release version of a product. You can establish classes for different stages or milestones; for example, you could establish one class for implementation, a second for testing, and a third for generations that have completed the first two stages. As each module progresses through each stage, you assign each generation to an appropriate class; thus, you can easily determine your progress by displaying the contents of the different classes, and you can later reconstruct any stage of development.

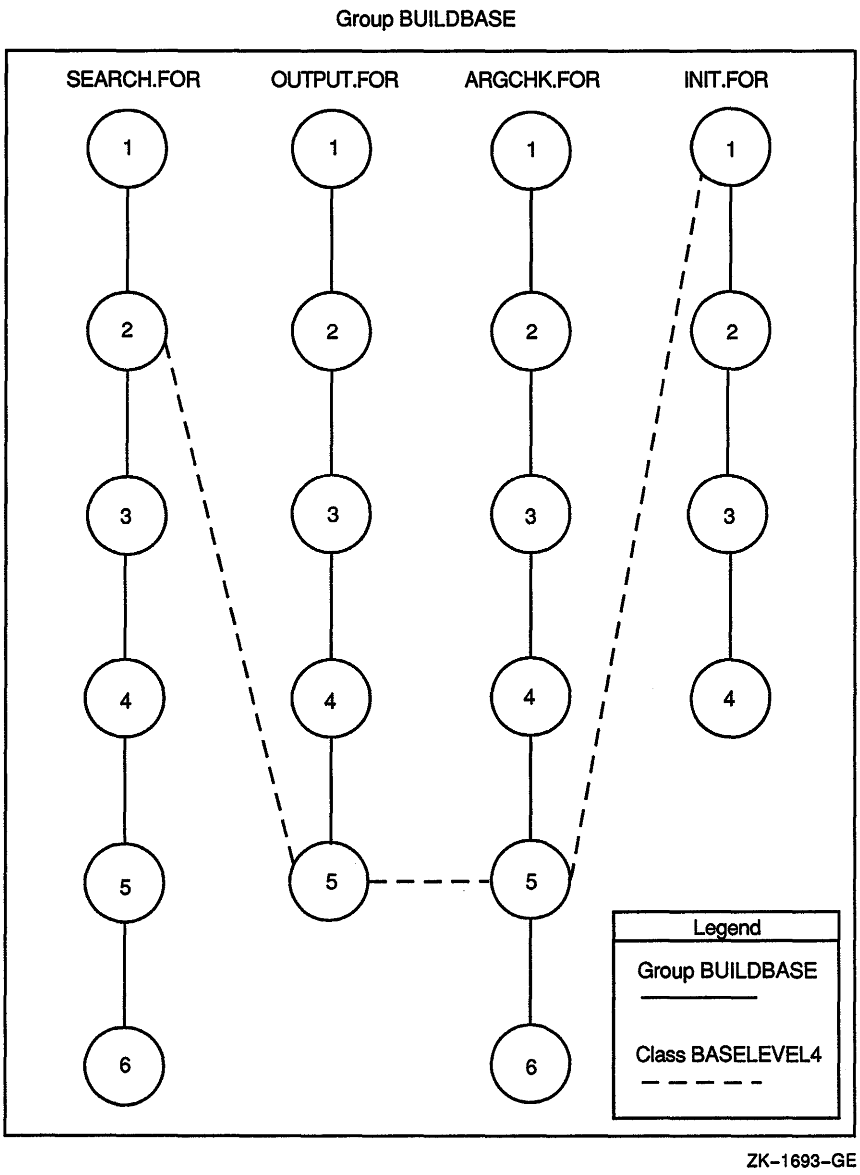
Once you insert an element generation into a class, further changes made to the element are not reflected in the contents of that class.

5.1.3 The Difference Between Groups and Classes

When you use groups, you manipulate elements. A group is an entity that allows you to give a name to a set of elements in the library and manipulate the set of elements with that name. You typically use groups to associate elements together; for example, you could create a group containing all the art figures in a manual, or a group containing all source modules that contain callable entry points.

When you use classes, you manipulate specific generations of elements. A class is an entity that allows you to give a name to a set of specific generations of elements in the library and manipulate the set with that name. In contrast to groups, classes contain only one generation from an element. You typically use classes to take a "timed snapshot" of a set of generations; that is, the generations that are meaningful to a project at a particular time. For example, you could create a class containing the specific generations that are included in a code freeze or field test kit, or a class containing the specific generations that make up the state of the project on some other significant date. Figure 5–1 shows the the relationship between a group and a class.

Figure 5–1: Groups and Classes



The circles in the figure represent four elements and their generations. The number in each circle represents the generation number of the element generation. These four elements and their respective generations are contained in group BUILDBASE, a group containing the modules needed to build part of a database.

If you retrieve group BUILDBASE, you receive the latest generation on the main line of descent of each of the following elements in the group:

SEARCH.FOR, generation 6
OUTPUT.FOR, generation 5
ARGCHK.FOR, generation 6
INIT.FOR, generation 4

The dashed line that connects element generations represents class BASELEVEL4. Class BASELEVEL4 contains the element generations that comprise the state of the library on March 12, the date the project moved to base level 4.

If you retrieve class BASELEVEL4, you receive the following element generations:

SEARCH.FOR, generation 2
OUTPUT.FOR, generation 5
ARGCHK.FOR, generation 5
INIT.FOR, generation 1

5.2 Manipulating Groups

The following sections describe how to create and use groups.

5.2.1 Creating Groups

Groups can contain elements, other groups, or a combination of both. You establish an empty group with the CREATE GROUP command. For example:

```
$  CMS CREATE GROUP USER_MANUAL "user documentation"
%CMS-S-CREATED, group DISKX:[PROJECT.CMSLIB]USER_MANUAL created
```

This command creates an empty group named USER_MANUAL.

5.2.2 Inserting Elements into Groups

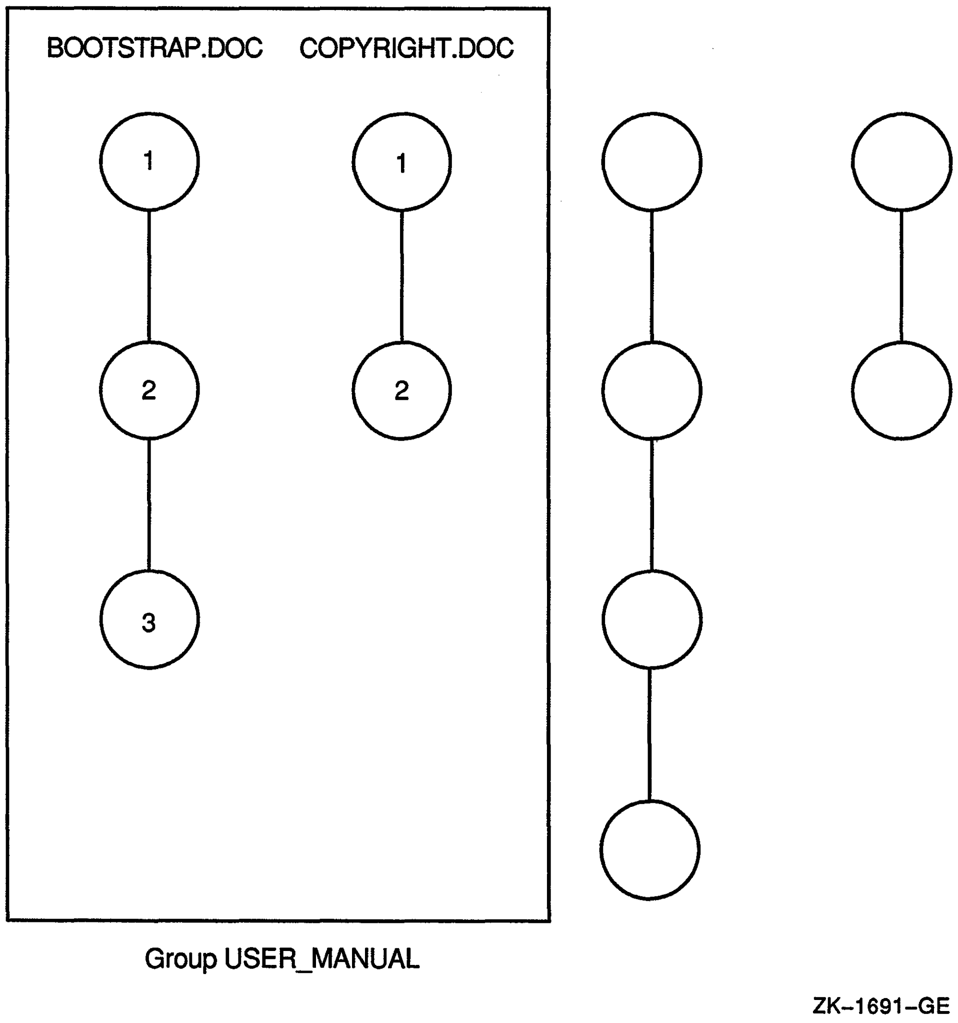
After you establish a group, you place one or more elements in the group with the INSERT ELEMENT command.

The following command inserts the elements COPYRIGHT.DOC and BOOTSTRAP.DOC into the group named USER_MANUAL:

```
$ CMS INSERT ELEMENT COPYRIGHT.DOC,BOOTSTRAP.DOC USER_MANUAL
_Remark:  copyright page
%CMS-I-INSERTED, element DISKX:[PROJECT.CMSLIB]COPYRIGHT.DOC inserted into
group DISKX:[PROJECT.CMSLIB]USER_MANUAL
%CMS-I-INSERTED, element DISKX:[PROJECT.CMSLIB]BOOTSTRAP.DOC inserted into
group DISKX:[PROJECT.CMSLIB]USER_MANUAL
%CMS-S-INSERTIONS, 2 insertions completed
```

Figure 5–2 shows the group USER_MANUAL, which contains two elements, BOOTSTRAP.DOC and COPYRIGHT.DOC.

Figure 5–2: Generations in a Group



This figure shows that all generations of the two elements are associated with the group. Therefore, you can access any generation of the elements in a group.

The element expression specified on the INSERT ELEMENT command can be one or more element names, group names, or a wildcard expression (for information about element expressions, see Section 10.2.4). If you specify a group name with the INSERT ELEMENT command, CMS enters the names of all of the elements in that group into the destination group. For instance, if you use INSERT ELEMENT to insert the contents of group A into group B, the contents of group B are not affected by any subsequent changes of the contents of group A.

You can also use the INSERT GROUP command to insert groups (and, thus indirectly, elements) into a group. For example:

```
$ CMS INSERT GROUP USER_MANUAL CODE_AND_DOCS
%CMS-S-INSERTED, group DISKX:[PROJECT.CMSLIB]USER_MANUAL inserted into
DISKX:[PROJECT.CMSLIB]group CODE_AND_DOCS
```

This command inserts the group USER_MANUAL into the group CODE_AND_DOCS. The INSERT GROUP command enters the group name USER_MANUAL into the list of entries for the group CODE_AND_DOCS. If the contents for the group USER_MANUAL change, the elements accessible through CODE_AND_DOCS also change.

5.2.3 Retrieving and Removing Elements from a Group

After you create a group and insert elements or other groups into that group, you can retrieve all generations of elements in the group with a single FETCH or RESERVE command. For example:

```
$ CMS FETCH USER_MANUAL "copy for internal sites"
%CMS-I-FETCHED, generation 4 of element DISKX:[PROJECT.CMSLIB]BOOTSTRAP.DOC fetched
%CMS-I-FETCHED, generation 1 of element DISKX:[PROJECT.CMSLIB]COPYRIGHT.DOC fetched
%CMS-S-FETCHES, 2 elements fetched
```

When you enter the FETCH command, CMS places a copy of the latest generation on the main line of descent of each element belonging to the group named USER_MANUAL into your current default directory.

When you retrieve a group of elements, by default, you get the latest generation on the main line of descent of each element in the group. By using the /GENERATION qualifier, you can gain access to a specific generation. Note that when you use the /GENERATION qualifier with groups, the generation expression is applied across the group. Thus, if you were to fetch a group of elements and you specified /GENERATION=2, CMS would retrieve the second generation of each element in the group.

The REMOVE ELEMENT command allows you to remove an element from a group; however, it does not alter or delete the element itself. For example:

```
$  CMS REMOVE ELEMENT SPEC.RNO DOCUMENTATION
_Remark:  User's manual ready for first review
%CMS-S-REMOVED, element DISKX:[PROJECT.CMSLIB]SPEC.RNO removed from group
DISKX:[PROJECT.CMSLIB]DOCUMENTATION
```

This command removes the element SPEC.RNO from the group DOCUMENTATION.

You can also use the REMOVE GROUP command to remove groups from other groups. For example:

```
$  CMS REMOVE GROUP USER_MANUAL CODE_AND_DOCS "removing group"
%CMS-S-REMOVED, group DISKX:[PROJECT.CMSLIB]USER_MANUAL removed from group
DISKX:[PROJECT.CMSLIB]CODE_AND_DOCS
```

This command removes the group USER_MANUAL from the group CODE_AND_DOCS. However, CMS does not delete or alter the groups being removed.

5.2.4 Displaying the Group Structure of a Library

To find out what groups are defined in your library, use the SHOW GROUP command. CMS lists group names in alphabetical order with the remark that is associated with the group. To obtain a list of all elements and groups in a specific group, use the SHOW GROUP command with the /CONTENTS qualifier. For example, to display the contents of the group named DATA_ROUTINES, you would type the following command:

```
$  CMS SHOW GROUP/CONTENTS DATA_ROUTINES

Groups in CMS Library DISKX:[PROJECT.CMSLIB]

DATA_ROUTINES "routines for input & conversion"
    ADCONVERT.BAS
    SAMPLE.BAS
```

5.2.5 Deleting Groups

The DELETE GROUP command deletes one or more groups from a CMS library. The group must not contain any elements. For example:

```
CMS>  DELETE GROUP TIME_TST "superseded by comparison tests"
%CMS-S-DELETED, group DISKX:[PROJECT.CMSLIB]TIME_TST deleted
```

This command deletes the group named TIME_TST.

If the group is not empty, or if it belongs to another group, CMS returns an error and does not delete the group. Use the REMOVE ELEMENT or REMOVE GROUP command to remove elements or groups from the group before entering the DELETE GROUP command.

5.3 Manipulating Classes

The following sections describe how to create and use classes.

5.3.1 Creating Classes

You establish an empty class with the CREATE CLASS command. For example:

```
$ CMS CREATE CLASS INTERNAL_RELEASE "for use in-house only"
%CMS-S-CREATED, class DISK:[PROJECT.CMSLIB]INTERNAL_RELEASE created
```

This command creates a class called INTERNAL_RELEASE. The class does not yet contain any element generations.

5.3.2 Inserting Element Generations into Classes

You place an element generation into a class with the INSERT GENERATION command.

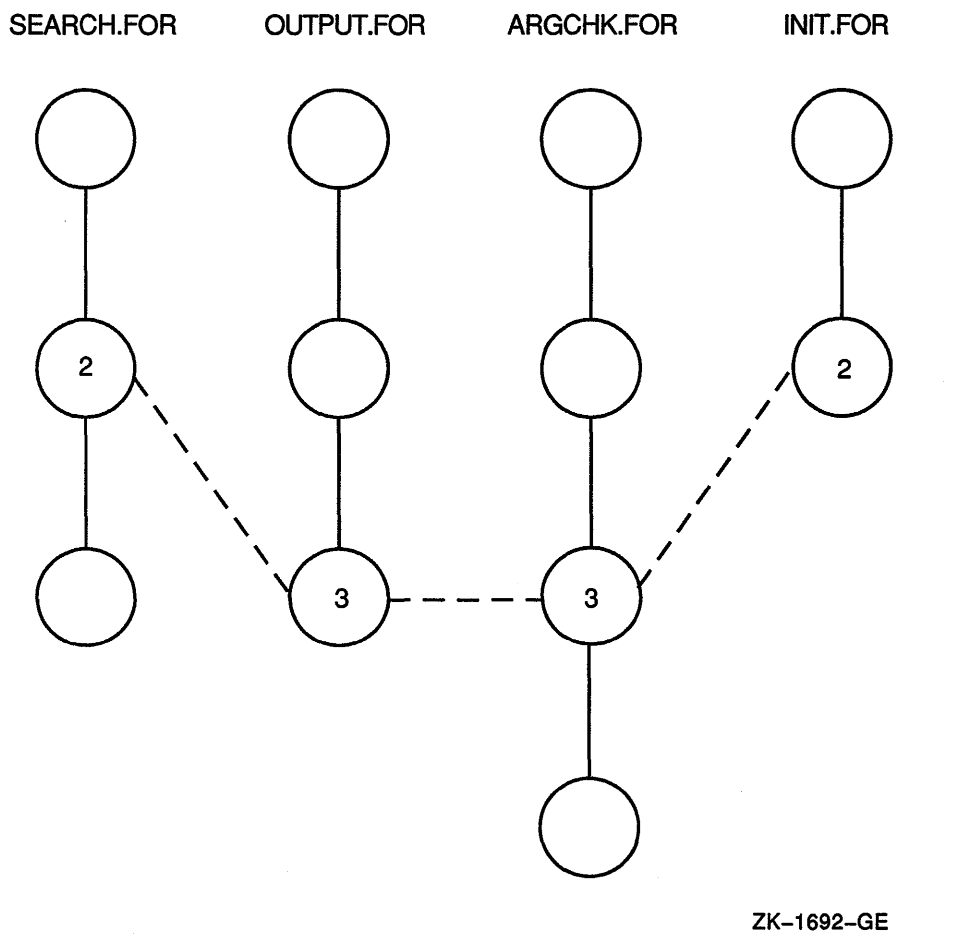
The following commands place generations of INIT.FOR and ARGCHK.FOR into the class INTERNAL_RELEASE:

```
CMS> INSERT GENERATION INIT.FOR INTERNAL_RELEASE
_Remark:  Initialization routine for demo
%CMS-S-GENINSERTED, generation 2 of element DISKX:[PROJECT.CMSLIB]INIT.FOR
inserted in class DISKX:[PROJECT.CMSLIB]INTERNAL_RELEASE
CMS> INSERT GENERATION ARGCHK.FOR/GENERATION=3 INTERNAL_RELEASE
_Remark:  Demo semantic analyzer
%CMS-S-GENINSERTED, generation 3 of element DISKX:[PROJECT.CMSLIB]ARGCHK.FOR
inserted in class DISKX:[PROJECT.CMSLIB]INTERNAL_RELEASE
```

The INSERT GENERATION command uses the latest generation on the main line of descent unless you specify the /GENERATION qualifier. A class can contain no more than one generation of an element. A generation can belong to zero, one, or more classes.

Figure 5–3 shows the relationship of elements, generations, and the class INTERNAL_RELEASE.

Figure 5–3: The Relationship Between Groups and Elements



The class INTERNAL_RELEASE contains generation 2 of INIT.FOR, generation 2 of SEARCH.FOR, generation 3 of OUTPUT.FOR, and generation 3 of ARGCHK.FOR.

5.3.3 Retrieving and Removing Generations from a Class

You can retrieve an element generation from a class by specifying the class name on the /GENERATION qualifier on the FETCH or RESERVE command. A class can contain no more than one generation of an element; the class name specifies the generation of the element to be retrieved. For example:

```
$ CMS RESERVE SEARCH.FOR/GENERATION=INTERNAL_RELEASE
_Remark:  add support for alternate two-character graphics
%CMS-S-RESERVED, generation 2 of element DISKX:[PROJECT.CMSLIB]SEARCH.FOR reserved
```

This command reserves generation 2 of SEARCH.FOR, as that generation belongs to the class INTERNAL_RELEASE.

If a class is established to contain each version or base level of a system, you can accurately reconstruct any previous version of the system. For example, if the users of your system use version 1, the element generations that constitute version 1 could belong to the class VER1. If the software has changed because you are in the process of developing version 2 and a bug is reported in version 1, you can retrieve the generation of the element in which the bug appeared because you know that it belongs to class VER1.

The REMOVE GENERATION command allows you to remove an element generation from a class. For example:

```
$ CMS REMOVE GENERATION DISPLAY.BAS BASELEVEL1 "no longer needed"
```

In this example, a generation of the element DISPLAY.BAS is removed from class BASELEVEL1. CMS then revises information about BASELEVEL1 so that no generation of DISPLAY.BAS is included in the class.

5.3.4 Displaying the Class Structure of a Library

To find out what classes are defined in your library, use the SHOW CLASS command. CMS lists class names in alphabetical order with the remark that is associated with the class. To obtain a list of all generations in a specific class, use the SHOW CLASS command with the /CONTENTS qualifier. For example:

```
$ CMS SHOW CLASS/CONTENTS BASELEVEL1

Classes in CMS Library DISKX:[PROJECT.CMSLIB]
```

```
BASELEVEL1 "Specifying all generations for first base level"
    ADCONVERT.BAS      5
    DISPLAY.BAS        2
    SAMPLE.BAS         6
    SYNCHRON.BAS       4
```

This command displays all the elements and their generations in class BASELEVEL1.

5.3.5 Deleting Classes

The DELETE CLASS command deletes one or more classes from a CMS library. The class must not contain any element generations. For example:

```
$  CMS DELETE CLASS PRE_RELEASE "no longer necessary"
%CMS-S-DELETED, class DISKX:[PROJECT.CMSLIB]PRE_RELEASE deleted
```

This command deletes the class named PRE_RELEASE.

If any element generations belong to the class, CMS issues an error message and does not delete the class. Use the REMOVE GENERATION command to remove element generations from a class before entering the DELETE CLASS command.

5.4 Group and Class Attributes

You can change the name, the remark, and the **read-only** attribute of both groups and classes by using the MODIFY GROUP and MODIFY CLASS commands.

You can use the /NAME qualifier on the MODIFY GROUP command to change the name of a group that was created with the CREATE GROUP command. Similarly, you can use the /NAME qualifier on the MODIFY CLASS command to change the name of a class that was created with the CREATE CLASS command.

You can use the /REMARK qualifier on the MODIFY GROUP and MODIFY CLASS commands to specify a new remark to be substituted for the remark created with the CREATE GROUP and CREATE CLASS commands.

You can use the /READ_ONLY qualifier on the MODIFY GROUP and MODIFY CLASS commands to assign read-only access to groups or classes. A group or a class that is set to read-only access cannot be changed; you cannot insert or remove any items to or from the group or class. In addition, you cannot change the name of a group or a class that is set to read-only access.

The following example sets the group DIAGNOSTICS to read-only access:

```
$  CMS MODIFY GROUP DIAGNOSTICS/READ_ONLY
_Remark:  diagnostics for use with V2 compiler
```

After this command has been executed, the group cannot be altered. To change the group, use the /NOREAD_ONLY qualifier with the MODIFY GROUP command. Similarly, you can use the /READ_ONLY and /NOREAD_ONLY qualifiers with the MODIFY CLASS command to enable or disable modifications to a class.

In addition, you can use the SET ACL and SHOW ACL commands to specify and display access control lists for groups and classes (as well as for other CMS library objects). See Chapters 7 and 8 for more information.

Chapter 6

Variants and Merging

This chapter provides information on lines of descent, creating variant lines of descent, and merging element generations.

6.1 Lines of Descent

The line of descent for a specified generation consists of all ancestors and direct descendants of that generation. The main line of descent consists of generation 1 and its direct descendants (generation 2, generation 3, and so on). A variant line of descent contains one or more variant generations; for example, the line of descent for generation 3A1B2 consists of the following generations: 1, 2, 3, 3A1, 3A1B1, 3A1B3, and so on. Generation 1 is the beginning of every line of descent.

Some projects require alternate development paths, such as a trial development of a slightly different internal program structure, a change in the scope of an existing program, or a version to run on a different operating system. Variant generations are the mechanism that CMS uses to organize concurrent, parallel changes to a library element.

6.1.1 Creating a Variant Generation

To create a variant generation, use the /VARIANT=x qualifier on the REPLACE command. This creates a variant line of descent that CMS can distinguish from the main line of descent. The parameter x, called the variant letter, is any single alphabetic character (A through Z). If you enter the variant letter as a lowercase character, CMS converts it to uppercase. CMS copies the replaced file into the library and labels the variant generation by appending the variant letter and the number 1 to the generation number of the ancestor generation. For example, if you reserved generation 7 of an element named INIT.FOR, you could create a variant as follows:

```
$  CMS REPLACE INIT.FOR/VARIANT=T
_Remark:  Routine added for multi-user system
%CMS-S-GENCREATED, generation 7T1 of element DISKX:[PROJECT.CMSLIB]INIT.FOR created
```

The variant letter (in this case, T) identifies the new line of descent. The variant letter has no meaning to CMS; you can use it for mnemonic purposes. For instance, you can choose a variant letter that indicates the purpose of the variant line, such as F for fixes, E for enhancements, and so forth.

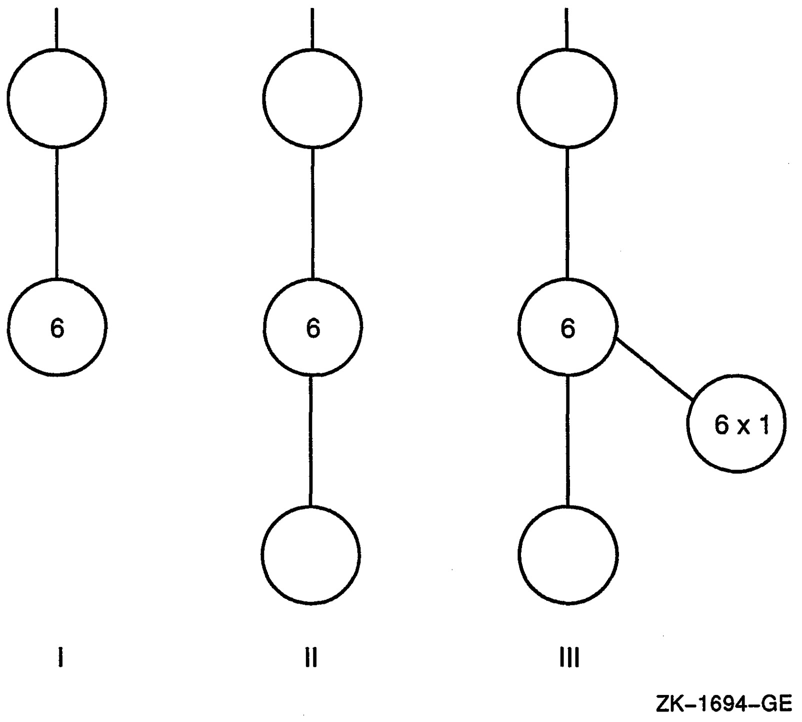
The number after the variant letter identifies successive generations on that new line of descent. For example, if you reserve and replace generation 7T1 of INIT.FOR, generation 7T2 is created. Each variant can have variants of its own using the same method; for example, you could replace a variant to generation 7T1 with the REPLACE/VARIANT=E command to create generation 7T1E1.

You can create a variant line for any reason; however, there are two cases in which you must create a variant in order to replace an element.

First, when two or more reservations are in effect for the same generation of the same element at the same time, all but one of the reservations must be replaced as a variant. CMS manages concurrent changes by allowing only one replacement to become the next generation on the same line of descent. The other replacements must begin variant lines of descent; the changes can then be merged back into the original line of descent (see Section 6.2.1).

Figure 6–1 shows one element at three different stages of development. In stage I, the element has six generations. At this point, two users reserve generation 6 of the element. One user replaces his reservation, creating generation 7 (stage II); the second user replaces her reservation, creating the variant generation 6X1 (stage III).

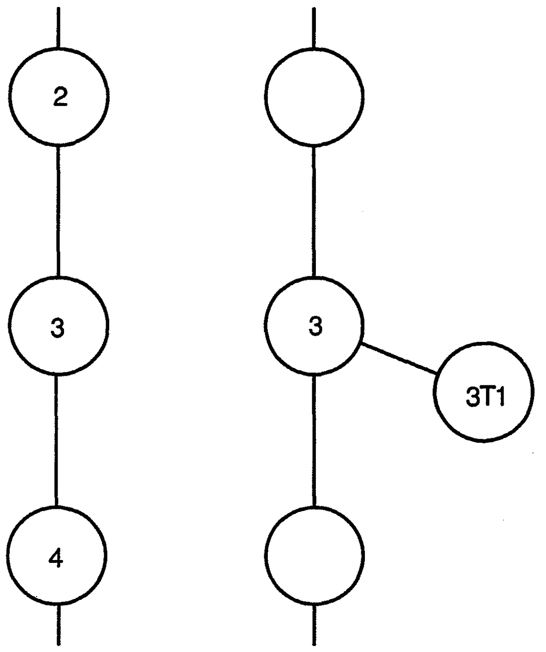
Figure 6–1: Creating a Variant Generation



Second, when you reserve a generation other than the most recent one on a line of descent, you must always create a variant successor because the successor on the same line of descent already exists. For example, if you reserved an earlier generation to modify software that has already been released, you must create a variant to store the modification. The change can then be merged into the original line of descent (see Section 6.2).

Figure 6–2 shows one element at two stages of development. If you reserve generation 3 of the element, you must create a variant (shown here as generation 3T1) when you replace the generation with the REPLACE command, because generation 4 already exists.

Figure 6–2: Extending a Variant Generation from an Earlier Generation



ZK–1695–GE

6.1.2 Accessing Variant Generations

Variant generation numbers can be used like any other generation numbers. You retrieve a variant generation of an element by using the /GENERATION qualifier with the FETCH or RESERVE command. You must specify a generation number or a class name when you use the /GENERATION qualifier. When you replace a reserved variant generation, the new generation is created on the same variant line. For example:

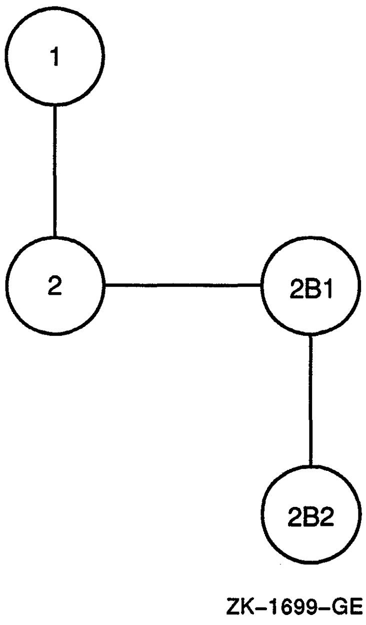
```
$  CMS RESERVE SEMANTICS.PAS/GENERATION=3C1
_Remark:  checks for multiple CASE labels
%CMS-S-RESERVED, generation 3C1 of element DISKX:[PROJECT.CMSLIB]SEMANTICS.PAS reserved
    .
    .
    .
(modify and test element file)
    .
    .
    .
$  CMS REPLACE SEMANTICS.PAS
_Remark:  error checking on multiple CASE labels done
%CMS-S-GENCREATED, generation 3C2 of element DISKX:[PROJECT.CMSLIB]SEMANTICS.PAS created
```

In this example, the /GENERATION qualifier on the RESERVE command specifies that generation 3C1 is to be reserved. The REPLACE command returns the element to the library and creates generation 3C2, which is on the same line of descent as its predecessor, 3C1.

6.1.3 Ancestor and Descendant Generations

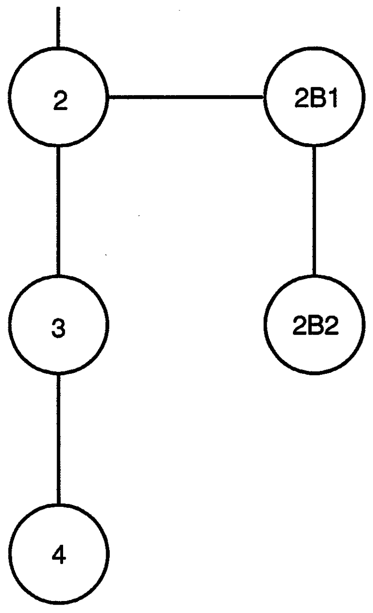
The ancestors of a generation on the main line of descent are all the preceding generations back to the first generation of the element (generation 1). The ancestors of a variant generation are all the preceding generations on the variant line of descent, which includes all generations on the path back to the first generation of the element. Figure 6–3 shows the path to the ancestors of generation 2B2.

Figure 6–3: Ancestors on a Tree of Descent



The descendants of a generation consist of all successive generations (on the same line of descent) and all their variant generations. Figure 6–4 shows the paths that connect the descendants of generation 2.

Figure 6–4: Descendants on a Tree of Descent



ZK–1700–GE

To display the ancestors or descendants of a generation, use the SHOW GENERATION command with the /ANCESTORS or /DESCENDANTS qualifier, respectively.

6.2 Merging Two Generations of an Element

At some point in the development cycle, you may want to combine changes made in two generations of an element. For instance, if concurrent changes are made to a generation of an element, those changes must be replaced as two separate generations, at least one of which must be a variant. The changes made in these new generations can now be merged into a single generation of the element.

Two conditions are necessary for a merge transaction:

- The generations must belong to the same element; that is, you cannot merge generations of different elements.

- One generation cannot be an ancestor of the other; that is, they must be on different lines of descent. For example, in Figure 6–4, you could merge generation 2B1 and 3 or 2B2 and 3, but you could not merge generations 2 and 3, or 2 and 2B1, or 2 and 2B2.

The following sections describe how to merge generations and how the merging process works.

6.2.1 Merging Element Generations

When you merge generations, CMS identifies the generation you specify with the /MERGE qualifier, the generation being fetched or reserved, and the common ancestor for the two generations. The common ancestor for the two generations is the most recent generation that is on both lines of descent. (Note that the two generations used in the merge transaction cannot be on the same line of descent.)

CMS then compares the changes that have been made in both generations being merged against the common ancestor. CMS looks for identical regions of text between each of the generations being merged and the common ancestor. These identical regions provide "anchor" points. The location of changes is determined relative to these anchor points—not relative to any particular line number. For example, CMS could consider line 200 in one generation being merged to be at the same location as line 500 in the other generation.

Any changes found in only one of the generations are included in the new file. These are called successful merges. Identical changes (modifications, insertions, deletions) made at identical locations in the merged generations are also included in the new file (also called successful merges). Different changes made at identical locations are flagged by CMS and require manual resolution. These changes are called merge conflicts. (Section 6.2.2 explains how conflicts between two generations are treated in the merging process.) CMS then creates an output file containing the results of the merge transaction, and places it in your current default directory. CMS assigns the current time as the creation and revision time of the output file; the output file does not inherit these values from the reserved generation.

NOTE

Because of Record Management Services (RMS) record storage restrictions, CMS does not merge element generations that have fixed-length records of different size. CMS does merge element generations that have fixed-length records with identical formats, however. If you try to merge fixed-length records of different size, you receive the following error messages:

```
CMS-E-SIZEMISMAT, cannot merge generations with different size records
CMS-E-GENRECSIZE, generation ## has ##-byte records, ## has ##-byte records
```

If at least one of the merged generations has variable-length records, no restrictions apply, and the resulting generation has variable-length records.

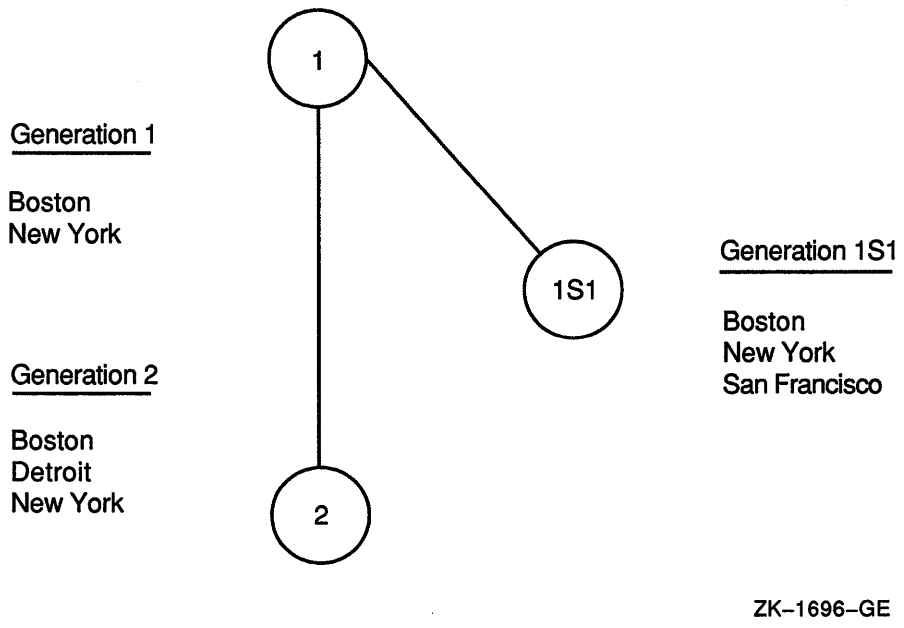
The /MERGE qualifier identifies the element generation that CMS merges into the generation being retrieved with the FETCH or RESERVE command.

For example, the following command merges generation 3A1 of the element DATACHAP.TXT into generation 7B3:

```
$  CMS FETCH DATACHAP.TXT/GENERATION=7B3/MERGE=3A1
```

Figure 6–5 shows the contents of three generations of the element CITY.TXT (generations 1, 2, and 1S1) and the relationship between the element generations.

Figure 6–5: The Relationship Between a Generation and an Element



In this example, generation 2 is the most recent on the main line of descent. Therefore, you can merge generations 2 and 1S1 with the following command:

```
$ CMS RESERVE CITY.TXT/MERGE=1S1
_Remark: merge generations 2 and 1S1
%CMS-I-MERGECOUNT, 2 changes successfully merged with no conflicts
%CMS-S-RESERVED, generation 2 of element DISKX:[PROJECT.CMSLIB]CITY.TXT reserved
and merged with generation 1S1
```

This command merges generation 1S1 into generation 2 of CITY.TXT. The output file (named CITY.TXT) contains the text common to both generations and the changes made to both generations. (The file is placed in your current default directory, or, if you use the /OUTPUT qualifier, another location.) CMS marks generation 2 of the element CITY.TXT as reserved. The generation indicated by the /MERGE qualifier (in this example, generation 1S1) is not reserved.

CMS determines the changes made in each of the generations being merged by comparing them against generation 1, which is the common ancestor. In this case, the changes were made to different parts of the file; thus, no conflicts exist. The resulting file looks like this:

```
BOSTON
DETROIT
NEW YORK
SAN FRANCISCO
```

The line DETROIT is the only difference between generation 1 and generation 2. This change occurs after the line BOSTON in the common ancestor. The line SAN FRANCISCO is the only difference between generation 1 and generation 1S1. This change occurs after the line NEW YORK in the common ancestor. Because the changes in generations 1S1 and 2 occur at different places in the common ancestor, both changes can be applied without conflict.

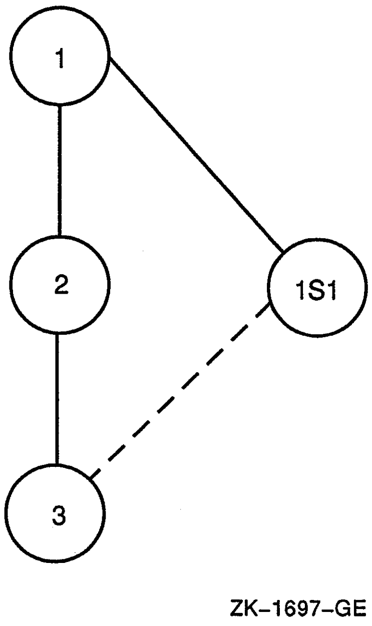
The merge transaction combines two lines of descent in a file outside the library. When you merge with the RESERVE command, you can subsequently replace the element in the library. The following command replaces the file created by merging generation 1S1 into generation 2 of CITY.TXT:

```
$ CMS REPLACE CITY.TXT "completed new format"
%CMS-S-GENCREATED, generation 3 of element DISKX:[PROJECT.CMSLIB]CITY.TXT created
```

The generation created by the replacement is a successor only to the generation that was reserved. Because generation 2 was specified as the retrieved generation when it was reserved, the REPLACE command creates generation 3.

Figure 6–6 shows the relationship of the generations of CITY.TXT after the replacement transaction. Note that no ancestor or line of descent relationship exists between generation 1S1 and generation 3.

Figure 6–6: A Generation After Replacement in the Library



If you do not want to create a new generation but want to produce a merged file, use the FETCH/MERGE command to merge two lines of descent. You can also use the ANNOTATE/MERGE command to create a single file that contains the text common to both generations and the changes made to both generations. See the ANNOTATE command in the Command Dictionary for more information.

For information on verifying the merge transaction, see Section 6.2.3.

6.2.2 Conflicts in the Merging Process

Different changes made at identical locations in a generation are called conflicting changes. A conflicting change can be one of the following:

- An insertion of one or more lines
- A deletion of one or more lines

- A replacement of n lines by m lines (n may or may not be equal to m)

If CMS detects conflicting changes in the merged generations, it notifies you by including the changes from both generations in the resulting file and surrounding them with asterisks.

Suppose that generation 2 of the element CITY.TXT contains an additional line of text and looks like the following:

```
BOSTON
DETROIT
NEW YORK
PORTLAND
```

Under these circumstances, the same merge transaction discussed in Section 6.2.1 produces different results:

```
$ CMS RESERVE CITY.TXT/MERGE=1S1
_Remark: merge two generations
%CMS-W-MERGECONFLICT, 1 change successfully merged with 1 conflict
%CMS-S-RESERVED, generation 2 of element DISKX:[PROJECT.CMSLIB]CITY.TXT reserved
and merged with generation 1S1
```

The resulting file looks like this:

```
BOSTON
DETROIT
NEW YORK
*************** Conflict 1     *************************************************
PORTLAND
********************************************************************************
SAN FRANCISCO
******** End of Conflict 1     *************************************************
```

When the two generations are merged, one change is successfully merged and one conflict exists. The line DETROIT from generation 2 is applied to the common ancestor without conflict. That is, there is no change from generation 1S1 in the same location. However, the line PORTLAND from generation 2 and the line SAN FRANCISCO from generation 1S1 both occur at the same location. Each conflict is flagged with the word "Conflict" and a sequential conflict number in a line of asterisks. (For files with short fixed-length records, CMS attempts to fit the "Conflict" label; if the Conflict label does not fit, CMS outputs only asterisks.) Following the asterisks, CMS displays the conflicting segments of text.

When CMS reports conflicts from a merge transaction, you must resolve conflicting lines with a text editor. For example, you may want to delete one set of changes.

NOTE

You must delete the conflict flags (the lines containing asterisks). If you do not delete them and the merged element is reserved, the REPLACE command replaces those lines into the library.

6.2.3 Verifying Merged Changes

The merging process is based solely on the text in the files being merged and is performed with no understanding of the meaning of that text. Thus, the resulting file from a "successful" merge may not have the desired form. For example, consider a document where both changes include the same paragraph, but at different places in the file. The successfully merged copy will contain a redundant paragraph. Or consider simultaneous changes made to a code module where one change deleted an unused routine while the other called that routine. The merged version would contain the call but no routine to be called, and yet the merge would be considered successful by CMS.

You should always verify that the merge transaction had the intended results. You can use the ANNOTATE/MERGE command to produce an annotated listing that shows all changes made to a file, or you can use the DIFFERENCES/FULL command to compare the contents of the files. If you use DIFFERENCES, you should perform the differences transaction three times: once against the new file and each of the merged generations (to ensure that their contents were preserved) and once against the new file and the common ancestor.

Also, because CMS does not understand the meaning of the text in the files being merged, where applicable you should always compile and link the file as a precautionary measure.

For more information on the ANNOTATE and DIFFERENCES commands, see the Command Dictionary.

Chapter 7

Security Features

You can use two types of security mechanisms to protect your CMS library and the objects in your library:

- Standard VMS file protection mechanisms based on user identification codes (UICs) and access control lists (ACLs)
- CMS ACLs

You use VMS file protection mechanisms to control access to VMS files and directories. In general, UIC-based protection is useful for denying or granting access to a user or group of users (as defined by the UIC group number) or to all users on the system. VMS ACL-based protection is useful for specifying access for a collection of users that are not in the same UIC group.

CMS ACLs are useful for controlling access to CMS objects and to CMS operations (commands) performed on those objects. Generally, you should use CMS ACLs whenever CMS-specific control is needed instead of or in addition to VMS protection mechanisms. CMS ACLs are very similar to VMS ACLs; the difference is that while VMS ACLs are used to specify read, write, execute, and delete access, CMS ACLs are used to specify access types for CMS operations.

This chapter discusses both security mechanisms; however, you should fully understand the composition of VMS ACLs and their syntax requirements before using CMS ACLs. For more information on VMS ACLs, see the *VMS DCL Dictionary*, the *Guide to Using VMS*, the *Guide to VMS Files and Devices*, the *VMS DCL Concepts Manual*, and the *Guide to VMS System Security*.

7.1 VMS File Access

When you try to access a directory or file, VMS determines whether or not you are allowed access by checking the protection mask against your UIC (unless there is an ACL on the directory or file that grants immediate access to the directory or file). Specifically, VMS follows these steps to determine whether a user is allowed access to a particular directory or file:

1. It evaluates any ACLs and grants or restricts the associated access
2. If an ACL does not specifically grant or deny access to the user, or if there is no associated ACL, it uses UIC-based protection to determine access.

BYPASS privilege or GRPPRV, READALL, or SYSPRV privileges may grant the user access, even if it is denied by the UIC- or ACL-based protection schemes.

To fully access a CMS library, you must have the following VMS protection scheme:

- Read and write access to the CMS library directory
- Read and write access to the 00CMS.CMS control file
- Read, write, and delete access to the 00CMS.HIS control file
- Read and delete access to the element data files
- Execute access to the CMS images SYS$SYSTEM:CMS.EXE, SYS$SHARE:CMSSHR.EXE, and SYS$SHARE:CMSPROSHR.EXE.

To use the default event action handler, you need the following access:

- Execute access to the CMS image SYS$SHARE:CMS$EVENT_ACTION.EXE.

To define CMS messages to the VMS Message Utility, you need the following access:

- Execute access to the CMS image SYS$MESSAGE:CMSMSG.EXE.

To use the CMS DECwindows interface, you additionally need the following access:

- Read access to the files DECW$SYSTEM_DEFAULTS:CMS$DW.UID, DECW$SYSTEM_DEFAULTS:CMS$DW_DEFAULTS.DAT, and SYS$HELP:CMS$DW_HELP.HLB
- Execute access to the CMS image SYS$SYSTEM:CMS$DW.EXE

If you allow read-only access to a library directory or the 00CMS.CMS file, users cannot make changes to the contents of the library. You must have delete access to an element data file to delete, reserve, or replace a generation of the element. To modify the name of an element, you must have delete access to the element data file and to its corresponding reference copy, if one exists.

You should set up a library so that at least one account has read, write, and delete access to every element data file in the library. All three types of access are necessary to execute the VERIFY/RECOVER and VERIFY /REPAIR commands (see Chapter 9).

In some cases when you use the VMS file protection scheme, the methods you use to manipulate a file may modify certain fields in the file header. When you next use CMS on the library, CMS informs you that some other means has been used to access the library; you must then execute the VERIFY/REPAIR command (see Section 9.2.3).

The following sections summarize procedures that you can use to define VMS access to your CMS library. For more information, see the *VMS DCL Dictionary*, the *Guide to Using VMS*, the *VMS DCL Concepts Manual*, and the *Guide to VMS System Security*.

7.1.1 Assigning UIC Protection

UIC-based protection controls access to directories and files as well as other VMS objects. In VMS, each user has an associated UIC. Typically, UICs are presented in numeric or alphanumeric format, for example, [221,253], or [PROJECT,JONES].

In addition, every file has a protection mask and owner UIC associated with it. When a user tries to gain access to a directory or file, the system first checks for existing ACLs, then, if none exist, checks the UIC-based protection mask. A UIC protection mask allows or denies the following types of access:

- Read (R)
- Write (W)
- Execute (E)
- Delete (D)

The protection mask describes the categories of users who have access to a directory or file, and the type of access that each category has. The categories of users are as follows:

- System (S)
- Owner (O)
- Group (G)
- World (W)

You use the DCL command SET PROTECTION to specify a particular protection mask for a directory and its contents. The following example shows the protection mask that you can use to allow system, owner, and group access to the library directory [PROJECT]CMSLIB.DIR:

```
$  SET PROTECTION=(S:RWE,O:RWE,G:RWE,W) [PROJECT]CMSLIB.DIR
```

Note that this protection mask denies world access to the library. Similarly, you can use the SET PROTECTION command to specify a UIC protection mask for an individual file within the library directory. For example:

```
$  SET PROTECTION=(S:RWD,O:RWD,G:RWD,W) [PROJECT.CMSLIB]00CMS.CMS
```

For more information, see the *Guide to VMS System Security*.

7.1.2 Assigning VMS ACL Protection

An ACL consists of access control entries (ACEs) that grant or deny access to a directory or file (or other VMS object) to specific users. You use ACLs with a library directory to define access to an entire library. You use ACLs with library files to establish greater control over access to library contents. Generally, VMS ACLs are used in conjunction with the standard UIC-based protection as a way to fine-tune protection.

You can use the following DCL commands to manipulate entire VMS ACLs or individual ACEs:

EDIT/ACL
SET ACL
SET FILE/ACL
SET DEVICE/ACL
SET DIRECTORY/ACL

You can use the following DCL commands to display VMS ACLs:

SHOW ACL
DIRECTORY/ACL

DIRECTORY/FULL
DIRECTORY/SECURITY

See the *VMS DCL Dictionary* for more information on these commands. See the *Guide to VMS System Security* for more information on using ACLs and ACEs.

7.1.2.1 Using VMS ACLs on Directories

VMS directory ACLs provide three means of controlling access to a directory:

- By controlling access to the directory file itself. For example:

```
$  SET FILE/ACL=(IDENTIFIER=DBASEGRP,ACCESS=READ+WRITE) CMSLIB.DIR
```

  This ACE grants read and write access to the directory file CMSLIB.DIR to users who have the DBASEGRP identifier.

- By specifying a default UIC protection mask to be assigned to each new file created in the directory. To specify a particular UIC protection mask, use the DEFAULT_PROTECTION keyword as the first field of an ACE. For example:

```
$  SET FILE/ACL=(DEFAULT_PROTECTION,S:RWED,O:RWED,G:RWED) CMSLIB.DIR
```

  This ACE specifies that the UIC protection (S:RWED,O:RWED,G:RWED) be applied to each new file created in the directory. (It does not affect any files that may already exist in the directory.) If no other ACEs impose stricter limitations, the system, owner, and group users are granted full access to new files in the library.

- By specifying a default ACL to be assigned to each file created in the directory. To specify a default ACL, use the OPTIONS=DEFAULT clause in the second field of an ACE that is applied to a directory file. For example:

```
$  SET FILE/ACL=(IDENTIFIER=DBASEGRP,OPTIONS=DEFAULT,ACCESS=READ+ -
_$  WRITE+DELETE) CMSLIB.DIR
```

  The OPTIONS=DEFAULT clause directs the operating system to duplicate this ACE in the ACL of every new file that is created in the directory. This ACE grants read, write, and delete access to users who have the DBASEGRP identifier.

7.1.2.2 Using VMS ACLs on Files

To exercise greater control over library access, you can explicitly set the file protection for each file in the library. Once you have created the first generation of an element, you can add the necessary ACEs to the ACL for the element data file. Every time you create a new generation of the file, CMS creates a new version of the file in the library directory, and the operating system automatically duplicates the ACL.

For example, you might establish the following ACL for an element data file:

```
$  SET FILE/ACL=(IDENTIFIER=CMSMGR,ACCESS=READ+WRITE+DELETE),-
_$  (IDENTIFIER=[JONES],ACCESS=READ+WRITE+DELETE), -
_$  (IDENTIFIER=[507,*],ACCESS=READ) [PROJECT.CMSLIB.CMS$000]EXAMPLE.PAS
```

This ACL allows both the user with the CMSMGR identifier and user JONES read, write, and delete access to the element EXAMPLE.PAS. Users in the UIC group identified by number 507 can only read (fetch) but cannot modify the element.

You must have both read and delete access to an element data file to reserve and replace generations of the corresponding element. If you reserve a generation of an element and then the access changes (so that either your account and the element data file ACE no longer have the same identifier, or you no longer have delete access to the element data file), you cannot replace the reserved generation.

Table 7–1 shows a list of the CMS commands and the protection required for each object (an element data file, a control file, or a library directory) that the command accesses.

Table 7–1: File Access Required for CMS Commands

Command	Library Directory and Subdirectories	00CMS.CMS	00CMS.HIS	Element Data File	Reference Copy File	Reference Copy Directory
ACCEPT GENERATION	RW	RW	RW			
ANNOTATE	R	R		R		
CANCEL REVIEW	RW	RW	RW			
CONVERT LIBRARY						

(continued on next page)

Table 7–1 (Cont.): File Access Required for CMS Commands

Command	Library Directory and Subdirectories	00CMS.CMS	00CMS.HIS	Element Data File	Reference Copy File	Reference Copy Directory
—V2-library-name	R	RW	R	R		
—V3-library-name	RW[1]					RW
COPY ELEMENT	RW[2]	RW	RW	R		RW
CREATE CLASS	RW	RW	RW			
CREATE ELEMENT	RW	RW	RW			RW
CREATE GROUP	RW	RW	RW			
CREATE LIBRARY	RW[1]					
DELETE CLASS	RW	RW	RW			
DELETE ELEMENT	RW	RW	RW	RD	D	RW
DELETE GENERATION	RW	RW	RW	RD	D	RW
DELETE GROUP	RW	RW	RW			
DELETE HISTORY	RW	RW	RWD			
DIFFERENCES[3]	R	R		R		
FETCH	RW[4]	R	RW[4]	R		
INSERT ELEMENT	RW	RW	RW			
INSERT GENERATION	RW	RW	RW			
INSERT GROUP	RW	RW	RW			
MARK GENERATION	RW	RW	RW			
MODIFY CLASS	RW	RW	RW			
MODIFY ELEMENT	RW	RW	RW	RD	D	RW
MODIFY GENERATION	RW	RW	RW			
MODIFY GROUP	RW	RW	RW			
MODIFY LIBRARY	RW	RW	RW		R	R

[1]The directory must be empty.
[2]You must have read access to the source library and both read and write access to the destination library.
[3]You must have access to the library and its contents only when you specify an element generation in the differences transaction.
[4]You must have write access to the library directory and read and write access to the history file only if you enter a remark for the fetch transaction.

(continued on next page)

Table 7–1 (Cont.): File Access Required for CMS Commands

Command	Library Directory and Subdirectories	00CMS.CMS	00CMS.HIS	Element Data File	Reference Copy File	Reference Copy Directory
REJECT GENERATION	RW	RW	RW			
REMARK	RW	RW	RW			
REMOVE ELEMENT	RW	RW	RW			
REMOVE GENERATION	RW	RW	RW			
REMOVE GROUP	RW	RW	RW			
REPLACE	RW	RW	RW	RD	D	RW
RESERVE	RW	RW	RW	RD		
RETRIEVE ARCHIVE[5]						
REVIEW GENERATION	RW	RW	RW			
SET ACL	RW	RW	RW			
SET LIBRARY	R	R				
SHOW commands	R	R				
SHOW ARCHIVE[5]						
SHOW HISTORY	R	R	R			
SHOW LIBRARY	R					
UNRESERVE	RW	RW	RW			
VERIFY/RECOVER	RW	RW	RW	RWD		
VERIFY/REPAIR	RW	RW	RW	RWD	RD	RW

[5]You must have read access to the archive file.

If you have set up a restrictive file protection scheme and there is a system failure during a CMS transaction that leaves your library in an inconsistent state, a user with sufficient access to the library and its files should execute the VERIFY/RECOVER command (see Chapter 9). You can also recover the library if you have BYPASS privilege (see Section 7.3), or read, write, and delete access to all the library files.

7.2 CMS ACLs

A CMS ACL is used to control access to CMS library objects. You can assign CMS ACLs to the following types of objects:

- Elements
- Groups
- Classes
- Element list
- Group list
- Class list
- History
- Library attributes
- Commands

When there is no ACL on a command or other object, access to the command or other object is unrestricted. Assigning an ACL to an object limits access to the specified user or users.

To determine whether access to an object is allowed, CMS evaluates the ACL on that object. If no ACL exists, access to the object is granted. If an ACL does exist, CMS searches the ACL sequentially for the first ACE that the user matches. A match is determined by comparing the identifiers specified in the ACE against the identifiers held by the user. If the user holds all the identifiers specified in the ACE, that ACE is a match. CMS grants the specified access of the first ACE matched; if another ACE further down in the ACL also matches, it has no effect. If none of the ACEs match, access is denied.

Note that if you are granted access to an object by CMS ACLs, you still need access to the files via VMS protection mechanisms. (However, the use of BYPASS privilege will allow you access; see Section 7.3 for more information.)

There are two ways in which you can use CMS ACLs:

- To control and restrict access to CMS commands

  For example, you can create an ACL specifying certain users who are not allowed to use the DELETE ELEMENT command, or users who are allowed to only use the FETCH, RESERVE, and REPLACE commands. See Section 7.2.2 for more information.

- To control and restrict access to CMS objects

  For example, you can create an ACL specifying certain users who are not allowed to insert or modify a particular element. You can put ACLs on elements, groups, and classes as well as on the element, group, and class lists. You can also put an ACL on the entire library and on the library history. See Section 7.2.3.2 and Section 7.2.3.3 for more information.

You can also use CMS ACLs to define CMS events (see Chapter 8).

7.2.1 Creating CMS ACLs

An ACL consists of ACEs that grant or deny access to a command or other object to specific users.

You can use two types of ACEs in CMS:

- Identifier ACEs—control which users can perform which CMS operations on a specified object.
- Action ACEs—define CMS events and specify actions to be taken when the events occur (these are described in Chapter 8).

The following sections describe the format of an ACE and the format of an ACL.

7.2.1.1 ACE Format

An identifier ACE has the following format:

```
(IDENTIFIER=identifier [,OPTIONS=options] [,ACCESS=access])
```

identifier
This field can contain any valid VMS identifier. You use identifiers to specify the users in an ACL. There are three types of identifiers:

- UIC identifiers
- General identifiers
- System-defined identifiers

UIC identifiers are described in Section 7.1.1. General identifiers identify groups of users on the system. For example, DBASEGRP, or CMSPROJ_MEMBR are general identifiers. System-defined identifiers are described in the *Guide to VMS System Security*.

You can specify multiple identifiers by separating them with a plus sign (+). The plus sign indicates the logical AND operation. CMS grants the access included in the ACE only for the user who matches all the identifiers. For example:

```
(IDENTIFIER=PROJ_LEADER + [PROJ,*])
```

In this example, the multiple identifier is matched only if the user both holds the PROJ_LEADER identifier and belongs to the PROJ group.

options
This field can contain the keyword DEFAULT or NONE. This option is valid only for object lists, that is, an element list, group list, or class list. It is not valid for commands. See Section 7.2.3.2 for more information on the options clause.

access
This field specifies the type of access that CMS allows the user or users identified in the identifier clause of the ACE. You can specify multiple access types by separating them with a plus sign (+). The plus sign indicates the logical OR operation. For example:

```
(IDENTIFIER=PROJ_LEADER, ACCESS=MODIFY+DELETE)
```

This example indicates that both the modify and delete operations are allowed for the user holding the PROJ_LEADER identifier.

The next section provides more detail on CMS access types.

7.2.1.2 Access Types

Figure 7–1 shows all the possible access types for CMS ACLs, along with the object types for which they are meaningful.

Figure 7–1: CMS ACL Access Types

Access	Element	Element List	Group	Group List	Class	Class List	History	Library Attributes	Commands
ACCEPT	X	X							
ANNOTATE	X	X							
BYPASS	X	X							
CANCEL	X	X							
CONTROL	X	X	X	X	X	X	X	X	X
COPY	X	X							
CREATE		X		X		X			
DELETE	X	X	X	X	X	X	X		
EXECUTE									X
FETCH	X	X							
INSERT			X	X	X	X			
MARK	X	X							
MODIFY	X	X	X	X	X	X		X	
REJECT	X	X							
REMARK							X		
REMOVE			X	X	X	X			
REPAIR	X	X						X	
REPLACE	X	X							
RESERVE	X	X							
REVIEW	X	X							
UNRESERVE	X	X							
VERIFY	X	X						X	

ZK–7993–GE

EXECUTE Access

To perform any CMS operation, you must have EXECUTE access to the command in addition to the appropriate access to the object or objects accessed by the command. For example, to create an element, you need the following:

- EXECUTE access to the CREATE ELEMENT command
- CREATE access to the element list

To create an element and reserve it, you need the following:

- EXECUTE access to the CREATE ELEMENT command
- CREATE access to the element list
- EXECUTE access to the RESERVE command
- RESERVE access to the element

To copy an element, in the source library, you need the following:

- EXECUTE access to the COPY ELEMENT command
- COPY access to the element

To copy an element, in the destination library, you need the following:

- EXECUTE access to the CREATE ELEMENT command
- CREATE access to the element list

CONTROL Access

To modify or delete an ACL on an object, you must have CONTROL access to the object. In addition, you must have EXECUTE access to the SET ACL command.

You can prevent other users from modifying or deleting an ACL on an object by giving only yourself CONTROL access. Note that at least one user must have CONTROL access; if not, then you must use BYPASS privilege to modify or delete that ACL.

See Section 7.2.2 for information on specifying ACLs on commands. See Section 7.2.3 for more information on specifying ACLs on other object types.

7.2.1.3 ACL Format

You use the CMS SET ACL command to specify ACEs on commands and other objects in the CMS library. The SET ACL command has the following format:

```
SET ACL /OBJECT_TYPE=type object-expression "remark"
```

The object expression depends on the object type; they must be related as shown in Table 7–2.

Table 7–2: Object Types and Related Expressions

Object Type	Object Expression
ELEMENT	An element expression
GROUP	A group expression
CLASS	A class expression
COMMAND	The name of a command, or a list of commands
LIBRARY	ELEMENT_LIST GROUP_LIST CLASS_LIST HISTORY LIBRARY_ATTRIBUTES

If the object type is LIBRARY, the object expression must be one or more keywords (called subtypes), as specified in Table 7–2. You can abbreviate these subtypes.

The SET ACL command is described in detail in the Command Dictionary. Sections 7.2.2 and 7.2.3 discuss specifying ACLs with commands and other objects.

7.2.2 Specifying ACLs with Commands

Specifying a CMS ACL on a command allows you to restrict one or more users from accessing that command. This provides a broad protective mechanism that allows greater control over the CMS library than using VMS ACLs and UICs.

You use CMS ACLs on commands and other objects; in most cases, using CMS ACLs on commands is the most effective method to suit most user's needs.

When you use the SET ACL command to set an ACL on a command, the object type must be COMMAND, as specified in Table 7–2. The object expression must be one of the following commands:

ACCEPT_GENERATION	MARK_GENERATION
ANNOTATE	MODIFY_CLASS

CANCEL_REVIEW	MODIFY_ELEMENT
COPY_ELEMENT	MODIFY_GENERATION
CREATE_CLASS	MODIFY_GROUP
CREATE_ELEMENT	MODIFY_LIBRARY
CREATE_GROUP	REJECT_GENERATION
DELETE_CLASS	REMARK
DELETE_ELEMENT	REMOVE_ELEMENT
DELETE_GENERATION	REMOVE_GENERATION
DELETE_GROUP	REMOVE_GROUP
DELETE_HISTORY	REPLACE
DIFFERENCES	RESERVE
FETCH	REVIEW_GENERATION
INSERT_ELEMENT	SET_ACL
INSERT_GENERATION	UNRESERVE
INSERT_GROUP	VERIFY

You can display this list of commands by issuing the SHOW ACL /OBJECT_TYPE=COMMAND * command. Note that commands containing two words must include an underscore.

To access a command, you must have EXECUTE access to that command.

7.2.2.1 Examples of ACLs on Commands

1.
```
$ CMS SET ACL/OBJECT_TYPE=COMMAND RESERVE,REPLACE -
_$    /ACL=(IDENTIFIER=[PROJECT,WILSON],ACCESS=EXECUTE) ""
```

   This command specifies that the user with the UIC [PROJECT,WILSON] is allowed EXECUTE access to the RESERVE and REPLACE commands.

2.
```
$ CMS SET ACL/OBJECT_TYPE=COMMAND INSERT_ELEMENT -
_$  /ACL=(IDENTIFIER=JONES,ACCESS=CONTROL) ""
%CMS-S-MODACL, modified access control list for command
DISKX:[PROJECT.CMSLIB]INSERT_ELEMENT

$ CMS INSERT ELEMENT ELEMENT.2 GROUP2 ""
%CMS-E-NOINSERT, error inserting DISKX:[PROJECT.CMSLIB]ELEMENT.2
into group DISKX:[PROJECT.CMSLIB]GROUP2
-CMS-E-NOACCESS, no execute access to INSERT ELEMENT command
$ CMS SET ACL/OBJECT_TYPE=COMMAND INSERT_ELEMENT -
_$  /ACL=(IDENTIFIER=JONES,ACCESS=EXECUTE+CONTROL)""
%CMS-S-MODACL, modified access control list for command
DISKX:[PROJECT.CMSLIB]INSERT_ELEMENT

$ CMS SHOW ACL/OBJECT_TYPE=COMMAND INSERT_ELEMENT
```

```
ACLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

INSERT_ELEMENT
         (IDENTIFIER=[WORK,JONES],ACCESS=EXECUTE+CONTROL)

$  CMS INSERT ELEMENT ELEMENT.2 GROUP2 ""
%CMS-S-INSERTED, element DISKX:[PROJECT.CMSLIB]ELEMENT.2 inserted into
group DISKX:[PROJECT.CMSLIB]GROUP2
```

In this example, user JONES assigns an ACL containing CONTROL access to the INSERT ELEMENT command. The SHOW ACL command displays the ACL on INSERT ELEMENT. (Note that commands containing more than one word must be specified with an underscore.) The example then shows that JONES tries to insert another element into another group. The attempt fails because, although JONES has CONTROL access to the INSERT ELEMENT command, he does not also have EXECUTE access to it.

CONTROL access allows you to modify the ACL. Because JONES has CONTROL access, he modifies the ACL to allow himself EXECUTE access to the INSERT ELEMENT command. (You must have EXECUTE access to use any commands.) He can then insert elements successfully.

3.
```
$  CMS SET LIBRARY [WORK.CMSLIB],[PROJECT.CMSLIB]
%CMS-I-LIBIS, library is DISKX:[WORK.CMSLIB]
%CMS-I-LIBINSLIS, library DISKX:[PROJECT.CMSLIB] inserted at end of
library list
%CMS-S-LIBSET, library set

$  CMS SET ACL/ACL=((IDENTIFIER=SMITH,ACCESS=CONTROL),(IDENTIFIER=*, -
_$  ACCESS=NONE)) DELETE_ELEMENT/OBJECT_TYPE=COMMAND/OCCLUDE=NOOTHER ""
%CMS-S-MODACL, modified access control list for command
DISKX:[WORK.CMSLIB]DELETE_ELEMENT
%CMS-S-MODACL, modified access control list for command
DISKX:[PROJECT.CMSLIB]DELETE_ELEMENT
%CMS-S-MODACLS, 2 access control lists modified
```

This example shows the use of occlusion. The SET ACL command is used to restrict access to the DELETE ELEMENT command in both libraries [WORK.CMSLIB] and [PROJECT.CMSLIB]. See Section 3.3 for more information on occlusion.

7.2.3 Specifying ACLs with Other CMS Objects

For users requiring more restrictive control, you can fine tune access by using CMS ACLs in combination with objects besides commands. These other objects include:

- Elements, groups, and classes
- Element lists, group lists, and class lists

- Library history and library attributes

The following sections describe these objects in detail.

7.2.3.1 Specifying ACLs on Elements, Groups, and Classes

Specifying a CMS ACL on an element, group, or class allows you to restrict one or more users from accessing that object. For example, you can create an ACL specifying certain users who are not allowed to insert or modify a particular element.

When you use the SET ACL command on an object, the object type must be ELEMENT, GROUP, or CLASS as specified in Table 7–2. The object expression must be an element, group, or class expression, respectively.

See Figure 7–1 for all the possible access types that are allowed with these objects. Note that not all access types have meaning for all objects. For example, giving a user RESERVE access to a class is meaningless, because the RESERVE command does not operate on classes.

7.2.3.1.1 Examples of ACLs on Elements, Groups, and Classes

1.
```
$ CMS SET ACL EXAMPLE.PAS/OBJECT_TYPE=ELEMENT -
$_ /ACL=(IDENTIFIER=[555,*],ACCESS=FETCH) ""
```

   This command specifies that users with the UIC [555,*] are allowed only FETCH access to the element EXAMPLE.PAS.

2.
```
$ CMS SET ACL/OBJECT_TYPE=ELEMENT ELEMENT.1 -
_$ /ACL=(IDENTIFIER=JONES,ACCESS=RESERVE+CONTROL)""
%CMS-S-MODACL, modified access control list for element
DISKX:[PROJECT.CMSLIB]ELEMENT.1

$ CMS SET ACL/OBJECT_TYPE=ELEMENT ELEMENT.1/ACL=(IDENTIFIER=JONES, -
_$ ACCESS=NONE) ""
%CMS-S-MODACL, modified access control list for element
DISKX:[PROJECT.CMSLIB]ELEMENT.1

$ CMS RESERVE ELEMENT.1 ""
%CMS-E-NOFETCH, error reserving element DISKX:[PROJECT.CMSLIB]ELEMENT.1
-CMS-E-NOACCESS, no reserve access to element ELEMENT.1

$ CMS SET ACL/OBJECT_TYPE=ELEMENT ELEMENT.1 -
_$ /ACL=(IDENTIFIER=JONES,ACCESS=RESERVE+CONTROL) ""
%CMS-E-NOMODACL, error modifying access control list for element
DISKX:[PROJECT.CMSLIB]ELEMENT.1
-CMS-E-NOACCESS, no control access to element ELEMENT.1
```

   In this example, user JONES assigns an ACL containing RESERVE and CONTROL access to the element ELEMENT.1. Then, another user (who has BYPASS privilege) sets an ACL on ELEMENT.1 containing

ACCESS=NONE, thus restricting JONES from reserving that element, and removing any prior access that JONES had assigned. JONES then tries to reserve the element. His attempt is unsuccessful because he no longer has RESERVE access to the element. He also does not have CONTROL access to the element, which would allow him to modify the ACL assigned by the second user.

3.
```
$ CMS SET ACL/OBJECT_TYPE=CLASS CLASS1 -
_$ /ACL=(IDENTIFIER=JONES,ACCESS=CONTROL) ""
%CMS-S-MODACL, modified access control list for class
DISKX:[PROJECT.CMSLIB]CLASS1

$ CMS SHOW ACL/OBJECT_TYPE=CLASS CLASS1

ACLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

CLASS1
         (IDENTIFIER=[WORK,JONES],ACCESS=CONTROL)

$ CMS MODIFY CLASS/NOREAD_ONLY CLASS1 ""
%CMS-E-NOMODIFY, error modifying class DISKX:[PROJECT.CMSLIB]CLASS1
-CMS-E-NOACCESS, no modify access to class CLASS1

$ CMS SET ACL/OBJECT_TYPE=CLASS CLASS1 -
_$ /ACL=(IDENTIFIER=JONES,ACCESS=MODIFY+CONTROL) ""
%CMS-S-MODACL, modified access control list for class
DISKX:[PROJECT.CMSLIB]CLASS1

$ CMS MODIFY CLASS/NOREAD_ONLY CLASS1 ""
%CMS-S-MODIFIED, class DISKX:[PROJECT.CMSLIB]CLASS1 modified
```

In this example, user JONES assigns an ACL giving himself CONTROL access to the class CLASS1. He then tries to modify the class, but is unsuccessful because, although he has CONTROL access to the class, he does not also have MODIFY access. However, since JONES has CONTROL access, this allows him to issue the SET ACL command. He then assigns another ACL containing both CONTROL and MODIFY access to the class, and then successfully modifies the class.

7.2.3.2 Specifying ACLs on Element Lists, Group Lists, and Class Lists

The difference between an object and its list is important in the understanding of CMS ACLs. Conceptually, element, group, and class lists are objects representing all the elements, groups, and classes already existing or yet to be created in a CMS library. Object lists are used solely with CMS ACLs.

When you use the SET ACL command on an object list, the object type must be LIBRARY. The object expression must be one of the following keywords: ELEMENT_LIST, GROUP_LIST, or CLASS_LIST, as specified in Table 7–2 (see Section 7.2.3.3 for information on the HISTORY and LIBRARY_ATTRIBUTES keywords). See Figure 7–1 for all the possible access types that are allowed with these objects.

Specifying an ACL on an object list grants the right to create new objects in the library. For example:

```
$  CMS SET ACL/OBJECT_TYPE=LIBRARY GROUP_LIST -
_$ /ACL=(IDENTIFIER=PROJ_TEAM,ACCESS=CREATE) ""
```

This example assigns an ACL to the group list, and allows only the holders of the identifier PROJ_TEAM to create groups in the library.

You can also specify a default ACL to be used on newly created objects in the library. You do this by specifying the OPTIONS=DEFAULT clause in the ACL of an object list. For example:

```
$  CMS SET ACL/OBJECT_TYPE=LIBRARY ELEMENT_LIST/ACL=(IDENTIFIER=PROJ_TEAM, -
_$ OPTIONS=DEFAULT, ACCESS=FETCH) ""
```

This example specifies that only holders of the PROJ_TEAM identifier will be able to FETCH newly created elements.

Each time you create a new object, CMS searches for the ACEs containing the OPTIONS=DEFAULT clause in the ACL of the corresponding object list. If any exist, the newly created object (or objects) are automatically assigned the ACEs containing the OPTIONS=DEFAULT clause. For example, if you specify ACEs containing OPTIONS=DEFAULT in the ACL of a group list, CMS assigns the default ACEs in the ACL to any newly created groups.

OPTIONS=DEFAULT is only valid for object lists. Note that the OPTIONS=DEFAULT clause does not affect any objects already in the list, only new objects. You can assign default ACEs to existing objects by specifying the SET ACL/DEFAULT command.

Because it is not possible to assign an ACL granting CREATE access to an object that does not yet exist, the only access types that are meaningful for an object list ACE not containing the OPTIONS=DEFAULT clause are CREATE and CONTROL access. All other access types are meaningful only if the OPTIONS=DEFAULT clause is present.

CAUTION

Because default ACEs do not grant access, when you use default ACEs, you should assign another ACE granting yourself or another user a minimum of CONTROL access to an object; otherwise, you could restrict your own access to the object.

When you use the COPY ELEMENT command, the source element's ACL is not assigned to the target element. Instead, the target element receives the default ACL (if any) that is set on the element list.

If you do not use the OPTIONS=DEFAULT clause, newly created objects are not affected by the ACL (if any) on the object list. The OPTIONS=NONE clause indicates that new objects are not assigned that ACE from the object list. NONE is equivalent to not specifying a clause. Note that the OPTIONS=NONE clause is not displayed when you issue the SHOW ACL command.

7.2.3.2.1 Examples of ACLs on Lists

1.
```
$ CMS SET ACL/OBJECT_TYPE=LIBRARY ELEMENT_LIST -
_$   /ACL=((IDENTIFIER=JONES,OPTIONS=DEFAULT,ACCESS=RESERVE -
_$   +CONTROL),(IDENTIFIER=JONES,ACCESS=CREATE+CONTROL)) ""
```

   This command places two ACEs on the element list. The first ACE is a default ACE, which will cause all new elements created in the library to inherit an ACE giving RESERVE access to the user with the identifier JONES. The second ACE defines the access to the element list itself. Because CREATE access is specified, the user with the identifier JONES is allowed to create elements in the library. Note that both ACEs also grant control access; this is necessary to allow modification of the ACL once it has been created.

2.
```
$ CMS SET ACL/OBJECT_TYPE=LIBRARY CLASS_LIST -
_$ /ACL=(IDENTIFIER=JONES,ACCESS=CONTROL) ""
%CMS-S-MODACL, modified access control list for subtype
DISKX:[PROJECT.CMSLIB]CLASS_LIST

$ CMS CREATE CLASS CLASS4 ""
%CMS-E-NOCREATE, error creating class DISKX:[PROJECT.CMSLIB]CLASS4
-CMS-E-NOACCESS, no create access to CLASS_LIST

$ CMS SET ACL/OBJECT_TYPE=LIBRARY CLASS_LIST -
_$ /ACL=(IDENTIFIER=JONES,ACCESS=CREATE+CONTROL) ""
%CMS-S-MODACL, modified access control list for subtype
DISKX:[PROJECT.CMSLIB]CLASS_LIST

$ CMS SHOW ACL/OBJECT_TYPE=LIBRARY CLASS_LIST

ACLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

CLASS_LIST
         (IDENTIFIER=[WORK,JONES],ACCESS=CREATE+CONTROL)

$ CMS CREATE CLASS CLASS4 ""
%CMS-S-CREATED, class DISKX:[PROJECT.CMSLIB]CLASS4 created
```

   In this example, JONES assigns an ACL containing CONTROL access to the class list. Assigning an ACL to the class list will affect the creation of new classes in the library. However, when he tries to create a new class, he receives an error because he does not also have CREATE access to the class list. Because he has CONTROL access, he then assigns a new ACL giving himself both CONTROL and CREATE access. He can then create new classes.

3.
```
$ CMS SET ACL/OBJECT_TYPE=LIBRARY ELEMENT_LIST -
_$ /ACL=((IDENTIFIER=JONES,ACCESS=CREATE+CONTROL), -
_$  (IDENTIFIER=FLYNN,OPTIONS=DEFAULT,ACCESS=FETCH), -
_$  (IDENTIFIER=SMITH,OPTIONS=DEFAULT,ACCESS=RESERVE+REPLACE)) ""
%CMS-S-MODACL, modified access control list for subtype
DISKX:[PROJECT.CMSLIB]ELEMENT_LIST

$ CMS SHOW ACL/OBJECT_TYPE=LIBRARY ELEMENT_LIST

ACLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

ELEMENT_LIST
         (IDENTIFIER=[WORK,JONES],ACCESS=CREATE+CONTROL)
         (IDENTIFIER=[WORK,FLYNN],OPTIONS=DEFAULT,ACCESS=FETCH)
         (IDENTIFIER=[WORK,SMITH],OPTIONS=DEFAULT,ACCESS=REPLACE+RESERVE)

$ CMS CREATE ELEMENT ELEMENT.4 ""
%CMS-S-CREATED, element DISKX:[PROJECT.CMSLIB]ELEMENT.4 created

$ CMS SHOW ACL/OBJECT_TYPE=ELEMENT ELEMENT.4

ACLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

ELEMENT.4
         (IDENTIFIER=[WORK,FLYNN],ACCESS=FETCH)
         (IDENTIFIER=[WORK,SMITH],ACCESS=REPLACE+RESERVE)
```

In this example, user JONES assigns an ACL on the element list. The ACL specifies the following:

- JONES is allowed CREATE and CONTROL access to the element list.
- By using OPTIONS=DEFAULT, JONES assigns user FLYNN only FETCH access to new elements created in the library.
- By using OPTIONS=DEFAULT, JONES assigns user SMITH only REPLACE and RESERVE access to new elements created in the library.

JONES then successfully creates a new element named ELEMENT.4. When the SHOW ACL command is issued, the default access on the element for each user is displayed. User JONES's access is not displayed because he has access to the element list, not the element itself.

7.2.3.3 Specifying ACLs on Libraries and History

Specifying a CMS ACL on the library or the library history allows you to restrict one or more users from certain types of access to the library, or from certain types of access to the library history. You can restrict users from the following types of access to the library:

- MODIFY
- REPAIR
- VERIFY

You can restrict users from the following types of access to the library history:

- DELETE
- REMARK

REPAIR and VERIFY Access

REPAIR access is required to use VERIFY/REPAIR on a library. If you have REPAIR access to a library object, you can issue VERIFY/REPAIR, even if you do not have VERIFY access to that library.

Because CMS cannot determine whether access control information is valid until it verifies the database, the VERIFY and REPAIR access types apply only to element data file verification. Once the database has been verified, CMS checks the following:

- Access to the VERIFY command
- VERIFY or REPAIR access to the library
- VERIFY or REPAIR access to each element

When you use the SET ACL command on a library or history, the object type must be LIBRARY, as specified in Table 7–2. The object expression must be either LIBRARY_ATTRIBUTES or HISTORY.

See Figure 7–1 for all the possible access types that are allowed on a library or history.

7.2.3.3.1 Examples of ACLs on History and the Library

1.
```
$ CMS SET ACL/OBJECT_TYPE=LIBRARY HISTORY -
_$ /ACL=(IDENTIFIER=JONES,ACCESS=CONTROL) ""
%CMS-S-MODACL, modified access control list for subtype
DISKX:[PROJECT.CMSLIB]HISTORY

$ CMS REMARK "Add a remark to history"
%CMS-E-NOREMARK, error adding remark to library
-CMS-E-NOACCESS, no remark access to library history

$ CMS SET ACL/OBJECT_TYPE=LIBRARY HISTORY -
_$ /ACL=(IDENTIFIER=JONES,ACCESS=REMARK+CONTROL) ""
%CMS-S-MODACL, modified access control list for subtype
DISKX:[PROJECT.CMSLIB]HISTORY

$ CMS REMARK "Add a remark to history"
%CMS-S-REMARK, remark added to history file
```

   In this example, JONES assigns an ACL giving herself CONTROL access to the library history. She then tries to add a remark to the library history, but is unsuccessful because she does not have REMARK access to the history. She then assigns another ACL containing both CONTROL and REMARK access, and can then successfully add a remark to the library history file.

2.
```
$ CMS SET ACL/OBJECT_TYPE=LIBRARY LIBRARY_ATTRIBUTES -
_$ /ACL=(IDENTIFIER=JONES,ACCESS=CONTROL) ""
%CMS-S-MODACL, modified access control list for subtype
DISKX:[PROJECT.CMSLIB]LIBRARY_ATTRIBUTES

$ CMS VERIFY/REPAIR
%CMS-I-VERCLS, class list verified
%CMS-I-VERCMD, command list verified
%CMS-I-VERELE, element list verified
     .
     .
     .
%CMS-I-VERCON, control file verified
%CMS-E-ERRVEREDFS, element data files verified with errors
-CMS-E-NOACCESS, no repair access to library DISKX:[PROJECT.CMSLIB]
%CMS-E-NOREPAIR, error repairing library

$ CMS SET ACL/OBJECT_TYPE=LIBRARY LIBRARY_ATTRIBUTES -
_$ /ACL=(IDENTIFIER=JONES,ACCESS=CONTROL+REPAIR) ""
%CMS-S-MODACL, modified access control list for subtype
DISKX:[PROJECT.CMSLIB]LIBRARY_ATTRIBUTES

$ CMS SHOW ACL/OBJECT_TYPE=LIBRARY LIBRARY_ATTRIBUTES

ACLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

LIBRARY_ATTRIBUTES
         (IDENTIFIER=[WORK,JONES],ACCESS=CONTROL+REPAIR)
```

```
$ CMS VERIFY/REPAIR
%CMS-I-VERCLS, class list verified
%CMS-I-VERCMD, command list verified
%CMS-I-VERELE, element list verified
     .
     .
     .
%CMS-I-VERCON, control file verified
%CMS-E-VEREDFERR, element DISKX:[PROJECT.CMSLIB]ELEMENT.1 verified with errors
-CMS-E-NOACCESS, no repair access to element ELEMENT.1
%CMS-I-VEREDF, element DISKX:[PROJECT.CMSLIB]ELEMENT.2 verified
%CMS-I-VEREDF, element DISKX:[PROJECT.CMSLIB]ELEMENT.3 verified
%CMS-E-VEREDFERR, element DISKX:[PROJECT.CMSLIB]ELEMENT.4 verified with errors
-CMS-E-NOACCESS, no repair access to element ELEMENT.4
%CMS-E-ERRVEREDFS, element data files verified with errors
%CMS-E-NOREPAIR, error repairing library

$ CMS SHOW ACL/OBJECT_TYPE=ELEMENT ELEMENT.1, ELEMENT.4

ACLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

ELEMENT.1
        (IDENTIFIER=[WORK,JONES],ACCESS=NONE)

ELEMENT.4
        (IDENTIFIER=[WORK,FLYNN],ACCESS=FETCH)
        (IDENTIFIER=[WORK,SMITH],ACCESS=REPLACE+RESERVE)
```

This example demonstrates how REPAIR access is used. First, JONES assigns an ACL to the library indicating that he is allowed CONTROL access to the library. He then tries a VERIFY/REPAIR operation on the library. This attempt is unsuccessful because he does not also have REPAIR access to the library. He assigns a new ACL containing both CONTROL and REPAIR access to the library, and tries another VERIFY /REPAIR operation on the library. This attempt is also unsuccessful because, although he has REPAIR access to the library, he does not have REPAIR access to the elements ELEMENT.1 and ELEMENT.4 (as displayed by the SHOW ACL command). When issuing VERIFY /REPAIR, you must have REPAIR access both to the library and to the individual elements in the library.

7.3 VMS BYPASS Privilege and CMS BYPASS Access

The VMS BYPASS privilege allows a user read, write, execute, and delete access to all files, bypassing UIC protection. A user holding BYPASS privilege is also granted access to any CMS object or command, regardless of any VMS or CMS protections.

Whenever you define ACLs for objects, remember that users with BYPASS privilege are granted complete access; for this reason, BYPASS privilege is usually reserved for experienced users who need this privilege.

Being granted CMS BYPASS access is not equivalent to holding VMS BYPASS privilege. The CMS BYPASS access type allows you *only* to unreserve or replace another user's reservation for an element. (VMS BYPASS privilege also allows you to unreserve or replace another user's reservation.)

The following example shows the use of CMS BYPASS access:

```
  CMS SHOW RESERVATIONS

eservations in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

LEMENT.2
    (1)    FLYNN       1       12-JAN-1989 18:57:43 ""

  CMS REPLACE ELEMENT.2/IDENTIFICATION_NUMBER=1 ""
CMS-E-NOREPLACE, error replacing DISKX:[PROJECT.CMSLIB]ELEMENT.2
CMS-E-IDENTNOTRES, reservation 1 is not reserved by you

  CMS SET ACL/OBJECT_TYPE=ELEMENT ELEMENT.2 -
_$  /ACL=(IDENTIFIER=JONES,ACCESS=BYPASS+REPLACE+CONTROL) ""
CMS-S-MODACL, modified access control list for element
ISKX:[PROJECT.CMSLIB]ELEMENT.2

  CMS SHOW ACL/OBJECT_TYPE=ELEMENT ELEMENT.2

CLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

LEMENT.2
         (IDENTIFIER=[WORK,JONES],ACCESS=CONTROL+BYPASS+REPLACE)

  CMS REPLACE ELEMENT.2/IDENTIFICATION_NUMBER=1 ""
lement DISKX:[PROJECT.CMSLIB]ELEMENT.2 currently reserved by:
    (1)    FLYNN       1       12-JAN-1989 18:57:43 ""
eplace (1) ELEMENT.2 generation 1, held by FLYNN? [Y/N] (N):  Y
CMS-S-GENCREATED, generation 2 of element DISKX:[PROJECT.CMSLIB]ELEMENT.2 created
```

This example shows the use of BYPASS access to replace another user's reservation. The user JONES unsuccessfully tries to replace FLYNN's reservation 1 of the element ELEMENT.2. JONES then assigns an ACL allowing him CONTROL, BYPASS, and REPLACE access to the element. CONTROL allows him to modify the ACL again after he replaces the element. BYPASS allows him to replace FLYNN's reservation. REPLACE is needed to perform the actual replacement. Both BYPASS and REPLACE are required; he can then successfully replace FLYNN's reservation of the element.

7.4 Combining VMS and CMS Security Mechanisms

When CMS ACLs are used in conjunction with VMS protection mechanisms, you should ensure that you allow sufficient access via VMS protection so that all users can perform necessary operations, but you should not allow unnecessary access. In other words, you should set the VMS file protections to allow only as much access as is needed by users to perform operations, as shown in Table 7–1. (A set of users can be defined by their UIC, identifiers, or both.)

If a set or sets of users still need to perform a subset of operations beyond the VMS protection you have set up, you can use CMS ACLs to obtain a more restrictive protection scheme.

For example, suppose a group of CMS users is divided into those holding the identifier LIBRARIAN, and those holding the identifier PROGRAMMER. Members of both groups are allowed to reserve elements, but only holders of the LIBRARIAN identifier are allowed to replace them.

As listed in Table 7–1, both the RESERVE and REPLACE commands require the same access to all files in the library Thus, allowing users holding the PROGRAMMER identifier sufficient access to the library files to perform a reserve operation implicitly allows them access to perform a replace operation. Using VMS file protection mechanisms, it is not possible to allow access to RESERVE while disallowing access to REPLACE. However, in CMS, you can place a CMS ACL on the REPLACE command that allows access to holders of the LIBRARIAN identifier, but disallows access to holders of the PROGRAMMER identifier.

To successfully operate in a CMS library, the library directory, control files, and the element data files must be accessible through the VMS system (including ACLs and UIC protection mechanisms). Also, the commands you issue in the library, and the objects referenced by those commands, must be accessible through the CMS ACL mechanism.

NOTE

The use of both VMS and CMS ACLs does not ensure complete library security. The library can still be accessed using means other than CMS. However, keep in mind that the library should never be accessed by means other than CMS; this may result in unrecoverable library corruption.

7.4.1 Example of Protection Scheme Using VMS and CMS Mechanisms

This example shows a possible protection scheme using both VMS and CMS security mechanisms.

Suppose a project team consists of the members Smith, Brown, Jones, Anderson, and Nelson. Smith is the project leader, Brown and Jones are senior developers, and Anderson and Nelson are junior developers. All project team members except Nelson hold the PROJECT identifier.

These project members require the following types of access to the library:

- Smith requires full access to the library.
- Brown and Jones are allowed to perform all operations except DELETE ELEMENT and DELETE GENERATION.
- Anderson is allowed to perform all operations except DELETE ELEMENT, DELETE GENERATION, and REPLACE.
- Nelson is allowed access only to the FETCH command.

In this example, the access required to the library files is set according to Table 7–1. A VMS ACL for each file could be set up as follows:

```
Library Directory and Subdirectories:
        (IDENTIFIER=PROJECT, ACCESS=READ+WRITE)
        (IDENTIFIER=NELSON, ACCESS=READ)
        (IDENTIFIER=*, ACCESS=NONE)
        (IDENTIFIER=PROJECT,OPTIONS=DEFAULT,ACCESS=READ+WRITE+DELETE)
        (IDENTIFIER=NELSON,OPTIONS=DEFAULT,ACCESS=READ)
        (IDENTIFIER=*,OPTIONS=DEFAULT,ACCESS=NONE)

00CMS.CMS:
        (IDENTIFIER=PROJECT, ACCESS=READ+WRITE)
        (IDENTIFIER=NELSON, ACCESS=READ)
        (IDENTIFIER=*, ACCESS=NONE)

00CMS.HIS:
        (IDENTIFIER=PROJECT, ACCESS=READ+WRITE+DELETE)
        (IDENTIFIER=NELSON, ACCESS=NONE)
        (IDENTIFIER=*, ACCESS=NONE)

Element Data Files:
        (IDENTIFIER=PROJECT, ACCESS=READ+WRITE+DELETE)
        (IDENTIFIER=NELSON, ACCESS=READ)
        (IDENTIFIER=*, ACCESS=NONE)
```

Note that Nelson is allowed only to use the FETCH command without also specifying a remark. This is due to Nelson's lack of access to the library directory and 00CMS.HIS. Also note that the ACE containing the ACCESS=NONE clause denies access to all library files to anyone not on the project team. The OPTIONS=DEFAULT ACEs on the library directory ensure that newly created element data files will receive the proper ACL.

While the ACLs assigned to the library files provide the access needed by the members of the project team, they still do not sufficiently restrict access as originally required. To do this, CMS ACLs must be set up on the various commands. To ensure that these ACLs are not changed except by the project leader, an additional requirement is that only Smith can use the SET ACL command. Smith must also have CONTROL access to each of the commands in order to change their ACLs once they have been assigned. The CMS ACLs could be set up as follows:

```
FETCH:
        (IDENTIFIER=SMITH, ACCESS=EXECUTE+CONTROL)
        (IDENTIFIER=PROJECT, ACCESS=EXECUTE)
        (IDENTIFIER=NELSON, ACCESS=EXECUTE)

DELETE ELEMENT and DELETE GENERATION:
        (IDENTIFIER=SMITH, ACCESS=EXECUTE+CONTROL)

REPLACE:
        (IDENTIFIER=ANDERSON, ACCESS=NONE)
        (IDENTIFIER=SMITH, ACCESS=EXECUTE+CONTROL)
        (IDENTIFIER=PROJECT, ACCESS=EXECUTE)

SET ACL:
        (IDENTIFIER=SMITH, ACCESS=EXECUTE+CONTROL)

All other commands:
        (IDENTIFIER=SMITH, ACCESS=EXECUTE+CONTROL)
        (IDENTIFIER=PROJECT, ACCESS=EXECUTE)
```

If an identifier does not match any ACE in an ACL (assuming an ACL exists) CMS denies access to the object. Thus, Nelson is denied access to all commands except FETCH. Even though Anderson holds the PROJECT identifier, he matches the first ACE in the ACL on the REPLACE command, and so is also denied access. Similarly, the ACE for Smith must be placed before the ACE for PROJECT; otherwise, Smith will match the PROJECT ACE and would not receive CONTROL access.

Chapter 8

Event Handling and Notification

You can specify lists of people who are to be notified when certain events occur in the library. An *event* is an operation involving one or more of the following objects:

Elements
Element list
Classes
Class list
Groups
Group list
History
Commands
Library attributes

The following sections describe how to specify events, how to use the default or a user-written handler, and how to use notification. Section 8.3 shows examples of using notification.

8.1 Event Handling

You specify and detect events by using CMS access control lists (ACLs) and access control entries (ACEs). CMS notifies users of events by processing one or more action ACEs in an object's ACL. The following sections discuss how to specify events and how events are detected by means of action ACEs.

8.1.1 Specifying Action ACEs

CMS ACLs support two types of ACEs: identifier ACEs and action ACEs. You use identifier ACEs to control which users can perform which CMS operations on a specified object (see Section 7.2.1). You use action ACEs to define CMS events. An action ACE allows you to specify a particular action to be taken when a CMS object is accessed in a certain way.

An action ACE has the following format:

```
(ACTION[=image], PARAMETER=string [,IDENTIFIER=identifier] [,OPTIONS=options]
[,ACCESS=access])
```

The ACTION clause identifies the ACE as an action ACE; you can optionally use it to specify a shareable image containing your own event handler routine, CMS$EVENT_ACTION (see Section 8.1.3). Do not include the .EXE file extension in the event handler name. If you do not specify a user-written event handler routine on the ACTION clause, CMS uses the default event handler SYS$SHARE:CMS$EVENT_ACTION image.

If you use the default image, the string specified on the PARAMETER clause must be a valid MAIL recipient specification, such as MYNODE::JONES or @DISTLIST, or a list of specifications separated with commas. You may need to enclose the string in quotation marks if the string contains a list, a period, a comma, or other nonalphanumeric characters. You should also enclose the string in quotation marks when differentiating between uppercase and lowercase, or CMS will convert the string to uppercase.

You can use the NOTIFY clause as a synonym for the ACTION,PARAMETER= (that is, the ACTION clause without the =image parameter) when you specify an action ACE. For example, the following specifications are equivalent:

```
(NOTIFY=@LIST)
(ACTION, PARAMETER=@LIST)
```

You cannot use the NOTIFY clause, however, if you specify a user-written handler.

The IDENTIFIER clause is optional in action ACEs. If it is not specified, CMS assumes the IDENTIFIER=* clause by default. The IDENTIFIER, OPTIONS and ACCESS clauses are described in detail in Section 7.2.1.

See Section 8.3 for examples of action ACEs.

3.1.2 Detecting Events

A CMS event occurs only when the user has been granted the right to perform the operation and the operation has been successfully performed. Therefore, you cannot use an event handler to prevent a command from performing its operation, nor does the command fail if the event handler cannot be invoked.

Multiple events can occur as a result of a single CMS command being executed. For example, if action ACEs have been assigned to the elements A.TXT and B.TXT and to the command RESERVE, then three independent events may be triggered by the command RESERVE A.TXT,B.TXT, one for each of the three objects.

An exception is the INSERT and REMOVE commands. Execution of the INSERT and REMOVE commands involves two types of objects: the objects being inserted or removed, and the objects being inserted into or removed from. CMS does not check ACLs associated with objects of the first type; therefore, insert and remove operations involving objects of the first type cannot trigger events. For example:

```
$  CMS INSERT ELEMENT A.TXT,B.TXT TEST_BAS ""
```

This command could trigger an event associated with the group TEST_BAS, but not with the elements A.TXT and B.TXT.

8.1.3 Using Your Own Event Handler

When CMS detects that a specified event has occurred, it invokes the event handler routine CMS$EVENT_ACTION in the SYS$SHARE:CMS$EVENT_ACTION image, or, if you have written your own shareable image, in your user-provided image.

You specify your own shareable image name in the ACTION clause of the ACE that is defining the event. You do this by specifying the image name after the ACTION keyword as follows:

ACTION=image_name

See Section 8.1.1 for more information.

You must include a routine named CMS$EVENT_ACTION in your image. CMS dynamically activates the CMS$EVENT_ACTION routine's image, if necessary, by calling LIB$FIND_IMAGE_SYMBOL, and then calls the CMS$EVENT_ACTION routine.

Your CMS$EVENT_ACTION routine should follow the rules for callback routines (see the *VAX DEC/Code Management System Callable Routines Reference Manual*). The CMS$EVENT_ACTION routine calling format and arguments are as follows:

CMS$EVENT_ACTION *(library_data_block,*
user_param,
library_specification_id,
ace_parameter_id,
history_record_id)

library_data_block
Type: cntrlblk
Access: read
Mechanism: by reference
Specifies the library data block for the current library.

user_param
Type: undefined
Access: modify
Mechanism: undefined
Specifies the user_arg value passed in a call to a callable CMS routine whose action caused the event. If no user argument was specified in the user call, or if the event did not occur as a result of a user call to a callable CMS routine, then the call frame entry for user_param points to a location containing the value zero. In this case, user_param is allocated as read-only storage.

library_specification_id
Type: address
Access: read
Mechanism: by reference
Specifies a string identifier for the current CMS library directory specification.

ace_parameter_id
Type: address
Access: read
Mechanism: by reference
Specifies a string identifier for the string found in the PARAMETER clause of the current ACE (the ACE that defines the current event).

history_record_id
Type: address
Access: read
Mechanism: by reference
Specifies a string identifier for the string containing the history record written as a result of the current CMS operation (the operation that caused the current event).

The library_data_block argument should be used only by the default CMS$EVENT_ACTION routine; any user-written CMS$EVENT_ACTION routine should ignore it. The user_param argument is provided so that a user-written CMS$EVENT_ACTION routine can interpret it; the default CMS$EVENT_ACTION routine ignores it. If you use the default CMS$EVENT_ACTION routine, then CMS expects the ace_parameter_id argument to point to a string containing a list of valid MAIL recipient specifications.

When the default CMS$EVENT_ACTION routine encounters errors, it signals error conditions. If the severity code of the condition is an informational or warning status code, CMS handles it without interrupting the execution of the CMS$EVENT_ACTION routine. On completion, the CMS$EVENT_ACTION routine returns a completion status code; this status code is signaled by CMS if it does not indicate success.

A user-provided CMS$EVENT_ACTION routine should not issue calls to callable CMS routines other than CMS$GET_STRING or CMS$PUT_STRING. Otherwise, a call issued by CMS$EVENT_ACTION might cause a new CMS event to occur, and possibly trigger an infinite chain of events. A user-provided CMS$EVENT_ACTION routine, however, can call the default CMS$EVENT_ACTION routine as part of its event-handling action.

8.2 Notification of Events

User notification is divided into two parts:

- Detecting and dispatching events
- Notifying users of those events by using the VMS Mail Utility (MAIL)

The default CMS$EVENT_ACTION routine determines the name of the current user (the user whose action caused the event). It then sends one or more notification messages through MAIL. You specify the recipients of the messages in the PARAMETER clause of the ACE defining the event. Each notification message is formatted as follows:

```
From:    <string specifying the current user>
To:      <string specifying the recipient>
Subj:    CMS notification for library <library name>

<message in a CMS history record format>
```

You use the SET ACL command to associate action ACEs with CMS library objects, and to define events involving these objects. For example:

```
CMS> SET ACL SPEC.RNO/OBJECT=ELEMENT/ACL=(NOTIFY=JOEUSER,ACCESS=MODIFY)
_Remark:  send notification if element modified
```

This command specifies that a notification message be sent to user JOEUSER on the local node each time SPEC.RNO is modified.

The text of a notification message is identical to the CMS history record written about the same event. Therefore, CMS notification allows users to receive selected history records through MAIL.

Note that transactions that are not logged in the library history and therefore have no history line (such as ANNOTATE, SHOW, and DIFFERENCES) do not cause an event.

The default CMS$EVENT_ACTION routine makes only one attempt to send each notification message. If the attempt fails, the specified event is not affected in any way. No record of failed MAIL messages is maintained, although the user whose action triggered the event receives any error messages incurred by the default CMS$EVENT_ACTION routine.

To avoid duplicate MAIL messages, you should define action ACEs such that only one event occurs as a result of a single CMS command being executed. Similarly, you should carefully select the recipients of notification messages to avoid unnecessary failed MAIL messages.

8.3 Examples

The following examples show how to use ACLs and notification on objects.

1.
   ```
   $  CMS SET ACL/OBJECT=ELEMENT EXAMPLE.PAS -
   _$       /ACL=(NOTIFY=LEADER,ACCESS=MODIFY) "notify project leader"
   ```

   This example specifies that the LEADER account is notified (through MAIL) when a user modifies the element EXAMPLE.PAS.

2.
```
$ CMS SET ACL/OBJECT=GROUP DATA_STRUCTURES -
_$      /ACL=(ACTION,PARAMETER="MYNODE::JONES",ACCESS= MODIFY)
_Remark:  notify when group DATA_STRUCTURES is modified
```

   This example specifies that when a user modifies the group DATA_STRUCTURES, CMS calls the default image SYS$LIBRARY:CMS$EVENT_ACTION.EXE (because you specified ACTION with no file specification) with the parameter MYNODE::JONES. CMS$EVENT_ACTION.EXE then notifies MYNODE::JONES that a user has modified the group DATA_STRUCTURES.

3.
```
$ CMS SET ACL/OBJECT=ELEMENT EXAMPLE.PAS -
_$      /ACL=(ACTION=CMSLOG_PRO,PARAMETER="NOTIFY.LOG",ACCESS=DELETE)
_Remark:  call event handler when element EXAMPLE.PAS is deleted
```

   This example specifies that when the element EXAMPLE.PAS is deleted, CMS calls the user-written event handler image CMSLOG_PRO, and passes the parameter NOTIFY.LOG. Note that the event handler routine name is specified without the .EXE file extension, and the NOTIFY.LOG parameter is enclosed in quotation marks because it contains a nonalphanumeric character (a period).

Chapter 9 Library Maintenance

CMS automatically performs maintenance on CMS libraries. You can also perform other types of maintenance to ensure a valid and responsive library.

This chapter presents information on library maintenance that CMS performs, and on the functions that CMS makes available for you to maintain your library and validate its integrity. It also provides some hints on dealing with library problems.

9.1 Command Rollback

If a CMS command is terminated before it has finished executing, CMS automatically initiates a process called command rollback. Rollback evaluates the state of the library and then takes appropriate action to return the library to a consistent state so that you can enter subsequent CMS commands.

Depending on the point at which the transaction is terminated, rollback takes the following actions:

- If the library contents have not been modified, rollback cancels the command.
- If the transaction is terminated before the update is complete, rollback cancels the command and restores the library to the state it was in before the command was entered. CMS closes and deletes any new files that were created in the library as a result of the command. In addition, rolling back a transaction involves restoring any files in the current default directory to the state they were in before the command was entered.

For example, if you run out of disk space during execution of a REPLACE command, CMS may not finish integrating the changes into the element file. In this case, rollback cancels the command, deletes any files that were placed in the library as a result of the command, and restores the library and your current default directory to the state they were in before the command was entered.

- If the library contents have been completely modified, restoration is not necessary. Rollback recognizes that the command has already been completed and takes no action. For example, a command may be terminated after execution but before control is returned to DCL command level or CMS subsystem level. In this case, the rollback mechanism determines that the command has been executed and rolling back the transaction is not necessary.

The following are examples of errors that can cause rollback:

- You press CTRL/Y and then enter a command (except STOP); this terminates the transaction. If you enter the DCL command CONTINUE after pressing CTRL/Y, the CMS command continues executing. This CTRL/Y CONTINUE sequence works the same as with any DCL command.
- A system-generated error occurs (such as running out of disk space).
- Certain CMS errors occur, causing CMS to issue an error message.
- CMS is terminated by a VMS exception condition.

CMS cannot initiate command rollback under the following circumstances:

- You press CTRL/Y and then enter the DCL command STOP.

CAUTION

> Never abort the CMS process by pressing CTRL/Y, then entering the DCL command STOP. CMS cannot perform rollback under this circumstance. To abort CMS, press CTRL/Y and enter the DCL command EXIT. This allows CMS to roll back the library into a usable state.

- The system is shut down during the execution of a command.
- There is a system failure as a result of a hardware or software error.
- An error occurs during the rollback process itself.

If one of these errors occurs, you must restore the library with the VERIFY/RECOVER command (see Section 9.2.1). CMS informs you if issuing VERIFY/RECOVER is necessary.

9.2 Verifying Data in a CMS Library

The VERIFY command checks your CMS library to confirm that the library structure and library files are in a valid form. If you use the VERIFY command under normal conditions, the command executes successfully, and VERIFY returns a success code. A successful VERIFY command indicates that CMS considers the information in your CMS library to be valid.

However, as a result of certain occurrences (for example, a library file is manipulated by a program other than CMS, or the system fails), the data in a CMS library may not be valid. In these cases, when you issue VERIFY, CMS detects the corruption, and VERIFY returns an error code.

The VERIFY command checks the following conditions:

- The library must be set to a valid CMS library directory or a list of library directories.
- The last CMS command entered on the library must have finished executing (if it did not, CMS attempts automatic recovery before continuing).
- All library control files (00CMS.type) that should be in the library are present and accessible.
- The element, reference copy, class, and group information, reservation information, command list, security information, and internal database structures are in a valid format.
- All element files have been manipulated only by CMS.
- All element files have valid checksums (see Section 9.2.2) indicating that data has not been lost from or added to the files.
- Only element files and other files used by CMS are present in the library (that is, there are no nonelement and no non-CMS files).
- All element files that should be there (one for each element) are present.

If the last transaction was prematurely terminated and was not automatically rolled back, use the VERIFY/RECOVER command. If any file in the library was not closed by CMS or if the checksum for one or more files is invalid or missing, use the VERIFY/REPAIR command. When you use VERIFY/REPAIR, you must be sure that data has not been lost or added. See Section 9.2.3 for more information.

You cannot use the /RECOVER and /REPAIR qualifiers on the same VERIFY command. If conditions exist that call for the execution of both VERIFY/RECOVER and VERIFY/REPAIR, you must enter VERIFY/RECOVER first, then VERIFY/REPAIR.

The following sections describe the /RECOVER and /REPAIR qualifiers in detail.

9.2.1 Using VERIFY/RECOVER

Most CMS commands update several files in the library. If a command is terminated while it is updating the library, the library can be left in a state in which some files have been modified and others have not. Usually, if a command is terminated prematurely, the rollback mechanism cancels and rolls back the transaction (see Section 9.1). If CMS cannot roll back the library, you must use the VERIFY/RECOVER command to restore the library to a consistent state.

If you terminate a command at a time when the files in the library may have been left in an inconsistent state, CMS recognizes that the command execution was incomplete. When any user tries to enter a subsequent CMS command to the same library, CMS attempts automatic recovery. If automatic recovery fails, CMS advises the user to enter VERIFY/RECOVER. In this case, users cannot access the CMS library until VERIFY/RECOVER has been executed.

The VERIFY/RECOVER and VERIFY/REPAIR commands use earlier versions of files in the library to restore the library. You should not delete or purge any files from the library, because CMS performs its own cleanup functions.

The VERIFY/RECOVER command cancels only the previous transaction. If the event that causes the premature termination (for example, a system failure) also corrupts data in the library (that is, data stored in files that were present before the event), you must use other means to restore the library. VERIFY/REPAIR corrects some of the unusual occurrences within a CMS library (see Section 9.2.2). CMS may inform you if library repair is necessary after certain commands are issued. In this case, you receive the following message:

```
%CMS-E-USEREPAIR, use VERIFY/REPAIR
```

The VERIFY/RECOVER command affects only the currently set CMS library or libraries, not your default directory. An incomplete transaction may mean that the process of moving files into your directory or deleting files from your directory is incomplete. You must recognize these conditions yourself and, if necessary, remedy them with CMS or DCL commands.

For example, the REPLACE command generally uses a file from your current default directory to update the element file. If the system fails during a replacement transaction, the process of updating the library file may be incomplete. CMS never deletes any files from your directory until a transaction is complete. In this case, you would need to enter the VERIFY /RECOVER command to cancel the transaction. The file that was being copied would still be in your current default directory. Another REPLACE command creates a new generation as you originally intended.

If you have set up a restrictive file protection scheme and there is a system failure during a CMS transaction that leaves your library in an inconsistent state, a user with sufficient access to the library and its files should execute the VERIFY/RECOVER command. You can also recover the library if you have BYPASS privilege, or read, write, and execute access to all the library files. For more information, see Chapter 7.

The following commands do not update the library and thus cannot leave the library in an inconsistent state:

ANNOTATE
DIFFERENCES
FETCH (no remark)
RETRIEVE ARCHIVE
SET LIBRARY
SET NOLIBRARY
SHOW commands
VERIFY (no qualifiers)

9.2.2 Using VERIFY/REPAIR

You use the VERIFY/REPAIR command when the VERIFY command informs you of one of the following conditions:

- Element data files in the library were not closed by CMS.
- The checksum of elements in the library is invalid.
- Generations in the library have an invalid maximum record size.
- The last recorded transaction time is greater than the current system time.
- The reference copy for an element is missing.
- A reference copy is found for an element with the /NOREFERENCE_ COPY qualifier.
- There are duplicate reference copies for an element.

- The reference copy is invalid.

CMS uses information in the file header of a library file to confirm that the file was closed by CMS. If the file was not closed by CMS (for example, if it was opened and closed with a text editor), VERIFY/REPAIR repairs the file header so that it can be successfully verified.

For each element, CMS maintains a number known as a checksum. A checksum is a count that varies with the number of characters and the value of the characters in a file. Every time CMS writes a file in the library, the checksum is recalculated. The VERIFY command calculates the checksum for every element in the library. If this checksum does not equal the stored value, data has probably been lost from, added to, or changed in the file.

The VERIFY/REPAIR command corrects a bad checksum by recalculating the value based on the current contents of the file and then storing this value. The contents of the file are not altered. If you know that data has been lost from or added to the element, you must correct it manually. See Section 9.2.3 for more information.

The VERIFY/REPAIR command adjusts element generations that were created from files with fixed-length records by earlier versions of CMS and have a stored maximum record size of zero. VERIFY/REPAIR examines the element data file, determines what the correct size should be, and stores this value with the generation.

The VERIFY/RECOVER and VERIFY/REPAIR commands use earlier versions of files in the library to restore the library. You should not delete or purge any files from the library, because CMS performs its own cleanup functions.

9.2.3 Correcting Errors

If a program other than CMS has been used to manipulate the files in the CMS library, you may receive the following error message:

```
%CMS-E-VEREDFERR, element DISKX:[PROJECT.CMSLIB]TEST.SDML verified with errors
-CMS-E-NOTBYCMS, data file DISKX:[PROJECT.CMSLIB]TEST.SDML;1 not closed by CMS
```

If no other errors accompany this message, CMS considers the contents of the file valid despite manipulation from the outside program. In this case, you can use the VERIFY/REPAIR command to correct any errors (however, you should always investigate your source file to ensure that your file is still valid). Some examples of what can cause these errors are as follows:

- Entering the DCL command SET PROTECTION or SET FILE/PROTECTION

- Entering the DCL command SET ACL or SET FILE/ACL
- Restoring your CMS library from backup
- Entering the DCL command COPY

Other programs (such as a text editor) can also cause this error.

CMS may also issue the following error message:

```
%CMS-E-BADCRC, bad checksum in element
```

This error is usually accompanied by the CMS-E-NOTBYCMS error. A bad checksum indicates that the contents of the element data file are different from what CMS expects. This usually means that data in the file has been corrupted. Corruption can occur if something has changed the contents of the element data file; this can happen if you alter the element data file, or if a previous version of the element data file was restored from backup. Corruption can also occur if the library directory contains a revision of the CMS database (00CMS.CMS) that does not correspond to the element data file. This typically occurs if the 00CMS.CMS file was restored from backup, but the rest of the library contains more recent versions of element files and was not restored.

You can use the VERIFY/REPAIR command to correct BADCRC errors. If CMS finds more than one version of the element file, it keeps the version containing the correct checksum, and deletes the other files. If no file exists with the correct checksum, VERIFY/REPAIR records the checksum from the most recent file, and deletes any other copies. CMS can then use that value for future checks. CMS does not attempt to alter the contents of the file.

You should use VERIFY/REPAIR to correct BADCRC errors only if you understand the source of these errors and the potential impact of repairing them.

9.2.4 Reference Copies

If a library has a reference copy directory, the VERIFY/REPAIR command performs a comparison between the reference copy and the latest generation on the main line of descent for each element in the library.

If CMS finds a reference copy for an element that does not have the **reference copy** attribute, it prompts you for confirmation, then deletes the reference copy file.

If the **reference copy** attribute is enabled for an element and you enter the VERIFY/REPAIR command, one of the following situations may occur:

- If there is no valid reference copy in the reference copy directory, CMS prompts you for confirmation to delete the remaining copies, then fetches the latest main-line generation (1+) into the reference copy directory.
- If there is more than one reference copy and there is at least one valid copy, CMS keeps the valid copy (or the latest valid generation, if more than one valid copy exists) in the reference copy directory, and deletes the remaining copies.
- If the reference copy does not exist, CMS fetches the latest main-line generation (1+) into the reference copy directory.

9.3 Maintaining Library Efficiency

The following sections discuss the features that CMS provides to allow you to maintain the contents of your CMS library.

9.3.1 Deleting History Records

CMS maintains a history file in which all operations that modify the library are recorded. Each operation causes a single record (or one record for each item, when wildcards have been used) to be written into the 00CMS.HIS control file. As libraries get older, history files typically become quite large, taking up disk space and causing SHOW HISTORY performance to degrade. Because very old history is generally no longer useful, you can use the DELETE HISTORY command to reduce the size of the file.

Element generation information (for example, as displayed with the commands SHOW GENERATION, FETCH/HISTORY, and ANNOTATE) is part of each generation and is not stored in the history file; therefore, it is not affected by the deletion of the library history.

9.3.2 Deleting and Archiving Element Generations

When you enter a FETCH, RESERVE, or REPLACE command, CMS searches all of the generations of a specified element for the generation you are trying to access. As libraries get older, the number of generations usually increases, and CMS commands that operate on element generations respond more slowly.

You can alleviate this problem by deleting the generations of an element that you no longer need. For example, if you have an element with 100 generations, and generation 5 was released in version 1 of your product, generation 30 was released in version 2, generation 43 was released in version 3, and you are currently developing version 4, you probably do not need to reproduce generations prior to 43, with the exception of those specific generations that went into the released versions. You can use the DELETE GENERATION command to remove the unneeded generations (see the Command Dictionary for more information).

When you delete a generation, the definition of the generation is permanently removed from the corresponding element in the CMS library. Deleting a generation does not remove changes from subsequent generations that were originally made in the deleted generation. If you delete a generation from the end of a line of descent, all the changes representing that generation are removed from the delta file (see Section 4.4 and Appendix C). If you remove a generation from the middle of a line of descent, changes made in that generation are propagated into the surviving descendant and combined or eliminated from the delta file if possible, because later generations still depend on those changes. You should not rely on generation deletion to reduce the size of a delta file.

If you want to delete an element generation from the CMS library but may still want to access the contents of that generation, you can use the /ARCHIVE qualifier on the DELETE GENERATION command. This qualifier directs CMS to create an archive file containing all the information from the deleted generation.

The archive file is self-contained; you do not need a CMS library to restore the contents of the file. The archive file exists outside of the CMS library and can be backed up onto tape and deleted. You can use the SHOW ARCHIVE command to display the contents of an archive file; you can use the RETRIEVE ARCHIVE command to retrieve a copy of any of the generations in an archive file. You cannot restore a generation from the archive directly into the CMS library. To restore the generation, you must retrieve the generation into a file, use the RESERVE command to reserve a generation of the element in the library, and then use the REPLACE command to replace the reservation, using the retrieved file as input.

Although the VERIFY command does not operate on archive files, the files store a checksum of the information in the file. The RETRIEVE ARCHIVE command issues a warning message if it finds that the checksum of the data in the file does not match the stored checksum. An incorrect checksum does not prevent you from accessing the data in the file, but it may indicate that the file is corrupt. In this case, you should restore another copy from backup.

9.4 Unusual Occurrences

An unusual occurrence results from the execution of a CMS command that might, at times, have undesirable consequences. An unusual occurrence is always logged in the library history file. The following actions cause CMS to record an unusual occurrence:

- Entering a RESERVE command that creates a concurrent reservation
- Entering a REPLACE command that creates a concurrent replacement
- Entering a REPLACE or UNRESERVE command where BYPASS access was used to manipulate another user's reservation
- Entering the VERIFY/REPAIR command
- Entering the VERIFY/RECOVER command
- Entering the CONVERT LIBRARY command
- Entering the REMARK/UNUSUAL command

The SHOW HISTORY/UNUSUAL command displays the records of transactions that caused unusual occurrences. CMS identifies unusual occurrences in the library history by displaying an asterisk in the first column of the transaction record.

When the RESERVE or REPLACE command produces an unusual occurrence, CMS informs you of the potential unusual occurrence and asks whether you want to proceed. If you answer YES, the command is executed and the transaction is recorded as an unusual occurrence.

The VERIFY/RECOVER and VERIFY/REPAIR commands are logged as unusual occurrences because they are entered when something is wrong with the CMS library structure or its files. If you enter VERIFY/RECOVER or VERIFY/REPAIR on a valid library, or if you enter the VERIFY command without qualifiers, CMS does not log an unusual occurrence.

Chapter 10

Command Syntax

This chapter describes how to enter CMS commands and gives the syntax for command parameters, qualifiers, remarks, and wildcard characters.

10.1 Command Format and Prompting

The general format of a CMS command is as follows:

```
command [keyword] [parameter] [/qualifier...] [remark]
```

A CMS command consists of the name of the command, and a keyword if it is required by the syntax of the command. For example, the RESERVE command consists of only the command name. The SHOW command requires a keyword, for example, HISTORY. In general, you must use one or more spaces or tabs to separate items in a command string. Spaces or tabs preceding a qualifier are optional.

The formats of parameters, remarks, and qualifiers are described in Sections 10.2, 10.2.2, and 10.4, respectively.

A CMS command string can consist of 1024 characters if you use hyphen continuation characters (-). The command can contain any printing characters, spaces, and tabs.

CMS compresses multiple spaces and tabs to a single space (except in quoted strings). You can enter CMS commands in either lowercase or uppercase characters. CMS changes lowercase characters to uppercase (except in quoted strings). As a result, all commands recorded in the library history are in uppercase characters.

If you enter a command that requires a parameter and you do not specify one, CMS prompts you for one. Note, however, that if you use CMS in batch mode or in a command procedure, CMS does not prompt for missing items.

Some commands may require confirmation after you enter the command. In these cases, you are prompted for a YES or NO answer. In some cases, you can also supply one of the following responses:

Positive Response	Negative Response
1	0
TRUE	FALSE
ALL	QUIT or CTRL/Z

Typing ALL indicates that CMS should perform the action (or actions) specified by the command without any confirmation (for example, after the INSERT GENERATION command). Typing QUIT or pressing CTRL/Z indicates that CMS should not perform any actions specified by the command.

If you press RETURN, CMS uses the default, indicated in brackets ([]). Note that CMS checks only the first character of each confirmation response. Thus, typing YAHOO is equivalent to typing YES or Y. If you type any other characters, CMS continues to prompt you until you type an acceptable response.

To halt the execution of a CMS command, press CTRL/C. CTRL/C indicates that CMS should terminate the processing of that command. For more information on using CTRL/C, see Chapter 9.

10.2 Command Parameters

This section describes the parameters that can be used with CMS commands:

- Directory specifications
- Remarks
- Element names
- Element expressions
- Element generations
- Element generation expressions
- Group names
- Group expressions
- Class names

- Class expressions

In addition, you can use wildcard expressions as parameters to certain CMS commands. Wildcard expressions are described in Section 10.5.

10.2.1 Directory Specifications

You use a directory specification to refer to a directory that contains (or will contain) a CMS library or a reference copy directory. A directory specification is used as a parameter to the CREATE LIBRARY, SET LIBRARY, and SET NOLIBRARY commands, and as a qualifier value to the COPY ELEMENT command. In addition, it is a parameter to the DCL command CREATE/DIRECTORY, which is used to create a directory that will contain a CMS library (see Chapter 3) or a reference copy directory.

The format of a directory specification is as follows:

```
disk:[directory]
```

disk
Specifies one or more disks where the directory that contains your CMS library is located. If you omit the disk name, your current default disk is assumed.

directory
Specifies a directory that contains your CMS library. Directory names must be enclosed in square brackets ([ ]). Wildcards are not allowed.

For more information on how to specify disk and directory names, see the *VMS DCL Concepts Manual.*

Example

```
$  CMS SET LIBRARY [SWIFT.CMSLIB]
```

This example specifies the subdirectory CMSLIB under the top-level directory [SWIFT] on the current default disk.

10.2.2 Remarks

A *remark* is a character string that you supply to describe a transaction. All CMS commands that modify the library or its contents allow you to enter a remark, which is recorded in the library history as part of the transaction record. Remarks are useful in tracking modifications to a library element. For example, in the remark given on the REPLACE command, you could indicate what changes were made to the element for which you are creating a new generation. For example:

```
CMS> REPLACE DATAFIG3.SDML "updated figure to show new merge routine"
```

For the purpose of command-line interpretation, remarks are defined as parameters; thus, you can enter qualifiers after the remark. However, the remark must be the last parameter entered on the command line. Because remarks are defined as parameters, CMS attempts to translate the remark if other parameters are missing or incorrectly placed. If, for example, you omit an element name from the syntax of a command, but you enter a remark, CMS assumes that the remark is intended as the name of an element.

Quotation marks ("") are required to enclose the remark if you enter it on the same line as the command and the remark contains any spaces. For example, a one-word remark entered on the command line does not require enclosing quotation marks. The text can consist of any printing characters, spaces, and tabs. If you press CTRL/Z as part of a remark, it terminates the command input at that point, and CMS executes the command. If you press CTRL/C as part of a remark, CMS cancels the command. To insert a quotation mark (") within a remark, type it twice (""). If a remark consists only of two consecutive quotation marks (""), the remark text is null.

If you omit a remark on the command line of a command that requires a remark, CMS prompts you for the text of the remark on the next line. For example:

```
CMS> REPLACE DATAFIG3.SDML
_Remark:  updated figure to show new merge routine
```

Type the text of the remark immediately following the prompt. In this case, you need not enter quotation marks unless you want them to be included in the text of the remark. If you press RETURN in response to the prompt, you are not reprompted, and the remark text is entered as null.

When you start the remark on the same line as the CMS command, the total length of the remark (including quotation marks), added to the character count for the rest of the command, cannot exceed 256 characters. When you enter the remark in response to the prompt, the length of the remark cannot exceed 254 characters.

You cannot use the hyphen continuation character (-) to continue a remark. If you type a hyphen within a remark and then press RETURN, the hyphen becomes the last character in the logged remark. The closing quotation marks are assumed. To continue a remark, type the remark until the text wraps to the next line.

Examples

1.
```
CMS> REPLACE SYNTAX.PAS "RECORD declaration implemented"
```

Note that a blank must precede the first quotation mark in a remark. The remark, including the quotation marks, is recorded as part of the record of the REPLACE transaction in the project history.

2.
```
CMS> FETCH SEMANTICS.PAS
_Remark:  Get copy for code review
```

If you press RETURN before you enter a remark, CMS prompts for the remark. The remark is recorded in the project history. It looks the same as if the remark had been entered on the same line as the rest of the command (CMS encloses the remark in quotation marks).

3.
```
CMS> FETCH LEXICAL.PAS "check alternate two-character graphic impl
ementation for demo version of front end"
```

You cannot use the DCL continuation character (-) to continue the remark; you must continue typing until the text wraps to the next line.

10.2.3 Element Names

You name an element by specifying it as the parameter to the CREATE ELEMENT command.

The format of an element name is as follows:

filename.type

filename
Specifies the file-name component of a VMS file specification. The file name can be 0 to 39 characters. The file-name component must begin with an alphanumeric character. For a list of the characters that you can use in a file name, see the *VMS DCL Concepts Manual*.

type
Specifies the file-type component of a VMS file specification. The file type can be 0 to 39 characters. For a list of the characters that you can use in a file type, see the *VMS DCL Concepts Manual*.

Separate the file name and the file type with a period (.). An element name must contain a single period even if the file type or file name is null. Spaces and tabs are not legal element name characters.

NOTE

> Within a library, all element names must be unique. The file name component cannot be 00CMS because that name is reserved for CMS.

The following are examples of valid element names:

```
TEST.BAS
SAMPLE.SDML
ARGCHK.COM
MOD5.S
```

10.2.4 Element Expressions

An element expression lets you name multiple instances of an element in a single parameter field.

An element expression is composed of one or more of the following:

- An element name
- A group expression
- A wildcard expression (a wildcard character, or a wildcard character used in combination with a name or partial name)
- A list of the preceding items, with the items separated by commas

If you specify an element name, CMS manipulates a single element. If you specify more than one element name separated by commas or if you specify a group, a wildcard expression, or a combination of these, CMS operates on one or more elements. For example:

```
CMS>  SET LIBRARY [JONES.CMSLIB]
CMS>  CREATE ELEMENT ELE.SDML,*.LIS,DATASAM.PAS "element list"
```

This command sets the current CMS library to [JONES.CMSLIB] and creates the element ELE.SDML, all elements with a file type of .LIS, and the element DATASAM.PAS. These elements are created from files in your default disk and directory.

You must include a period (.) in the element expression to select one or more elements from the complete list of elements in the library. If you do not include a period, CMS interprets the parameter as a group name and selects elements based on the list of groups that are established in the library.

10.2.5 Element Generations and Expressions

An element generation is a specific version of an element. Each time you reserve and replace a version of an element in the library, CMS creates a new generation of that element. The first generation of an element is generation 1. Each element generation is assigned a unique generation number; by default, subsequent generations are numbered sequentially by adding 1 to the predecessor generation number.

You can create a variant generation number from an existing generation number by appending a variant letter to the existing generation number and starting a new level number sequence beginning at 1. For example, the generation 7A1 could be a variant generation of generation 7.

The syntax of a generation number is as follows:

```
level-number [variant-letter level-number]...
```

level-number	A positive integer. Leading zeros are not allowed.
variant-letter	A single alphabetic character (a through z, A through Z).

An element generation expression allows you to specify a particular generation of an element. You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. These methods can be combined or used separately.

The format of a generation expression is as follows:

$$\left\{ \begin{array}{l} \text{generation-number [+]} \\ \text{class-name [+]} \end{array} \right\} \text{[[;] relative-generation-offset]}$$

generation-number
Specifies a unique element generation.

class-name
Specifies a CMS class name according to the syntax rules in Section 10.2.8. If a class name value is given, the generation specification refers to the generation in the specified class.

+
Indicates the plus operator. CMS locates the latest generation on the same line of descent as the generation specified by the generation number or the class name.

;
Required to separate the relative generation offset from the generation specification. The semicolon is not allowed in cases where a generation number or class name has been omitted and CMS supplies a default value.

relative-generation-offset
Specifies an integer that directs CMS to locate an ancestor or direct descendant of the specified generation. If the relative generation number is negative, then CMS locates an ancestor generation. If the relative generation number is positive, then CMS locates a direct descendant. The absolute value of the relative generation number indicates how many steps should be taken to the next existing ancestor or descendant generation. A relative generation offset of zero has no effect.

If generations have been deleted, CMS selects the third existing generation prior to the generation you specified. For example, assume the current generation of SAMPLE.PAS in class VERSION1 is generation 7, and generations 5 and 6 have been deleted on the main line of descent for SAMPLE.PAS (thus, the line of descent appears as 1, 2, 3, 4, 7).

Examples

Assume the element SEMANTICS.PAS has six generations on the main line of descent. In addition, a variant line consists of generations 3C1 and 3C2. Generation 5 belongs to the class VERSION1. The following examples show valid forms of the /GENERATION qualifier for the element SEMANTICS.PAS.

1. `SEMANTICS.PAS/GENERATION=4`

   This reference selects generation 4 of SEMANTICS.PAS.

2. `SEMANTICS.PAS/GENERATION=3C1+`

   This reference selects the latest generation (generation 3C2) on variant line C that extends from generation 3 on the main line of descent. You can use this form if you know a variant line exists and want the most recent generation, but do not know how many generations are on that line.

3. `SEMANTICS.PAS/GENERATION=VERSION1`

   This reference selects the generation of SEMANTICS.PAS (generation 5) that belongs to the class VERSION1.

4. `SEMANTICS.PAS/GENERATION=VERSION1;-3`

   This reference uses a relative generation offset of –3 to select the third generation of SEMANTICS.PAS before the generation that is in class VERSION1. In this example, CMS locates generation 2 of SAMPLE.PAS.

10.2.6 Group Names

You name a group by specifying it as the parameter to the CREATE GROUP command. A group name can be up to 39 characters long, and can contain any of the following characters:

- Letters and digits (a through z, A through Z, and 0 through 9)
- Dollar signs ($)
- Hyphens (-)
- Underscores (_)

A group name must begin with an alphabetic character. Group names cannot contain a period (.) because CMS interprets a group name containing a period as an element name. You cannot use the same name for both a group and a class in the same library. The following are examples of valid group names:

```
GRAPHICS
DATA_IN
DATA$OUT
CREATE-MODULES
```

10.2.7 Group Expressions

A group expression lets you name one or more multiple instances of a group in a single parameter field. A group expression is composed of one or more of the following:

- A group name
- A wildcard expression (a wildcard character, or a wildcard character used in combination with a name or partial name)
- A list of the preceding items, with the items separated by commas

If you specify a group name, CMS operates on a single group. If you specify more than one group name separated by commas or a wildcard expression, CMS operates on one or more groups. The following are examples of valid group expressions:

```
GROUPA
*88
MAIN$MODULES
PHASE_*_DOCS
```

10.2.8 Class Names

You name a class by specifying it as the parameter to the CREATE CLASS command. A class name can be up to 39 characters long, and can contain any of the following characters:

- Letters and digits (a through z, A through Z, and 0 through 9)
- Periods (.)
- Underscores (_)
- Dollar signs ($)
- Hyphens (-)

A class name must begin with an alphabetic character. You cannot use the same name for both a class and a group in the same library. The following are examples of valid class names:

```
BASE_LEVEL3
DEMO.1
VERSION$A
FIELD-TEST
```

10.2.9 Class Expressions

A class expression lets you name multiple classes in a single parameter field. A class expression is composed of one or more of the following:

- A class name
- A wildcard expression (a wildcard character used in combination with a name or partial name)
- A list of the preceding items, with the items separated by commas

If you specify a class name, CMS operates on a single class. If you specify more than one class name separated by commas or a wildcard expression, CMS operates on one or more classes. The following are examples of valid class expressions:

```
VERSION1
BASELINE*
FIELD_TEST
DEMO.%
```

10.3 Comma Lists

Where a comma list is valid, you can specify more than one value for a parameter, separated by commas, on the command line. For example:

```
CMS> DELETE GROUP USER_VIEW,USER_INTFACE,TESTGRP
_Remark:  groups no longer necessary--superseded by field test
%CMS-I-DELETED, group DISKX:[PROJECT.CMSLIB]USER_VIEW deleted
%CMS-I-DELETED, group DISKX:[PROJECT.CMSLIB]USER_INTFACE deleted
%CMS-I-DELETED, group DISKX:[PROJECT.CMSLIB]TESTGRP deleted
%CMS-S-DELETIONS, 3 deletions completed
```

This command deletes the three groups USER_VIEW, USER_INTFACE, and TESTGRP. The same remark is logged in the history for each of these groups.

To cancel a comma list transaction before it has completed, press CTRL/C. If you press CTRL/C during a transaction using a comma list, CMS finishes the immediate transaction, but does not continue. For example, if you are replacing several elements and you press CTRL/C during the replacement of the first element, CMS finishes that replacement transaction but does not continue with the others.

When you enter a command using a comma list from DCL command level and then press CTRL/C during execution of the command, CMS returns control to DCL. If you enter the command from the CMS subsystem prompt level, control is returned to CMS.

10.4 Command Qualifiers

Command qualifiers always start with a slash character (/) and may or may not require a value. A command qualifier, if used, must follow the command (and the keyword, if any). Qualifiers can appear before or after any parameters that are specified on the command line, except when you use the /GENERATION qualifier with the DIFFERENCES command (see the description of the DIFFERENCES command in the Command Dictionary).

You can enter qualifiers after remarks. A command qualifier has the same meaning whether it follows the command name or a command parameter.

For example, the following two commands are equivalent:

```
$  CMS CREATE ELEMENT/KEEP CODEGEN.PAS ""
$  CMS CREATE ELEMENT CODEGEN.PAS/KEEP ""
```

The /KEEP qualifier specifies that the file CODEGEN.PAS is not to be deleted from the user's directory.

Many qualifiers on CMS commands have both a positive and a negative form. For example, /APPEND and /NOAPPEND are the positive and negative forms of the same qualifier.

If you specify the same qualifier more than once on a command or specify both the positive and negative form of the same qualifier, CMS uses only the last specification. For example:

```
$  CMS FETCH INIT.FOR/OUTPUT=TEST.FOR/OUTPUT=INITEST.FOR
```

If you enter this command, CMS uses the second output file specification (INITEST.FOR).

10.4.1 Qualifier Values

Various CMS command qualifiers require quoted strings, file specifications, directory specifications, numeric values, alphabetic values, times, or generation expressions as qualifier values.

You must separate a qualifier and its value with either an equal sign (=) or a colon (:). Zero, one, or more spaces and tabs can appear between the qualifier and the separator, and between the separator and the value. For example, the following two specifications are equivalent:

```
/OUTPUT = TESTFE.COM
/OUTPUT: TESTFE.COM
```

The following sections describe file specifications and the format for entering dates.

10.4.1.1 File Specifications

Many CMS commands allow you to specify input or output files. These commands accept full VMS file specifications as qualifier values. If you do not enter a full file specification, CMS uses the current directory, device, or node. For a complete description of a file specification, see the *VMS DCL Concepts Manual*.

10.4.1.2 Time Values

Several commands allow you to specify time values with the /BEFORE and /SINCE qualifiers. Each of these qualifiers accepts an absolute, delta, or combination time value. You can also specify one of the following keywords: YESTERDAY, TODAY, or TOMORROW.

An absolute time is a specific date or time of day, or both. A delta time value is the difference between the current time and a future time. A combination time consists of an absolute time value plus or minus a delta time value. For detailed information about time values, see the *VMS DCL Concepts Manual.*

10.4.2 Qualifier Defaults

Each command description in the Command Dictionary contains a list of qualifiers and qualifier defaults. The default indicates the action taken when you omit the qualifier.

Qualifiers with simple positive and negative forms (those that do not take qualifier values) are listed in the command format sections with their defaults. For example:

```
/[NO]APPEND                /NOAPPEND
```

On the left, the qualifier is listed with brackets ([]) around the optional part of the qualifier (NO). On the right, the default is listed.

Some qualifiers have a positive form that allows a qualifier value, and a negative form that does not allow the value. These qualifiers are shown with their defaults. For example:

```
/MERGE=generation-exp      /NOMERGE
/NOMERGE
```

If you use the positive form, the generation expression is required. If you use the negative form, which is the default, the generation expression is not allowed.

The defaults (if any) for qualifier values are explained in the qualifier descriptions and are also indicated by the letter D next to the qualifier name.

10.5 Wildcard Expressions

You can use DCL wildcard expressions in the parameters for many CMS commands. The wildcard characters are the percent sign (%) for single-character substitution and the asterisk (*) for partial- or full-field substitution. By using these wildcards, you can direct CMS to operate on more than one element, group, or class at a time. In addition, you can use wildcards in input and output file specifications, and the directory-searching wildcards (the ellipsis (. . .) and the minus sign (–)) in input file specifications.

For elements and generations, wildcards can apply to either the file-name field or the file-type field, according to the position they occupy.

The following sections describe general rules for using wildcards.

10.5.1 Single-Character Wildcards

The percent sign (%) is the single-character wildcard indicator. When you use the percent sign in a command parameter, CMS selects elements, groups, or classes by substituting any single, allowable character for the percent sign.

For example, the wildcard expression DATA%.FOR might result in the following list of elements:

```
DATA1.FOR
DATA2.FOR
```

10.5.2 Partial- and Full-Field Wildcards

The asterisk (*) is the partial- and full-field wildcard indicator. When you use an asterisk in a command parameter, CMS selects objects whose names contain the character patterns given in the wildcard expression. CMS replaces the asterisk with any number of allowable characters (within the range of zero to the maximum size of the field).

For example, the element expression DATA*.FOR might result in the following list of elements:

```
DATA.FOR
DATA1.FOR
DATA2.FOR
DATA_IN.FOR
DATA_OUT.FOR
```

10.5.3 Canceling Wildcard Transactions

To cancel a wildcard transaction before it has completed, press CTRL/C. If you press CTRL/C during a wildcard transaction that updates the library, CMS finishes the immediate transaction, but does not continue. For example, if you are replacing several elements and you press CTRL/C during the replacement of the first element, CMS finishes that replacement transaction but does not continue with the others.

When you enter a wildcard command from DCL command level and then press CTRL/C during execution of the command, CMS returns control to DCL. If you enter the command from the CMS subsystem prompt level, control is returned to CMS.

10.6 Command Abbreviations

You can abbreviate command, keyword, and qualifier names by eliminating characters from the end of the specified command, keyword, or name. You cannot truncate the string "CMS" when entering a CMS command at DCL level. All commands and qualifiers are unique when truncated to their first four characters. You can truncate these names to fewer than four characters as long as the result is unique.

For example, VERIFY is the only CMS command that begins with the character V. Therefore, the VERIFY command can be truncated to CMS V at DCL level and V at CMS subsystem level.

You do not count the slash character (/) or the prefix NO on negative qualifiers when you count characters to determine the shortest allowable form of a qualifier. However, you must count the underscore (_) character.

Command Dictionary

This section contains a detailed description of each CMS command. The commands are arranged in alphabetical order. Each command description contains the following:

- General format of the command
- List of command qualifiers
- Restrictions on the use of the command, if applicable
- Descriptions of each parameter, if applicable
- Description of the command
- Descriptions of each command qualifier, and the qualifier default (qualifier defaults are marked with a D)
- Examples

ACCEPT GENERATION

Changes the review status of each specified element generation from pending to accepted and removes it from the review pending list.

Format

ACCEPT GENERATION *element-expression "remark"*

Command Qualifiers	Defaults
/[NO]CONFIRM	/NOCONFIRM
/GENERATION[=generation-expression]	See text
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL

Command Parameters

element-expression
Specifies one or more elements. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The ACCEPT GENERATION command changes the review status of each specified element generation from pending to accepted and removes it from the review pending list.

ACCEPT GENERATION

Use this command only on element generations that have reviews pending (see the description of the REVIEW GENERATION command for more information). If you access the generation once it has been accepted, CMS no longer informs you of any review status.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/GENERATION[=generation-expression]
Specifies a particular generation of the element to be accepted. If you omit /GENERATION, CMS accepts the most recently created generation with a review pending. You specify this qualifier only if more than one generation of an element is under review.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field

contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, CLASS)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
CLASS (D)
NOCLASS
NONE —equivalent to (NOELEMENT, NOGROUP, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

Examples

```
CMS> ACCEPT GENERATION EXAMPLE.SDML "this example cleared for publication"
%CMS-S-ACCEPTED, generation 3 of element DISKX:[PROJECT.CMSLIB]EXAMPLE.SDML accepted
```

This command accepts the most recently created generation of the element EXAMPLE.SDML. The generation is removed from the review pending list.

ANNOTATE

Creates a line-by-line file listing of the changes made to each specified element generation and places it in your current default directory or a specified directory.

Format

ANNOTATE *element-expression*

Command Qualifiers	Defaults
/[NO]APPEND	/NOAPPEND
/[NO]CONFIRM	/NOCONFIRM
/FORMAT=(data-format,data-partition)	/FORMAT=(ASCII,RECORDS)
/FULL	See text
/GENERATION[=generation-expression]	/GENERATION=1+
/[NO]LOG	/LOG
/MERGE=generation-expression	/NOMERGE
/NOMERGE	
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/OUTPUT[=file-specification]	/OUTPUT=element-name.ANN

Command Parameter

element-expression
Specifies one or more elements. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas. If you specify a group name, CMS annotates each element in the group. If you use wildcards, CMS produces one annotated listing file for each matching element. By default, the most recent generation of an element on the main line of descent is annotated.

Description

The ANNOTATE command documents the development of an element. This command creates an output file that contains an annotated listing; by default, the file name is the same as the element name and the file type is .ANN. The annotated listing file contains two parts:

- A history
- A source file listing

The history includes the generation number, date, time, user, and remark associated with each generation of the element (and other file-related information when you use the /FULL qualifier). The generations are listed in reverse chronological order. The generation numbers of the specified generation and its ancestors are marked with an asterisk (*).

The source file listing contains all the lines inserted or modified from generation 1 to the specified generation. The listing does not show lines deleted from the file. CMS inserts consecutive line numbers in the listing unless editor-assigned line numbers already exist. (The line numbers start with 1 for the first line and increase by 1 for each line.) The generation field starts at the first character position of each line. It contains the generation number of the most recent generation in which the line was inserted or modified. The generation field is blank if a line is unchanged since generation 1.

Command Qualifiers

/APPEND
/NOAPPEND (D)
Controls whether CMS appends the history and source file listing to an existing file, or creates a new file. If you specify /APPEND and the output file does not exist, CMS creates a new file. If you do not provide an output file specification (see the description for /OUTPUT), CMS searches your default directory for a file with the element file name and the file type .ANN.

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/FORMAT=(data-format,data-partition)
/FORMAT=(ASCII,RECORDS) (D)
Controls whether the history and source file listing is formatted, and specifies the type of formatting. You can specify one or both formatting parameters in any order.

data-format
Specifies the type of format. The following table lists the possible values for data formats:

Data Format Option	Action
ASCII (D)	Specifies that each byte of the data be displayed as an ASCII character. This option is most useful when files contain textual data. If no data partition is specified, data is partitioned into records. This option is the default.
DECIMAL	Specifies that each value be displayed as a decimal numeral. If no data partition is specified, data is partitioned into longwords. You cannot specify both DECIMAL and RECORDS.
HEXADECIMAL	Specifies that each value be displayed as a hexadecimal numeral. If no data partition is specified, data is partitioned into longwords. You cannot specify both HEXADECIMAL and RECORDS.
OCTAL	Specifies that each value be displayed as an octal numeral. If no data partition is specified, data is partitioned into longwords. You cannot specify both OCTAL and RECORDS.

data-partition
Specifies the type of data partition. A data partition is the size that data in each record is to be broken into before it is formatted. The following table lists the possible values for data partitions:

Data Partition Option	Action
BYTE	Specifies that the data displayed is to be partitioned into bytes.
LONGWORD	Specifies that the data displayed is to be partitioned into longword values. This is the default partitioning for DECIMAL, HEXADECIMAL, or OCTAL.
RECORDS (D)	Specifies that no further partitioning of data is to occur beyond the record partitioning already in the file. This partitioning is most useful when the files contain textual data. You can only specify RECORDS by itself or in conjunction with ASCII. It is mutually exclusive with all other options. This value is the default.
WORD	Specifies that the data displayed be partitioned into word values.

/FULL

Directs CMS to include the following information about the file used to create each generation:

- Creation time
- Revision time
- Revision number
- Record format
- Record attributes

CMS also indicates deleted lines in the source listing. Each set of one or more deleted lines is identified by a count of the deleted lines.

/GENERATION[=generation-expression]
/GENERATION=1+ (D)

Specifies a particular generation of the element to be annotated. If you omit /GENERATION, CMS annotates the most recent generation on the main line of descent.

The history contains a description of every generation of the element, including those created after the specified generation. (Generations created after the specified generation are not marked with an asterisk.)

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/MERGE=generation-expression
/NOMERGE (D)
Combines two generations of an element and creates a single file that contains the annotated listing. The parameter on the /MERGE qualifier specifies the generation that is merged into the retrieved generation. This command creates a file that contains the text common to both generations and the changes made to both generations. When changes that are not identical are made in the same position of the common ancestor, the changes from both generations are included in the resulting file and are marked as a conflict. By default, generations are not merged.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL —equivalent to (ELEMENT, GROUP, CLASS)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
CLASS (D)
NOCLASS
NONE —equivalent to (NOELEMENT, NOGROUP, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/OUTPUT[=file-specification]
/OUTPUT=element-name.ANN (D)
Directs CMS to write output to the specified file. CMS creates a new file if you do not specify /APPEND. If you omit the /OUTPUT qualifier (or if you specify /OUTPUT but do not provide a file specification), CMS creates a file with the element file name and the file type .ANN. Wildcards are allowed.

If you annotate more than one element (by specifying wildcards or a group name for the element expression parameter), and you do not specify wildcards in the output file specification, CMS creates successive versions of the file indicated by /OUTPUT.

Example

```
CMS> ANNOTATE TIMECVT.BAS/GENERATION=3
%CMS-S-ANNOTATED, element DISKX:[WORK.CMSLIB]TIMECVT.BAS annotated
```

This command produces a file named TIMECVT.ANN, which contains the annotated listing of generation 3 of TIMECVT.BAS. The contents of TIMECVT.ANN are as follows:

```
Annotated listing for element TIMECVT.BAS in CMS Library DISKX:[WORK.CMSLIB]
  25-APR-1990 15:50:29

 4    15-APR-1990 10:01:55 JAMES "additional error checks"
*3    12-APR-1990 15:49:01 JAMES "add check for invalid delta time"
*2    27-MAR-1990 12:39:58 JAMES "jp - fixed length string required"
*1    25-MAR-1990 15:37:11 JAMES "time conversion program"

Annotated listing for element TIMECVT.BAS in CMS Library DISKX:[WORK.CMSLIB]
  25-APR-1990 15:50:29
```

```
         1      10      rem Program to compute an absolute time given the present time
         2              rem and a delta time. The result is written to a file.
         3
         4      20      OPTION TYPE = EXPLICIT
         5              DECLARE STRING DELTA_TIME
2        6              MAP (STRING_LEN) STRING ASC_TIME = 80
         7              DECLARE LONG RETCODE
         8              DIM LONG BINARY_DELTA(1)
         9              DIM LONG NOW(1)
         10             DIM LONG BINARY_CVT_TIME(1)
         11
         12     100     EXTERNAL LONG CONSTANT SS$_NORMAL
3        13             EXTERNAL LONG CONSTANT SS$_IVTIME
         14             EXTERNAL LONG FUNCTION LIB$ADDX
         15             EXTERNAL LONG FUNCTION LIB$SUBX
         16             EXTERNAL LONG FUNCTION LIB$INT_OVER
         17             EXTERNAL INTEGER FUNCTION SYS$BINTIM (STRING BY DESC, LONG BY REF)
         18             EXTERNAL INTEGER FUNCTION SYS$GETTIM (LONG BY REF)
         19             EXTERNAL INTEGER FUNCTION SYS$ASCTIM (LONG BY REF,STRING BY DESC, &
         20                                                   LONG BY REF,LONG BY REF)
         21     150     LET RETCODE = LIB$INT_OVER(0)
         22             PRINT "Input delta time"
         23             INPUT DELTA_TIME
         24             LET RETCODE = SYS$BINTIM ( DELTA_TIME, BINARY_DELTA(0) )
3        25     175     IF (RETCODE = SS$_NORMAL) THEN GOTO 200
3        26             ELSE IF RETCODE = SS$_IVTIME THEN     &
3        27                  PRINT ,"INVALID TIME"
3        28                  GOTO DONE
3        29                  END IF
3        30             END IF
3        31     200     LET retcode = SYS$GETTIM(NOW(0))
3        32             IF (VAL( DELTA_TIME ) > 0 ) THEN      &
         33             retcode=LIB$ADDX(NOW(0),BINARY_DELTA (0) ,BINARY_CVT_TIME(0))
         34             END IF
         35             LET retcode = SYS$ASCTIM(,ASC_TIME,BINARY_CVT_TIME(0),)
         36             OPEN "TIME.TMP" FOR OUTPUT AS FILE #1
         37             PRINT #1,ASC_TIME
         38             CLOSE #1
         39     32767   Done:  END
         40
```

The element's history appears at the beginning of the file TIMECVT.ANN. The history lists the records of the transactions that created each of the four generations. However, because the third generation was annotated (ANNOTATE TIMECVT.BAS/GENERATION=3), changes made after generation 3 are not shown in the annotated listing. Generation 3 and its ancestors are marked with an asterisk in the history.

The source file listing shows each line of the file, including line numbers. The numbers farthest to the left are the generation numbers in which the line was most recently inserted or modified; the lines with no generation numbers have not changed since generation 1. The next column of numbers is assigned by CMS. The third column of numbers is included in the program itself.

CANCEL REVIEW

The CANCEL REVIEW command changes the review status of each specified element generation from pending to none and removes it from the review pending list.

Format

CANCEL REVIEW *element-expression "remark"*

Command Qualifiers	Defaults
/[NO]CONFIRM	/NOCONFIRM
/GENERATION[=generation-expression]	See text
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL

Command Parameters

element-expression
Specifies one or more elements. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The CANCEL REVIEW command changes the review status of each specified element generation from pending to none and removes it from the review pending list.

Use this command only on element generations that have reviews pending (see the description of the REVIEW GENERATION command for more information).

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/GENERATION[=generation-expression]
Specifies which generation of the element is to have its review pending status canceled. If you omit /GENERATION, CMS cancels the review of the most recently created generation with a review pending. You specify this qualifier only if more than one generation of an element is under review.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL —equivalent to (ELEMENT, GROUP, CLASS)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
CLASS (D)
NOCLASS
NONE —equivalent to (NOELEMENT, NOGROUP, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

Example

```
$  CMS CANCEL REVIEW EXAMPLE.SDML "review the final version only"
%CMS-S-CANCELED, review of generation 2 of element
DISKX:[PROJECT.CMSLIB]EXAMPLE.SDML canceled
```

This command cancels the review of the element EXAMPLE.SDML, and removes it from the review pending list.

CONVERT LIBRARY

Converts libraries that were created with Version 2.n of CMS for use with Version 3.n of CMS.

Format

CONVERT LIBRARY *V2-library-directory-specification*
V3-library-directory-specification

Command Parameters

V2-library-directory-specification
Specifies the directory specification of the existing CMS library you want to convert.

V3-library-directory-specification
Specifies the directory specification of the new CMS library you want to create. This directory must be empty.

Description

The CONVERT LIBRARY command creates a copy of an existing CMS library and converts the copy for use with this version of CMS. Libraries created with CMS Version 3.0 do not need to be converted.

To convert a library, you must first create an empty directory to contain the new, converted library. Conversion maintains all library setup in your existing library except for VMS ACL protection mechanisms assigned to the CMS library directory or to any CMS library files in that directory. The conversion process neither propagates original VMS directory or file ACLs, nor assigns them new default ACLs. Thus, you must assign the default protection on the converted library. For example, to assign read, write, and delete access to the system, owner, and group categories, you could use the following command:

```
$  SET FILE/ACL=(DEFAULT_PROTECTION,S:RWD,O:RWD,G:RWD) CMSLIB_V3.DIR
```

The conversion process also maintains the reference copy directory. After the library is converted, CMS automatically executes the VERIFY/REPAIR command to ensure that any existing reference copy directory is valid and current.

The CONVERT LIBRARY command causes an unusual occurrence to be logged in the history file.

Example

```
$  CREATE/DIRECTORY [PROJECT.CMSLIB_V3]
$  CMS
CMS> CONVERT LIBRARY
_V2 library:  [PROJECT.CMSLIB_V2]
_Directory for V3 library:  [PROJECT.CMSLIB_V3]
%CMS-S-CREATED, CMS Library DISKX:[PROJECT.CMSLIB_V3] created
%CMS-I-LIBINSLIS, Library DISKX:[PROJECT.CMSLIB_V3] inserted into the library list
%CMS-I-CONELE, element DATAPROG.BAS converted
%CMS-I-CONGRP, group TESTGRP converted
%CMS-I-CONCLS, class ETMETAL converted
%CMS-I-CONRES, all reservations converted
%CMS-I-CONHIS, history file converted
%CMS-S-CONVERTED, Version 2 library converted to Version 3 format
%CMS-I-VERCLS, class list verified
%CMS-I-VERCMD, command list verified
%CMS-I-VERELE, element list verified
%CMS-I-VERGRP, group list verified
%CMS-I-VERRES, reservation list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERARC, archive control block verified
%CMS-I-VER2, internal contiguous space verified
%CMS-I-VERCON, control file verified
%CMS-I-VEREDF, element DATAPROG.BAS verified
%CMS-I-VEREDFS, element data files verified
%CMS-I-REPAIRED, library [PROJECT.CMSLIB_V3] repaired
```

This example first creates a new directory to contain the converted library, and then converts the old library.

CMS automatically issues VERIFY/REPAIR to ensure that both the library and any reference copies are valid. If there are invalid reference copies, VERIFY/REPAIR repairs them. Even if you receive reference copy errors, your library is still converted and available for use.

COPY ELEMENT

Copies one or more existing elements to form one or more new elements. If you copy an element to the same library, the new element must have a different name. The COPY ELEMENT transaction preserves all element attributes, data, and history.

Format

COPY ELEMENT *old-element-expression new-element-name "remark"*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/NOCONFIRM
/LIBRARY[=directory-specification]	See text
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL

Command Parameters

old-element-expression
Specifies one or more existing elements to be copied. If you specify more than one element to be copied, you must use a wildcard character for the new element name. An old element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas.

new-element-name
Specifies the name for the new element. The name cannot be the same as any existing element name in the target library. The file-name component cannot be 00CMS because this name is reserved for CMS. Wildcards are allowed. If you specify more than one element with COPY ELEMENT, you must use a wildcard character for the new element name.

"remark"
Specifies a character string for the creation remark of the new element to be logged in the history file with this command. The remark is enclosed in quotation marks. If no remark was entered, then the remark from the old element is used for the creation remark of the new element, but a null remark ("") is logged in the history file.

Description

The COPY ELEMENT command uses an existing library element to copy and create a new element in the same library or in another library. The original element is left unchanged. The generation history, file characteristics, and element attributes are copied in full.

If the existing element has the **reference copy** attribute enabled (that is, if it was created or modified with /REFERENCE_COPY), the **reference copy** attribute is also enabled for the new element (assuming the **reference copy** attribute is established for the library).

If the existing element is reserved when you enter COPY ELEMENT, CMS informs you of the condition, then proceeds with the transaction. The new element is not reserved, regardless of whether the original element is reserved at the time of the copy transaction.

If a generation of the element is marked pending review, CMS informs you of the condition, then asks whether you want to proceed. If you type YES, CMS records the transaction as an unusual occurrence and proceeds with the command. The new element is not marked as pending review, regardless of whether the original element is marked at the time of the copy. If you type NO or press RETURN or CTRL/Z, no further action is taken.

CMS must be able to create one new element for each old element. When you use wildcards, a group name, or a comma list in the input element specification, CMS builds a list of elements to be copied. CMS uses this list as the point of reference during the copy transactions. If the output element specification does not allow CMS to create a new element for each element in the input list, the results may not be what you intend. For example, the following combination of wildcard expressions produces only one new element:

```
input element specification -  *.FOR
output element specification - NDATA.*
```

The first element that matches the input specification (*.FOR) produces one new element named NDATA.FOR. Each successive element that matches the input specification generates an error message because CMS can create only one unique element name from the given combination of wildcard expressions.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/LIBRARY[=directory-specification]
Identifies a valid CMS library that is the location of the element specified by the old-element-expression parameter. When you specify an alternative library, the new-element-name parameter is optional. If you do not specify a value for /LIBRARY, the current CMS library is used.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field

contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL —equivalent to (ELEMENT, GROUP)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
NONE —equivalent to (NOELEMENT, NOGROUP)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT and [NO]GROUP keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

Examples

1.
```
CMS> COPY ELEMENT INIT.FOR INITX.FOR "experimental version"
%CMS-S-COPIED, element DISKX:[PROJECT.CMSLIB]INIT.FOR copied to INITX.FOR
```

This command creates a new element named INITX.FOR in the same library as the old element.

2.
```
$ CREATE/DIRECTORY [RELEASE.CMSLIB]
$ CMS
CMS> CREATE LIBRARY [RELEASE.CMSLIB] "follows development library"
%CMS-S-CREATED, CMS Library DISKX:[RELEASE.CMSLIB] created

CMS> COPY ELEMENT *.*/LIBRARY=[PROJECT.CMSLIB] *.* "loading elements"
%CMS-I-COPIED, element DISKX:[PROJECT.CMSLIB]INIT.FOR copied to
DISKX:[RELEASE.CMSLIB]INIT.FOR
%CMS-I-COPIED, element DISKX:[PROJECT.CMSLIB]INITX.FOR copied to
DISKX:[RELEASE.CMSLIB]INITX.FOR
%CMS-I-COPIED, element DISKX:[PROJECT.CMSLIB]MSGDOC.FOR copied to
DISKX:[RELEASE.CMSLIB]MSGDOC.FOR
%CMS-I-COPIED, element DISKX:[PROJECT.CMSLIB]OUTPUT.FOR copied to
DISKX:[RELEASE.CMSLIB]OUTPUT.FOR
%CMS-I-COPIED, element DISKX:[PROJECT.CMSLIB]SEARCH.FOR copied to
DISKX:[RELEASE.CMSLIB]SEARCH.FOR
%CMS-I-COPIED, element DISKX:[PROJECT.CMSLIB]ARGCHK.FOR copied to
DISKX:[RELEASE.CMSLIB]ARGCHK.FOR
%CMS-S-COPIES, 6 copies completed
```

```
CMS>  SHOW HISTORY
History of DEC/CMS Library DISKX:[RELEASE.CMSLIB]
  9-MAY-1990 11:23:43 SMITH CREATE LIBRARY DISKX:[RELEASE.CMSLIB] "follows
        development library"
  9-MAY-1990 11:26:00 SMITH COPY ELEMENT/LIBRARY=DISKX:[PROJECT.CMSLIB] INIT.FOR
        INIT.FOR "loading elements"
  9-MAY-1990 11:26:04 SMITH COPY ELEMENT/LIBRARY=DISKX:[PROJECT.CMSLIB] INITX.FOR
        INITX.FOR "loading elements"
  9-MAY-1990 11:26:07 SMITH COPY ELEMENT/LIBRARY=DISKX:[PROJECT.CMSLIB] MSGDOC.FOR
        MSGDOC.FOR "loading elements"
  9-MAY-1990 11:26:15 SMITH COPY ELEMENT/LIBRARY=DISKX:[PROJECT.CMSLIB] OUTPUT.FOR
        OUTPUT.FOR "loading elements"
  9-MAY-1990 11:26:17 SMITH COPY ELEMENT/LIBRARY=DISKX:[PROJECT.CMSLIB] SEARCH.FOR
        SEARCH.FOR "loading elements"
  9-MAY-1990 11:26:19 SMITH COPY ELEMENT/LIBRARY=DISKX:[PROJECT.CMSLIB] ARGCHK.FOR
        ARGCHK.FOR "loading elements"
CMS>  SHOW GENERATION/DESCENDANTS INIT.FOR
Element generations in DEC/CMS Library DISKX:[RELEASE.CMSLIB]
INIT.FOR
  2       6-MAR-1990 17:34:04 SMITH "header offset and additional free space added"
  1       6-MAR-1990 17:26:10 SMITH "initialization routines"
```

This example creates a new directory for a new library, and then copies all of the elements from the library [PROJECT.CMSLIB] into the new library [RELEASE.CMSLIB]. Because the new elements are being created in a separate library, CMS can create new elements with the same names as the old elements; thus, a null string may be entered for the second parameter (for the new element name). In this case, CMS supplies the value *.*.

The SHOW HISTORY command that is executed after the copy transaction indicates that the library history contains only records of transactions performed on the new library (CREATE LIBRARY and COPY transactions). The SHOW GENERATION/DESCENDANTS command shows the generation history for one of the elements. The COPY ELEMENT transaction preserves the generation history for each element; thus, the record of replacement transactions (also the CREATE ELEMENT transaction that produced generation 1 of the element) is maintained from the old element to the new.

CREATE CLASS

Creates one or more empty classes.

Format

CREATE CLASS *class-name[,...] "remark"*

Command Qualifiers	**Defaults**
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL

Command Parameters

class-name
Specifies a name for the class. CMS reports an error if you specify a name that is already used for an existing class or group. (Class and group names must be unique.) If a previously used class or group name has been removed with the DELETE CLASS or DELETE GROUP command, you can reuse that name with CREATE CLASS. A class name can also be a list of class names separated by commas. Wildcards are not allowed.

"remark"
Specifies a character string for the creation remark of the class to be logged in the history file with this command. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The CREATE CLASS command establishes a class. After a class is created, you can place any related set of element generations in that class by using the INSERT GENERATION command. The CREATE CLASS command does not automatically place any generations in the created class. For more information on classes, see Chapter 5.

Command Qualifiers

/LOG (D)
/NOLOG

Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)

Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL —equivalent to OTHER
OTHER (D)
NOOTHER
NONE —equivalent to NOOTHER

You can specify either ALL, NONE, or the [NO]OTHER keyword.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object. If you specify /OCCLUDE=NOOTHER, CMS creates a class in every library in the search list.

Examples

1.
```
CMS>  CREATE CLASS INTERNAL_RELEASE "for internal use only"
%CMS-S-CREATED, class DISKX:[PROJECT.CMSLIB]INTERNAL_RELEASE created
```

This command creates a class named INTERNAL_RELEASE. Once the class name is established, element generations can be placed in the class with the INSERT GENERATION command.

2.
```
CMS>  CREATE CLASS FTEST1,FTEST2,V1 "for external release"
%CMS-I-CREATED, class DISKX:[PROJECT.CMSLIB]FTEST1 created
%CMS-I-CREATED, class DISKX:[PROJECT.CMSLIB]FTEST2 created
%CMS-I-CREATED, class DISKX:[PROJECT.CMSLIB]V1 created
%CMS-S-CREATES, 3 creations completed
```

This command creates the three classes FTEST1, FTEST2, and V1.

CREATE ELEMENT

Creates one or more new elements in a CMS library from an existing file.

Format

CREATE ELEMENT *element-expression "remark"*

Command Qualifiers	Defaults
/[NO]CONCURRENT	/CONCURRENT
/[NO]CONFIRM	/NOCONFIRM
/HISTORY="string"	/NOHISTORY
/NOHISTORY	
/INPUT[=file-specification]	See text
/[NO]KEEP	/NOKEEP
/[NO]LOG	/LOG
/NOTES="string"	/NONOTES
/NONOTES	
/POSITION=n	See text
/[NO]REFERENCE_COPY	See text
/[NO]RESERVE	/NORESERVE
/[NO]REVIEW	/NOREVIEW

Restrictions

- If you specify the /NOTES qualifier, you must also specify the /POSITION qualifier on the same command line.

Command Parameters

element-expression
Specifies one or more elements to be created. If you do not specify the /INPUT qualifier (or if you specify /INPUT without a value), the element name must correspond to an existing file in your current default directory. The name cannot be the same as any existing element name in the library.

Do not use the file name 00CMS because this name is reserved for library control files. Generation 1 of the new element is created. An element expression can also be a list of element names separated by commas, or a wildcard expression.

"remark"
Specifies a character string for the creation remark of the element to be logged in the history file with this command. The remark is stored with both the element and its first generation. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The CREATE ELEMENT command creates the first generation of a new element by moving the input file into a CMS library. By default, CMS searches for the file in your current default directory. You can direct CMS to use a file in a different directory by specifying the /INPUT qualifier. After the element is created, CMS deletes all versions of the file used to create the new element. If you specify either the /KEEP or /RESERVE qualifiers, CMS does not delete the file.

When you create an element, you can also define the **history**, **concurrent**, **notes**, **position**, **reference copy**, and **review** attributes for the element or establish a reservation.

CMS stores the creation date and time, the revision date and time, file attributes, and the file revision number of the file used to create generation 1 of the new element. When you fetch or reserve a generation of an element, CMS restores the times, attributes, and file revision number associated with the file used to create the element generation. You can also display this information by using the SHOW GENERATION/FULL command.

To change the creation remark associated with the element or generation 1 of the element, use the MODIFY ELEMENT or MODIFY GENERATION command, respectively.

Command Qualifiers

/CONCURRENT (D)
/NOCONCURRENT
Specifies whether this element can have multiple reservations. After you create the element, you grant or deny concurrent access by using the MODIFY ELEMENT command.

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/HISTORY="string"
/NOHISTORY (D)
Establishes the **history** attribute for the element. If an element has the **history** attribute, its history is included in the file when you retrieve it with the FETCH or RESERVE command.

The quoted string specifies the format of the history. The quoted string must contain the characters #H or #B (lowercase is allowed) and can contain other printing characters. To include a quotation mark in the output history string, type it twice (""). To include a number sign (#) in the output history string, type it twice (##). For a detailed explanation of the **history** attribute, see Section 4.5.

/INPUT[=file-specification]
Specifies the file to be used to create the element. When you specify an alternative location for the input file, CMS deletes the file from the alternative location (unless you specify /KEEP or /RESERVE). If you do not specify this qualifier, CMS searches your current default directory for a file with the same name as specified with the element expression parameter on the command line. Wildcards are allowed.

CMS must be able to create a unique element for each file in the input file list. Thus, if you use wildcards in the /INPUT file specification to specify more than one input file, you must also use wildcards in the element-name parameter.

/KEEP
/NOKEEP (D)
Controls whether CMS deletes all versions of the file used to create the new element. If you specify /KEEP, CMS does not delete the file.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/NOTES="string"
/NONOTES (D)
Establishes the **notes** attribute for the element. If an element has the **notes** attribute, notes are appended to the lines of the file when it is retrieved by the FETCH or RESERVE command.

The quoted string specifies the format of the note. The quoted string can contain text or the characters #G, #g, or both. If you specify /NOTES, you must also specify /POSITION. For a detailed explanation of the **notes** attribute, see Section 4.5.

/POSITION=n
Establishes the **position** attribute; that is, the character position where the note generated by the /NOTES qualifier begins on the line. The value n is required and must be an integer in the range 1 to 511. The /NOTES qualifier is required with the /POSITION qualifier.

The note is placed to the right of the text of the line. If the length of the line is less than n, the note appears at position n. If the length of the line is greater than or equal to n, the note is placed at the next tab stop after the end of the line. (Tab stops are at position 9 and every eight characters thereafter.) For a detailed explanation of the **position** attribute, see Section 4.5.

/[NO]REFERENCE_COPY
Controls whether CMS maintains a reference copy of the element. You must have established a reference copy directory.

The presence of the **reference copy** attribute for an element is inherited from the library, that is, if a reference copy directory is established for the library, the attribute is automatically enabled for the element. You can override the **reference copy** attribute by specifying /NOREFERENCE_COPY.

If a reference copy directory has been established for the CMS library, CMS creates a reference copy of the new element and updates the reference copy directory each time you create a new main-line generation of that element. When CMS places a file in the reference copy directory, it also deletes any earlier versions of that file in the reference copy directory.

/RESERVE
/NORESERVE (D)
Controls whether the new element is to be reserved after it is created. When you specify /RESERVE, CMS does not delete the file used to create the element. Generation 1 of the newly created element is automatically reserved.

If you omit both the /RESERVE and the /KEEP qualifiers, CMS deletes all versions of the file used to create the element.

/REVIEW
/NOREVIEW (D)
Specifies that new generations of the element are marked for review. By default, new generations of the element are marked for review only if the reserved generation either was rejected or has a review pending. If you specify CREATE ELEMENT/REVIEW, generation 1 of the element is also marked for review.

You can change the **review** attribute with the MODIFY ELEMENT command.

Example

```
CMS>  CREATE ELEMENT INIT.FOR "initialization routines"
%CMS-S-CREATED, element DISKX:[PROJECT.CMSLIB]INIT.FOR created
```

This command creates an element named INIT.FOR from a file with the same name in the current default directory, and then deletes all versions of that file in the current default directory.

CREATE GROUP

Creates one or more empty groups.

Format

CREATE GROUP *group-name[,...] "remark"*

Command Qualifiers	**Defaults**
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL

Command Parameters

group-name
Specifies a name for the group. CMS reports an error if you specify an existing group or class name. (Group and class names must be unique.) However, if a previously used group or class name has been removed with the DELETE GROUP or DELETE CLASS command, you can reuse that name with CREATE GROUP. A group name can also be a list of group names separated by commas. Wildcards are not allowed.

"remark"
Specifies a character string for the creation remark of the group to be logged in the history file with this command. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The CREATE GROUP command establishes a group. After a group is created, you can place any related set of elements or groups in that group by using the INSERT ELEMENT or INSERT GROUP command. The CREATE GROUP command does not automatically place any elements or groups in the created group. For more information about groups, see Chapter 5.

Command Qualifiers

/LOG (D)
/NOLOG

Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)

Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to OTHER
OTHER (D)
NOOTHER
NONE—equivalent to NOOTHER

You can specify either ALL, NONE, or the [NO]OTHER keyword.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object. If you specify /OCCLUDE=NOOTHER, CMS creates a group in every library in the search list.

Example

```
CMS> CREATE GROUP TIME_TST "files for time tests"
%CMS-S-CREATED, group DISKX:[PROJECT.CMSLIB]TIME_TST created
```

This command creates a group named TIME_TST. Once the group name is established, elements can be placed in the group with the INSERT ELEMENT command.

CREATE LIBRARY

Creates one or more new CMS libraries in one or more existing empty directories. You can have only one CMS library in each directory.

Format

CREATE LIBRARY *directory-specification[,...] "remark"*

Command Qualifiers	Defaults
/AFTER[=directory-specification]	See text
/BEFORE[=directory-specification]	See text
/[NO]LOG	/LOG
/REFERENCE_COPY=directory-specification	/NOREFERENCE_COPY
/NOREFERENCE_COPY	
/REVISION_TIME[=option]	See text

Restrictions

- You cannot specify both the /AFTER and the /BEFORE qualifiers on the same command line.

Command Parameters

directory-specification
Specifies one or more valid VMS directories. Each directory must not contain any files. A directory that is to be used as a CMS library cannot be your current default directory. If you specify more than one VMS directory, you must separate the directory specifications with commas. Wildcards are not allowed.

"remark"
Specifies a character string for the creation remark of the new library to be logged in the history file with this command. The remark is enclosed in

quotation marks. If no remark was entered, a null remark ("") is logged in the history file.

Description

The CREATE LIBRARY command builds CMS control files in a directory so that it can be used as a CMS library. After you establish a library with the CREATE LIBRARY command, you can enter CMS commands to manipulate the library. When you enter the CREATE LIBRARY command, your current CMS library is automatically set to the library (or libraries) specified. You can use CREATE LIBRARY only once on a library.

You can create more than one library at a time by specifying the CREATE LIBRARY command with more than one directory specification. The directory specifications must be separated by commas. For more information, see Chapter 3.

When you execute this command, CMS defines a logical name that begins with CMS$. These names are used by subsequent CMS commands. You should not define logical names beginning with CMS$ because this prefix is reserved for use by CMS.

Command Qualifiers

/AFTER[=directory-specification]
Instructs CMS to insert new libraries into the existing library search list immediately following the existing specified directory. The specified directory must be in the existing library search list. If you omit the directory specification, CMS automatically adds the libraries (in the order you specify) to the end of the list. You cannot specify both /AFTER and /BEFORE on the same command line. If neither /AFTER nor /BEFORE is specified, the CREATE LIBRARY command's library list supersedes any existing search list.

/BEFORE[=directory-specification]
Instructs CMS to insert new libraries into the existing library search list immediately in front of the existing specified directory. The specified directory must be in the existing library search list. If you omit the directory specification, CMS automatically adds the libraries (in the order you specify)

to the front of the list. You cannot specify both /AFTER and /BEFORE on the same command line. If neither /AFTER nor /BEFORE is specified, the CREATE LIBRARY command's library list supersedes any existing search list.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/REFERENCE_COPY=directory-specification
/NOREFERENCE_COPY (D)
Specifies a valid VMS directory to be used for reference copies of library elements. The directory cannot be a CMS library, nor should it be a subdirectory of a CMS library directory. Wildcards are not allowed.

If you use the CREATE LIBRARY command to create a search list of more than one library, you should specify a different reference copy directory for each library in the search list. Although CMS allows different libraries to be assigned the same reference copy directory, it is strongly recommended that you assign each CMS library its own unique reference copy directory. If you specify only one reference copy directory for more than one library, CMS creates one reference copy directory for the entire search list, *not* one reference copy directory for each library in the search list.

/REVISION_TIME[=option]
Controls whether CMS uses the original file revision time or the file storage time when a file is retrieved from the CMS library. The options field can contain one of the following keywords:

ORIGINAL (D)
STORAGE_TIME

Use the ORIGINAL keyword to indicate that the original revision time of files placed in a CMS library should be restored unchanged upon their retrieval. This is the default behavior.

Use the STORAGE_TIME keyword to indicate that the time when a file was stored in a CMS library (through a CREATE ELEMENT or REPLACE transaction) should be substituted for its original revision time upon retrieval.

Examples

1.
```
CMS> CREATE LIBRARY [RELEASE.CMSLIB] "follows development library"
%CMS-S-CREATED, CMS Library DISKX:[RELEASE.CMSLIB] created
```

   This command creates a CMS library in the subdirectory [RELEASE.CMSLIB]. The library does not contain any elements yet. Subsequent CMS commands refer to the library contained in [RELEASE.CMSLIB] until the user logs out or enters a SET LIBRARY or another CREATE LIBRARY command.

2.
```
CMS> CREATE LIBRARY [DOC.PRELIB],[DOC.TESTLIB],[DOC.FINLIB]
_Remark:  creating doc lib
```

   This example creates three CMS libraries in the subdirectories [DOC.PRELIB], [DOC.TESTLIB], and [DOC.FINLIB], and sets the library search list to the three libraries, in that order.

DELETE CLASS

Deletes one or more classes from a CMS library.

Format

DELETE CLASS *class-expression "remark"*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/NOCONFIRM
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL

Restrictions

- You cannot delete a class that contains any element generations.
- You cannot delete a class that has read-only access. (Use the MODIFY CLASS/NOREAD_ONLY command to change the access to the class.)

Command Parameters

class-expression
Specifies the class (or classes) to be deleted from the CMS library. A class expression can be a class name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

DELETE CLASS

Description

The DELETE CLASS command deletes one or more classes from a CMS library. The class must exist and must not contain any element generations. If any generations belong to the class, CMS issues an error message and does not delete the class. Use the REMOVE GENERATION command to remove element generations from a class before issuing the DELETE CLASS command.

Even though a class is deleted, records of transactions that created and used the class are retained in the library history. You can reuse the deleted class name to create a new class. However, there is no distinction between the two classes in the project history, except that their transactions are separated by entries for DELETE CLASS and CREATE CLASS commands.

To determine which generations belong to a class, use the SHOW CLASS command with the /CONTENTS qualifier.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to CLASS
CLASS (D)
NOCLASS
NONE—equivalent to NOCLASS

You can specify either ALL, NONE, or the [NO]CLASS keyword.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

Examples

1.
```
CMS> DELETE CLASS PRE_RELEASE "no longer necessary"
%CMS-S-DELETED, class DISKX:[PROJECT.CMSLIB]PRE_RELEASE deleted
```

This command deletes the class named PRE_RELEASE.

2.
```
CMS> REMOVE GENERATION *.* BETA*
_Remark:  beta sites converted to released product
%CMS-I-GENREMOVED, generation 3 of element DISKX:[PROJECT.CMSLIB]INI.FOR
removed from class DISKX:[PROJECT.CMSLIB]BETAFEB
%CMS-I-GENREMOVED, generation 4 of element DISKX:[PROJECT.CMSLIB]SRC.FOR
removed from class DISKX:[PROJECT.CMSLIB]BETAFEB
%CMS-I-GENREMOVED, generation 3 of element DISKX:[PROJECT.CMSLIB]INI.FOR
removed from class DISKX:[PROJECT.CMSLIB]BETAJAN
%CMS-I-GENREMOVED, generation 2 of element DISKX:[PROJECT.CMSLIB]SRC.FOR
removed from class DISKX:[PROJECT.CMSLIB]BETAJAN
%CMS-S-REMOVALS, 4 removals completed

CMS> DELETE CLASS BETA* "beta sites converted to released product"
%CMS-I-DELETED, class DISKX:[PROJECT.CMSLIB]BETAFEB deleted
%CMS-I-DELETED, class DISKX:[PROJECT.CMSLIB]BETAJAN deleted
%CMS-S-DELETIONS, 2 classes deleted
```

This example removes all element generations from all classes whose names begin with the string BETA, then deletes all of the empty classes. CMS does not prompt for confirmation during deletion unless you specify the /CONFIRM qualifier.

DELETE ELEMENT

Deletes one or more elements from a CMS library.

Format

DELETE ELEMENT *element-expression "remark"*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/CONFIRM
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL

Restrictions

- You cannot delete an element that belongs to a group or has a generation in a class.
- You cannot delete an element that has a generation reserved.
- You cannot restore a deleted element.
- You cannot delete an element that has a generation under review.

Command Parameters

element-expression
Specifies one or more elements that are to be deleted from the library. An element expression can be an element name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The DELETE ELEMENT command deletes an element from a CMS library. If the element is set to /REFERENCE_COPY and there is a current reference copy directory for the CMS library, CMS deletes the corresponding file from the reference copy directory.

There cannot be any existing reservations for the element. The element cannot be a member of a group, nor can one of its generations belong to a class, or be under review. If one of the element's generations is under review, use the CANCEL REVIEW command to remove it from the review list. If an element is reserved, you must unreserve or replace it before you can delete the element. If the element belongs to any groups or has generations in any classes, use the REMOVE ELEMENT or REMOVE GENERATION command to remove it.

Even though an element is deleted, records of transactions that created and used the element are retained in the library history. You can reuse the deleted element name to create a new element. However, there is no distinction between the two elements in the library history, except that their transactions are separated by entries for DELETE ELEMENT and CREATE ELEMENT commands.

Command Qualifiers

/CONFIRM (D)
/NOCONFIRM

Controls whether CMS prompts you for confirmation before each transaction.

When you run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to ELEMENT
ELEMENT (D)
NOELEMENT
NONE—equivalent to NOELEMENT

You can specify either ALL, NONE, or the [NO]ELEMENT keyword.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

Example

```
CMS> DELETE ELEMENT INITX.FOR "x-version no longer required"
Delete element INITX.FOR? [Y/N] (N): Y
%CMS-I-DELETED, element DISKX:[PROJECT.CMSLIB]INITX.FOR deleted
%CMS-S-DELETIONS, 1 deletion completed
```

This example uses INITX.FOR as an experimental module; when it is no longer needed, it can be deleted from the library.

DELETE GENERATION

Deletes one or more generations of an element.

Format

DELETE GENERATION *element-expression "remark"*

Command Qualifiers	**Defaults**
/AFTER[=generation-expression]	See text
/ARCHIVE[=file-specification]	/NOARCHIVE
/BEFORE[=generation-expression]	See text
/[NO]CONFIRM	/CONFIRM
/FROM[=generation-expression]	See text
/GENERATION[=generation-expression]	/GENERATION=1+
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/TO[=generation-expression]	See text

Restrictions

- You cannot delete generation 1 of an element.
- You cannot delete a generation that has variants off it.
- You cannot delete a generation that has a review pending.
- You cannot delete a generation that is reserved.
- You cannot delete a generation that is in a class.
- All generations in the specified range of generations to be deleted must be on the same line of descent.
- You cannot use /GENERATION in combination with /AFTER, /BEFORE, /FROM, or /TO.
- You cannot specify /AFTER and /FROM on the same command line.
- You cannot specify /BEFORE and /TO on the same command line.

Command Parameters

element-expression
Specifies one or more generations of an element. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The DELETE GENERATION command removes information about one or more generations of an element. Once a generation is deleted, it cannot be restored to its former place in the element in the CMS library. You can, however, archive the contents of the generation using the /ARCHIVE qualifier and later restore the contents of the generation (see Chapter 9).

If the generation or range of generations to be deleted has a direct descendant generation (that is, a descendant generation on the same line of descent), then the changes associated with those generations are combined, and then those changes are combined with the changes in the descendant generation. If there is no descendant generation, that is, the generation or range of generations to be deleted is at the end of the line of descent, then the changes associated with those generations are discarded. For more information, see Section 6.1.3.

You can specify a single generation with the /GENERATION qualifier. /GENERATION=1+ is the default. You can also specify a range of generations with either the /AFTER or /FROM qualifier to delimit the beginning of a range, and either the /BEFORE or /TO qualifier to delimit the end of a range. These sets of qualifiers can be paired to specify ranges with inclusive or exclusive endpoints (see the Restrictions section).

If you delete the latest generation on the main line of descent of an element that has the **reference copy** attribute, CMS deletes the generation's reference copy and creates a new reference copy that corresponds to the generation that is now the latest generation on the main line of descent.

Command Qualifiers

/AFTER[=generation-expression]
Specifies the start of a range of generations that are to be deleted, *excluding* the specified generation. You cannot specify both /AFTER and /FROM or both /AFTER and /GENERATION. You must specify the end of the range with either the /BEFORE or /TO qualifier.

/ARCHIVE[=file-specification]
/NOARCHIVE (D)
Specifies a file to which CMS writes archived generation information. If the file specification is omitted, CMS creates a file with the same name as each element, assigns a file type of .CMS_ARCHIVE, and places it in your default directory.

/BEFORE[=generation-expression]
Specifies the end of a range of generations that are to be deleted, *excluding* the specified generation. You cannot specify both /BEFORE and /TO or both /BEFORE and /GENERATION. You must specify the start of the range with either the /AFTER or /FROM qualifier.

/CONFIRM (D)
/NOCONFIRM
Controls whether CMS prompts you for confirmation before each transaction.

When you run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/FROM[=generation-expression]

Specifies the start of a range of generations that are to be deleted, *including* the specified generation. You cannot specify both /FROM and /AFTER or both /FROM and /GENERATION. You must specify the end of the range with either the /BEFORE or /TO qualifier.

/GENERATION[=generation-expression]
/GENERATION=1+ (D)

Specifies a particular generation of the element to be deleted. By default, the most recent generation on the main line of descent is deleted. You cannot combine /GENERATION with any of the following qualifiers: /FROM, /TO, /AFTER, and /BEFORE.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/LOG (D)
/NOLOG

Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)

Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, CLASS)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
CLASS (D)
NOCLASS
NONE—equivalent to (NOELEMENT, NOGROUP, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/TO[=generation-expression]

Specifies the end of a range of generations that are to be deleted, *including* the specified generation. You cannot specify both /TO and /BEFORE or both /TO and /GENERATION. You must specify the start of the range with either the /AFTER or /FROM qualifier.

Examples

1.
```
CMS> DELETE GENERATION/NOCONFIRM SAMPLE.PAS/GENERATION=5B1
_Remark:  Delete variant line
%CMS-S-GENDELETED, 1 generation of element DISKX:[PROJECT.CMSLIB]SAMPLE.PAS
deleted
```

This command deletes generation 5B1 of the element SAMPLE.PAS. The /NOCONFIRM qualifier directs CMS to suppress the prompt confirming the operation.

2.
```
CMS> DELETE GENERATION SAMPLE.*/AFTER=V1/BEFORE=V2
_Remark:  delete generations between released versions
Delete 5 generations after V1(1) before V2(7) of element SAMPLE.PAS?
[Y/N] (N):  Y
%CMS-S-GENDELETED, 5 generations of element DISKX:[PROJECT.CMSLIB]SAMPLE.PAS
deleted
```

This command deletes all generations of the element after the generation in class V1 and before the generation in class V2, excluding the two generations in classes V1 and V2.

3.
```
CMS> DELETE GENERATION SAMPLE.PAS/AFTER=1/BEFORE=V1
_Remark:  delete a range
%CMS-E-NOGENDELETED, no generations of DISKX:[PROJECT.CMSLIB]SAMPLE.PAS  deleted
-CMS-E-VARINRANGE, range has variants

CMS> DELETE GENERATION/ARCHIVE/FROM=2A1/TO=2A1+/NOCONFIRM SAMPLE.PAS
_Remark:  delete the variant range and archive the deleted generations
%CMS-S-GENDELETED, 3 generations of element DISKX:[PROJECT.CMSLIB]SAMPLE.PAS
deleted
```

The first command specifies that all generations be deleted between generation 1 and the generation in class V1. CMS could not delete the generations, however, because it found variants for the indicated generations.

The second command specifies a range of generations to be deleted from and including the variant generation 2A1 to and including the latest variant generation of the element SAMPLE.PAS. In this case, CMS deleted 3 generations of the element. The /ARCHIVE qualifier directs CMS to save the deleted generations in an archive file in your default directory.

To display the descendants of a generation and the classes containing the generations, use the SHOW GENERATION/DESCENDANTS/MEMBER command.

DELETE GROUP

Deletes one or more groups from a CMS library.

Format

DELETE GROUP *group-expression "remark"*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/NOCONFIRM
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL

Restrictions

- You cannot delete a group that contains any elements or groups.
- You cannot delete a group that belongs to another group.
- You cannot delete a group that has read-only access. (Use the MODIFY GROUP/NOREAD_ONLY command to change the access to the group.)

Command Parameters

group-expression
Specifies the group (or groups) to be deleted. A group expression can be one or more group names, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The DELETE GROUP command deletes a group from a CMS library. If the group is not empty, or if it belongs to another group, CMS returns an error and does not delete the group.

Even though a group is deleted, records of transactions that created and used the group are retained in the library history. You can reuse the deleted group name to create a new group. However, there is no distinction between the two groups in the library history, except that their transactions are separated by entries for DELETE GROUP and CREATE GROUP commands.

To determine which elements and groups belong to a group, use the SHOW GROUP command with the /CONTENTS qualifier.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to GROUP
GROUP (D)
NOGROUP
NONE—equivalent to NOGROUP

You can specify either ALL, NONE, or the [NO]GROUP keyword.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

Example

```
CMS> DELETE GROUP TIME_TST "superseded by comparison tests"
%CMS-S-DELETED, group DISKX:[PROJECT.CMSLIB]TIME_TST deleted
```

This command deletes the group named TIME_TST.

DELETE HISTORY

Deletes all or part of the library history.

Format

DELETE HISTORY *"remark"*

Command Qualifiers	**Defaults**
/BEFORE=date-time	/BEFORE=current-time
/[NO]CONFIRM	/CONFIRM
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/OUTPUT[=file-specification]	/OUTPUT=HISTORY.DMP

Command Parameter

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The DELETE HISTORY command deletes all or part of the library history. CMS writes the deleted history records to a file named HISTORY.DMP in your current default directory. CMS cannot access this file as a history file.

The DELETE HISTORY command does not delete the library creation history record.

Whenever you delete some of the library history, CMS records two transactions. As with other commands that modify the contents of the library, CMS records the deletion transaction. In addition, CMS logs a REMARK transaction at the point that corresponds to the /BEFORE value. If you do not specify the /BEFORE qualifier, the default is /BEFORE=current-time. The REMARK transaction record includes the following remark: "PREVIOUS HISTORY DELETED". Both the REMARK and the DELETE HISTORY transactions are unusual transactions. When you use the SHOW HISTORY command, CMS identifies unusual transactions by displaying an asterisk (*) in the first column of the transaction record.

Command Qualifiers

/BEFORE=date-time
/BEFORE=current-time (D)
Deletes all of the history information before a specified time. A single entry is made in the history file specifying that a section of the history data has been removed. This entry is made at the location in the history file where the lines were deleted.

The time value can be an absolute, delta, or combination time value, or one of the following keywords: TODAY, TOMORROW, or YESTERDAY.

If the time value is a future value, CMS uses the current time.

/CONFIRM (D)
/NOCONFIRM
Controls whether CMS prompts you for confirmation before each transaction.

When you run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to OTHER
OTHER (D)
NOOTHER
NONE—equivalent to NOOTHER

You can specify either ALL, NONE, or the [NO]OTHER keyword.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/OUTPUT[=file-specification]
/OUTPUT=HISTORY.DMP (D)
Directs CMS to write output to the specified file. If you omit the /OUTPUT qualifier (or if you specify /OUTPUT but do not provide a file specification), CMS creates a file named HISTORY.DMP in your current default directory.

Example

```
CMS> DELETE HISTORY/BEFORE=TODAY "old history in HISTORY.DMP"
Confirm DELETE HISTORY/BEFORE=10-MAY-1990 [Y/N] (N): y
%CMS-S-HISTDEL, 89 history records deleted
```

This example deletes all of the library history prior to the current day. The following code shows the first few records contained in the HISTORY.DMP file:

```
Deleted dump for CMS Library DISKX:[PROJECT.CMSLIB]

  6-MAR-1990 17:07:50 SNAKE  CREATE ELEMENT OUTPUT.FOR "ASCII format"
  6-MAR-1990 17:26:10 MARTIN CREATE ELEMENT INIT.FOR "init routines"
  8-MAR-1990 12:33:09 LISA   RESERVE INIT.FOR(1) "change header offset"
  9-MAR-1990 17:34:04 RONALD REPLACE INIT.FOR(2) "header offset and
        additional free space added"
    .
    .
    .
```

The CREATE LIBRARY transaction is not deleted from the library history.

DIFFERENCES

Compares two files, two generations of elements, or a file and a generation of an element. If the entities are different, it creates a file that contains the lines that differ between them. If they are the same, it issues a message to that effect and does not create a differences file (unless the /FULL qualifier is in effect).

Format

DIFFERENCES *file1 [file2]*

Command Qualifiers	**Defaults**
/[NO]APPEND	/NOAPPEND
/FORMAT=(data-format,data-partition, [no]generation-differences)	See text
/FULL	See text
/IGNORE=(keyword[,...])	See text
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/OUTPUT[=file-specification]	/OUTPUT=file1.DIF
/NOOUTPUT	
/[NO]PAGE_BREAK	/NOPAGE_BREAK
/[NO]PARALLEL	/NOPARALLEL
/SENTINEL=("begin-delimiter","end-delimiter")	See text
/SKIP=number-of-lines	See text
/WIDTH=n	See text

Parameter Qualifier	**Defaults**
/GENERATION[=generation-expression]	

Restrictions

- If both file1 and file2 are element generations, both generations must reside in the same library of the library search list or an error will occur.

Command Parameters

file1
Specifies the first file that is to be compared. If you specify /GENERATION for this parameter, file1 must be an element name; otherwise, a VMS file specification is assumed.

file2
Specifies the second file that is to be compared. If you specify /GENERATION for this parameter, file2 must be an element name; otherwise, a VMS file specification is assumed.

CMS follows these rules when you do not provide a second file specification:

- If you direct CMS to take file1 from a location that is not a CMS library, CMS uses the next lower file version in the same directory as file1.
- If you direct CMS to take file1 from a CMS library (by specifying /GENERATION), CMS uses the latest default directory version of file1 as the second input file.

Description

The DIFFERENCES command compares the contents of two files. If CMS finds differences, it creates a file named first-file-name.DIF in your current default directory (unless /OUTPUT is in effect.) If the files are the same, it issues a message to that effect and does not create a differences file. By default, CMS compares two files that are not located in a CMS library. However, you can direct CMS to use element generations from the current library by specifying the /GENERATION qualifier on one or both of the file name parameters.

A difference is defined as one of the following:

- A line (or lines) that are in one file and not in the other.
- A replacement of n lines by m lines (n may or may not be equal to m).

Only the lines that differ are displayed in the differences file (unless you specify /FULL).

A heading at the beginning of the differences file includes the name of the user that entered the command, the date and time the command was entered, and the file specifications of the two files being compared. If you direct CMS to use element generations and you have specified the /FORMAT option generation-differences, the differences listing contains a section labeled "Generation Differences" that contains the replacement history for the element. Each generation used in the comparison is identified by an asterisk (*) in the first column of the transaction record.

The differences between the files are contained in a section labeled "Text Differences." Each difference is formatted with the line (or lines) from the first file followed by the differing line (or lines) from the second file. If a difference consists of a line or lines in one file but not the other, only the lines from the file containing the additional text are displayed.

If you specify the /SKIP, /SENTINEL, and /IGNORE qualifiers on the same command line, they are processed in the following order:

1. /IGNORE=HISTORY
2. /IGNORE=NOTES
3. /SKIP
4. /SENTINEL
5. /IGNORE options other than HISTORY or NOTES

For example, if you specify /SKIP=5 and /SENTINEL=("sushi","bar"), DIFFERENCES disregards the first 5 lines in each of the compared files, and then searches the remainder of each file for the sentinel character strings "sushi" and "bar".

Command Qualifiers

/APPEND
/NOAPPEND (D)
Controls whether CMS appends the command output to an existing file, or creates a new file. If you specify /APPEND and the output file does not exist, CMS creates a new file. If you do not provide an output file specification (see the description for /OUTPUT), CMS searches your current default directory for a file with the file name specified in the file1 parameter and the file type .DIF.

/FORMAT=(data-format,data-partition, [no]generation-differences)

Controls whether the output file is formatted, specifies the type of formatting, and controls whether a list of generation differences is included in the DIFFERENCES output. You can specify the parameters in any order.

data-format

Specifies the type of format. The following table lists the possible values for data formats:

Data Format Option	Action
ASCII (D)	Specifies that data be presented as if each byte represents a value in the ASCII character set. This option is most useful when files contain textual data. If no data partition is specified, data is partitioned into records. This option is the default.
DECIMAL	Specifies that each value be displayed as a decimal numeral. If no data partition is specified, data is partitioned into longwords. You cannot specify both DECIMAL and RECORDS.
HEXADECIMAL	Specifies that each value be displayed as a hexadecimal numeral. If no data partition is specified, data is partitioned into longwords. You cannot specify both HEXADECIMAL and RECORDS.
OCTAL	Specifies that each value be displayed as an octal numeral. If no data partition is specified, data is partitioned into longwords. You cannot specify both OCTAL and RECORDS.

data-partition

Specifies the type of data partition. A data partition is the size that data in each record is to be broken into before it is formatted. The following table lists the possible values for data partitions:

Data Partition Option	Action
BYTE	Specifies that the data displayed be partitioned into bytes. Records are not partitioned further unless the data-format option indicates otherwise.
LONGWORD	Specifies that the data displayed be partitioned into longword values. This is the default partitioning for DECIMAL, HEXADECIMAL, or OCTAL.
RECORDS (D)	Specifies that no further partitioning of data is to occur beyond the record partitioning already in the file. This partitioning is most useful when the files contain textual data. You can only specify RECORDS by itself or in conjunction with ASCII. It is mutually exclusive with all other options. This value is the default.
WORD	Specifies that the data displayed be partitioned into word values. Data records are not partitioned further unless the data format indicates otherwise.

generation-differences
nogeneration-differences (D)
Specifies whether or not a list of generation differences is to be included in the DIFFERENCES output. This option is applicable only if two element generations are compared by the DIFFERENCES command. In any other case, this option is ignored. The following table lists the available keywords for the /GENERATION parameter qualifier:

Generation Differences Option	Action
GENERATION_DIFFERENCES	Specifies that a list of differences is to be included in the output.
NOGENERATION_DIFFERENCES (D)	Specifies that no list of differences is included in the output.

/FULL
Directs CMS to include a complete listing in the output file, including identical text and differences between file1 and file2.

/IGNORE=(keyword[,...])

Specifies one or more of the following keywords. Each keyword indicates a type of special character to be ignored during the comparison.

Keyword	Ignored Characters
CASE	Directs CMS to ignore any differences between the case of alphabetic characters (A through Z, a through z).
FORM_FEEDS	Directs CMS to remove formfeed characters as it compares records from the two files.
HISTORY	Directs CMS to ignore element generation history as it compares a file with a generation. At least one of the files must be an element generation with the **history** attribute enabled.
LEADING_BLANKS	Directs CMS to remove leading blanks and tabs as it compares records from the two files.
NOTES	Directs CMS to ignore notes as it compares a file with a generation. At least one of the files must be an element generation with the **notes** attribute enabled.
SPACING	Directs CMS to compress multiple blanks and tabs into a single space as it compares records from the two files.
TRAILING_BLANKS	Directs CMS to remove trailing blanks and tabs as it compares records from the two files.

If the HISTORY or NOTES keyword is specified, the history or notes text is not used for the comparison, and is also removed from the output generated by DIFFERENCES. For all other options, the output generated by DIFFERENCES contains the original records used for the comparison, instead of the modified form of the records designated by the /IGNORE qualifier.

/LOG (D)
/NOLOG

Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, CLASS)
ELEMENT (D)
NOELEMENT
CLASS (D)
NOCLASS
NONE—equivalent to (NOELEMENT, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/OUTPUT[=file-specification]
/OUTPUT=file1.DIF (D)
Directs CMS to write output to the specified file. CMS creates a new file if you do not specify /APPEND. If you omit the /OUTPUT qualifier (or if you specify /OUTPUT but do not provide a file specification), CMS creates a file with the file name specified in the file1 parameter and the file type .DIF. If you specify a file name but omit the file-type component, CMS writes the output to a file with the specified file name and a file type of .DIF.

/NOOUTPUT
Directs CMS to execute a comparison without creating an output file. This form of the comparison may be significantly faster because CMS stops the transaction when it encounters the first difference.

/PAGE_BREAK
/NOPAGE_BREAK (D)
Controls whether CMS allows page breaks in the output file. Page breaks are converted to the string "<PAGE>" in the output file. Use /PAGE_BREAK to request the inclusion of page breaks in the output file.

/PARALLEL
/NOPARALLEL (D)

Controls whether the differing lines from the two files are formatted side by side. If you specify /PARALLEL, the differences from the first file are displayed on the left and the differences from the second file are displayed on the right. The heading of the differences report displays the file specification of the first file on the left and the file specification of the second file on the right.

By default, the width of the listing is 132. Use the /WIDTH qualifier to control the width. Vertical lines separate the text on the left side of the report from the text on the right side. The text from each of the files is allotted equal space (half the width of the full report).

If a line from one of the files being compared is longer than half the width of the full report, the line is truncated on the right. A plus sign (+) is printed at the end of the line to indicate that the line has been truncated.

/SENTINEL=("begin-delimiter","end-delimiter")

Specifies a pair of strings used to delimit a section of text to be ignored during the comparison of both files. The delimiters can be up to 256 characters per line, and must be unique. Any text between and including the delimiters is treated as if it did not exist. If you do not enclose the sentinel strings in quotation marks, they are converted to uppercase before the comparison of the files. Sentinel strings may contain any characters, but if you include spaces or tabs, they must be enclosed in quotation marks.

Sentinel strings can appear anywhere in a file. If text delimited by a sentinel pair crosses record boundaries, the text after the delimited region appears in its own record in the output file; it is not appended to the contents of the record in which the begin delimiter was found.

/SKIP=number-of-lines

Indicates the number of lines at the beginning of each file (or generation) that are to be ignored during the comparison of both files. You must specify a nonnegative integer value indicating the number of lines to be ignored.

/WIDTH=n

Specifies the limit for the width of the differences report. The value n is required and must be an integer in the range 48 to 511. If n is less than 48, 48 is used. If n is more than 511, 511 is used. The default width is the same as the width of the output device.

The width of the report is rounded down to the nearest multiple of 8 minus 1. CMS rounds down so that if you have specified the /PARALLEL qualifier, CMS correctly interprets the horizontal tabs in the file on the right. For example, if you specify a value of 100 on the /WIDTH qualifier, the actual width is 95.

Parameter Qualifier

/GENERATION[=generation-expression]
Directs CMS to search for an element generation in the current CMS library. The /GENERATION qualifier must immediately follow the element name to which it applies. If you specify /GENERATION but do not provide a generation expression, CMS uses the latest generation on the main line of descent.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

Examples

In the following examples, the files FIB1.PAS and FIB2.PAS are used as input to the DIFFERENCES command. The contents of these files are as follows:

FIB1.PAS;1:

```
PROGRAM FIBONACCI(OUTPUT);
VAR FOLD1, FOLD2, FNEW, I: INTEGER;
BEGIN
   FOLD1 := 0; FOLD2 := 0; FNEW := 1; I := 1;
   REPEAT
      WRITELN(I, FNEW);
      FOLD1 := FOLD2; FOLD2 := FNEW; FNEW := FOLD1 + FOLD2;
      I := I + 1;
   UNTIL FNEW > (MAXINT DIV 2);
END .
```

FIB2.PAS;1:

```
PROGRAM FIBONACCI(OUTPUT);
VAR FOLD1, FOLD2, FNEW, I: INTEGER;
BEGIN
   WRITELN('  I              FNEW');
   FOLD1 := 0; FOLD2 := 0; FNEW := 1; I := 1;
   REPEAT
      WRITELN(I:3, FNEW:20);
      FOLD1 := FOLD2; FOLD2 := FNEW; FNEW := FOLD1 + FOLD2;
      I := I + 1;
   UNTIL FNEW > (MAXINT DIV 2);
END .
```

1.
```
CMS> DIFFERENCES FIB1.PAS FIB2.PAS
%CMS-I-DIFFERENT, files are different
```

This command writes the differences between FIB1.PAS and FIB2.PAS to a file called FIB1.DIF. The contents of FIB1.DIF are as follows:

```
DEC/CMS File Comparison Utility
Files Compared By SMITH On 30-APR-1990 15:30:37
   (1)  DISKX:[WORK.TESTS]FIB1.PAS;1
   (2)  DISKX:[WORK.TESTS]FIB2.PAS;1

+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
File DISKX:[WORK.TESTS]FIB2.PAS;1 Line 4
      2)    WRITELN('  I              FNEW');
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
File DISKX:[WORK.TESTS]FIB1.PAS;1 Line 6
      1)       WRITELN(I, FNEW);

File DISKX:[WORK.TESTS]FIB2.PAS;1 Line 6
      2)       WRITELN(I:3, FNEW:20);
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
**** End of Differences ****
```

The first difference consists of an additional line in the file FIB1.PAS. That line is displayed with a heading to indicate that it is line 4 of the file FIB1.PAS. The second difference shows that a line from FIB2.PAS differs from a line in FIB1.PAS. The differing lines from both files are included in the report.

2.
```
CMS> DIFFERENCES FIB1.PAS FIB2.PAS/PARALLEL/WIDTH=80/OUTPUT=FIB80
%CMS-I-DIFFERENT, files are different
```

This command specifies that the differences are to be written to the file FIB80.DIF (the default file type is used). The contents of FIB80.DIF are as follows:

```
CMS File Comparison Utility
Files Compared By SMITH On 30-APR-1990 15:31:50

   DISKX:[WORK.TESTS]FIB1.PAS;1         DISKX:[WORK.TESTS]FIB2.PAS;1


                    ----- Line     4 -+- Line     4 -----
                    ------------------|    WRITELN('  I              FNEW');
                                      +------------------

                    ----- Line     6 -+- Line     7 -----
      WRITELN(I, FNEW);               |      WRITELN(I:3, FNEW:20);
                    ------------------+------------------

**** End of Differences ****
```

The lines from FIB1.PAS are displayed on the left and the lines from FIB2.PAS are displayed on the right. The width of the report is 79 characters. This is the largest multiple of 8 minus 1 within the limit specified on the /WIDTH qualifier. The file specifications of the two files being compared are displayed as headings above the appropriate side of the report.

FETCH

Fetches a copy of a generation of one or more elements from a CMS library.

Format

FETCH *element-expression "remark"*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/NOCONFIRM
/GENERATION[=generation-expression]	/GENERATION=1+
/HISTORY="string"	See text
/NOHISTORY	
/[NO]LOG	/LOG
/MERGE=generation-expression	/NOMERGE
/NOMERGE	
/NOTES="string"	See text
/NONOTES	
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/OUTPUT[=file-specification]	/OUTPUT=element-name.type
/NOOUTPUT	
/POSITION=column-number	See text

Command Parameters

element-expression
Specifies one or more generations of an element to be retrieved from the library. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas. By default, CMS fetches the most recent generation on the main line of descent.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The FETCH command delivers a copy of the specified element generation to your current default directory. The element is not reserved, and CMS does not allow you to replace a fetched element. CMS allows you to fetch an element that is reserved, and notifies you of any current reservations for the element.

The presence or absence of a remark determines whether the FETCH transaction is recorded in the library history. If you enter a remark, CMS records the transaction in the history file. If you enter a null remark, CMS does not record the transaction in the history file.

If a version of a file with the same name as the element already exists in your current default directory when you execute the fetch transaction, CMS notifies you. A new version is then created with the next higher version number.

When you fetch a generation of an element from a CMS library, CMS restores the file creation time, revision time, revision number, and record format and attributes. The file that is placed in your directory has the same creation and revision times as the file that was used to create the generation that you are fetching.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/GENERATION[=generation-expression]
/GENERATION=1+ (D)
Specifies a particular generation of the element that is to be retrieved. If you omit /GENERATION, CMS fetches the most recent generation on the main line of descent.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/HISTORY="string"
Specifies that history is to be included in the retrieved file. The quoted string specifies the format of the history. The quoted string must contain the characters #H or #B (lowercase is allowed) and can contain other printing characters. To include a quotation mark in the output history string, type it twice (""). To include a number sign (#) in the output history string, type it twice (##). For a detailed explanation of the **history** attribute, see Section 4.5.

/NOHISTORY
Prevents CMS from appending the element history to the file. If you omit /NOHISTORY, and the retrieved element has the **history** attribute, the element history is included in the file when it is delivered to your current default directory. An element has the **history** attribute if you specified the /HISTORY qualifier on the CREATE ELEMENT or MODIFY ELEMENT command.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/MERGE=generation-expression
/NOMERGE (D)
Controls whether another generation of the element (called the merge generation) is to be merged with the generation that is being fetched (called the retrieved generation).

When you specify the /MERGE qualifier, CMS merges the lines of the two generations and delivers a single copy to your default directory. The file that is placed in your directory has the current creation and revision times. The merge generation cannot be on the same line of descent as the retrieved generation. When there is a conflict between blocks of one or more lines, CMS includes the conflicting lines and flags the conflict.

For a detailed explanation of how two generations are merged and how CMS treats conflicts between the generations, see Chapter 6.

/NOTES="string"
Temporarily establishes the **notes** attribute for the element, regardless of whether the element previously had the **notes** attribute enabled. If neither the /NOTES nor /NONOTES qualifier is specified for an element, but the element has the **notes** attribute enabled, notes are appended to the lines of the file when it is retrieved by the FETCH or RESERVE command.

The quoted string specifies the format of the note. The quoted string can contain text or the characters #G, #g, or both. If you specify /NOTES for an element that does not have the **notes** attribute enabled, then you must also specify /POSITION. For a detailed explanation of the **notes** attribute, see Section 4.5.

/NONOTES
Specifies that notes are not to be embedded in the output file. If you omit /NONOTES, and the retrieved element has the **notes** attribute, CMS embeds notes in the output file. An element has the **notes** attribute if you specified the /NOTES qualifier on the CREATE ELEMENT or MODIFY ELEMENT command.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, CLASS)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP

CLASS (D)
NOCLASS
NONE—equivalent to (NOELEMENT, NOGROUP, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/OUTPUT[=file-specification]
/OUTPUT=element-name.type (D)
Directs CMS to write output to the specified file. If you omit the /OUTPUT qualifier (or if you specify /OUTPUT but do not provide a file specification), CMS creates a file with the same name as the element.

If you fetch more than one element (by specifying wildcards or a group name for the element expression parameter), and you do not specify wildcards in the output file specification, CMS creates successive versions of the file indicated by /OUTPUT.

/NOOUTPUT
Specifies that the fetch operation is to be performed along with any history processing and error checking, but that no output file is to be created. By default, an output file with the same name as the element is created.

/POSITION=column-number
Specifies the column in which the note is to be placed. The column number is required and must be an integer in the range 1 to 511. The **notes** attribute or the /NOTES qualifier is required with the /POSITION qualifier.

If the length of the line is less than the specified column number, the note appears at the column number. If the length of the line is greater than or equal to the column number, the note is placed at the next tab stop after the end of the line. (Tab stops are at position 9 and every 8 characters thereafter.)

Example

```
CMS> FETCH INIT.FOR "check for correct spelling"
Element INIT.FOR currently reserved by:
    (1)    SMITH   2    30-APR-1990 15:48:35.65  "to add new routine"
%CMS-S-FETCHED, generation 2 of element DISKX:[PROJECT.CMSLIB]INIT.FOR fetched
```

This command fetches the latest generation on the main line of descent of element INIT.FOR. CMS indicates that the element is reserved, then continues with the fetch transaction.

HELP

Provides online CMS information.

Format

```
        ┌ topic                          ┐
        │ command                        │
HELP    │ command /qualifier             │
        │ command option                 │
        └ command option /qualifier      ┘
```

Command Parameters

topic
Specifies a subject that is related to CMS. For example, help on the topic OVERVIEW consists of general information on CMS and pointers to other topics that would be of interest to new users. Help on CLASSES defines the concept of a class and points to help on commands that manipulate classes.

command
Gives information about CMS either at DCL level or at CMS subsystem level. At DCL level, the DCL command HELP CMS provides online help on CMS commands, qualifiers, and other topics. For example:

```
$ HELP CMS
```

To get help on a specific CMS command, such as the CREATE ELEMENT command, type the command after HELP CMS. For example:

```
$ HELP CMS CREATE ELEMENT
```

You can get help at the CMS subsystem level by typing either HELP or HELP and the specific command. For example:

```
CMS> HELP CREATE ELEMENT
```

command /qualifier
Specifies a CMS command with an appropriate qualifier. For example:

```
$ HELP CMS DIFFERENCES/PARALLEL
```

This command gives you help at DCL level on the PARALLEL qualifier on the DIFFERENCES command.

command option
Specifies a CMS command with an appropriate option. For example:

```
$ HELP CMS SHOW ELEMENT
$ HELP CMS CREATE CLASS
```

These commands give you help at DCL level on the SHOW ELEMENT and CREATE CLASS commands.

command option/qualifier
Specifies a CMS command with an option and a qualifier. For example:

```
$ HELP CMS CREATE ELEMENT/RESERVE
```

This command gives you help at DCL level on the /RESERVE qualifier on the CREATE ELEMENT command.

Description

Online help for CMS is available at both the DCL and the CMS subsystem command level. At DCL level, you can type HELP CMS or CMS HELP to get information. At CMS subsystem level, you can type HELP to get information. If you omit a parameter after HELP CMS, you get an overview of the CMS HELP facility. In this overview, the general syntax of a CMS command is displayed, and all the keywords for which you can obtain more information are listed.

The command HELP CMS HELP gives a short explanation of how the HELP topics are organized.

INSERT ELEMENT

Inserts one or more elements or groups in the specified group (or groups).

Format

INSERT ELEMENT *element-expression group-expression "remark"*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/NOCONFIRM
/IF_ABSENT	See text
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL

Restrictions

- You cannot insert an element into a group that has read-only access.
- You cannot insert an element into a group if the element already belongs to the group.
- You cannot insert an element into a group in another library (the element and group must be in the same library).

Command Parameters

element-expression
Specifies one or more elements to be inserted into the group. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas.

group-expression
Specifies one or more groups into which the element (or elements) is to be inserted. A group expression can be a group name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The INSERT ELEMENT command places one or more elements or groups into one or more groups. The groups must already exist. When you use the INSERT ELEMENT command with a group name, you insert the contents of the group. For example, if you insert group A into group B by using the INSERT ELEMENT command, group B will contain the contents of group A at the time of the insertion transaction. If the contents of group A change at a later time, the contents of group B are not affected. To insert a group into another group, use the INSERT GROUP command.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/IF_ABSENT
Directs CMS to insert the element only if the group does not already contain that element. If the element already belongs to the group, CMS takes no action and does not return an error.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
NONE—equivalent to (NOELEMENT, NOGROUP)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT and [NO]GROUP keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

Examples

1.
```
CMS>  INSERT ELEMENT INIT.FOR TIME_TST "for time tests"
%CMS-S-INSERTED, element DISKX:[PROJECT.CMSLIB]INIT.FOR inserted into
DISKX:[PROJECT.CMSLIB]group TIME_TST
```

This command inserts the element INIT.FOR into the group named TIME_TST.

2.
```
CMS>  INSERT ELEMENT DBAS EXAMPLES "more examples for book"
%CMS-I-INSERTED, element DISKX:[PROJECT.CMSLIB]ARTFIG.CXS inserted into
DISKX:[PROJECT.CMSLIB]group EXAMPLES
%CMS-I-INSERTED, element DISKX:[PROJECT.CMSLIB]SNAKE.TXT inserted into
DISKX:[PROJECT.CMSLIB]group EXAMPLES
```

This command inserts the contents of group DBAS into group EXAMPLES.

INSERT GENERATION

Places one or more element generations in the specified class (or classes).

Format

INSERT GENERATION *element-expression class-expression "remark"*

Command Qualifiers	**Defaults**
/ALWAYS	See text
/[NO]CONFIRM	/NOCONFIRM
/GENERATION[=generation-expression]	/GENERATION=1+
/IF_ABSENT	See text
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/[NO]SUPERSEDE	/NOSUPERSEDE

Restrictions

- You cannot insert an element generation into a class that has read-only access.
- A class can contain only one generation of any particular element.
- You cannot insert a generation into a class in another library (the generation and class must be in the same library).

Command Parameters

element-expression
Specifies one or more elements whose generations are to be inserted into the class. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas. By default, the most recent generation on the main line of descent is inserted.

class-expression
Specifies an established class into which the element generation is being placed. The class must not have the **read-only** attribute. A class expression can be a class name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The INSERT GENERATION command places the specified element generation into one or more classes. The class (or classes) must already exist. (See the description of the CREATE CLASS command.)

A class can contain only one generation of an element. You cannot insert any generations into a class that has the **read-only** attribute. (See the description of the MODIFY CLASS command.)

Command Qualifiers

/ALWAYS
Directs CMS to insert the element generation into the class in all cases. If the class already contains a generation from the specified element, that generation is removed before the new one is inserted.

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/GENERATION[=generation-expression]
/GENERATION=1+ (D)
Specifies a particular generation of the element that is to be inserted into the class. If you omit /GENERATION, the INSERT GENERATION command uses the latest generation on the main line of descent.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/IF_ABSENT
Directs CMS to insert the element generation into the class only if a generation of that element is not already in the class. If a generation of the element is already in the class, CMS takes no action and does not return an error.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, CLASS)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
CLASS (D)
NOCLASS
NONE—equivalent to (NOELEMENT, NOGROUP, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/SUPERSEDE
/NOSUPERSEDE (D)
Controls whether CMS removes a generation of the element that exists in the class and replaces it with the specified generation. (Using /SUPERSEDE is the equivalent of using the REMOVE GENERATION command before the INSERT GENERATION command.)

If you specify /SUPERSEDE and there is no generation of the specified element already in the class, an error message is issued and the generation is not inserted into the class. You cannot use the /IF_ABSENT qualifier on the same command line as the /SUPERSEDE qualifier to override this action.

If you omit the /SUPERSEDE qualifier and a generation of the element already exists in the class, an error message is issued and no change is made to the library.

Examples

1.
```
CMS> INSERT GENERATION INIT.FOR PRE_RELEASE_V3 "internal version"
%CMS-S-GENINSERTED, generation 2 of element DISKX:[PROJECT.CMSLIB]INIT.FOR
inserted into class DISKX:[PROJECT.CMSLIB] PRE_RELEASE_V3
```

This command inserts the default generation of element INIT.FOR into the class PRE_RELEASE_V3.

2.
```
CMS> INSERT GENERATION INIT.FOR,SPEC.TXT/GENERATION=3/IF_ABSENT/CONFIRM
_Class name:  BASELEVEL_1 "inserting generation 3 for final baselevel
Insert generation 3 of element INIT.FOR into class BASELEVEL_1? [Y/N] (N):  Y
%CMS-I-GENINSERTED, generation 3 of element DISKX:[PROJECT.CMSLIB]INIT.FOR
inserted into class DISKX:[PROJECT.CMSLIB]BASELEVEL_1
Insert generation 3 of element SPEC.TXT into class BASELEVEL_1? [Y/N] (N):  Y
%CMS-I-GENINSERTED, generation 3 of element DISKX:[PROJECT.CMSLIB]SPEC.TXT
inserted into class DISKX:[PROJECT.CMSLIB]BASELEVEL_1
CMS-I-INSERTIONS, 2 insertions completed
```

This example inserts generation 3 of both elements INIT.FOR and SPEC.TXT into the class BASELEVEL_1. The /IF_ABSENT qualifier indicates that the generations should be inserted only if they are not already present in the class. The /CONFIRM qualifier directs CMS to prompt you for confirmation before each insertion.

INSERT GROUP

Places one or more groups into the specified group (or groups).

Format

INSERT GROUP *subgroup-expression group-expression "remark"*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/NOCONFIRM
/IF_ABSENT	See text
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL

Restrictions

- You cannot insert a group into another group that has read-only access.
- You can insert a group into another group only once.
- You cannot create recursive groups; that is, a group cannot directly or indirectly be a member of itself.
- You cannot insert a group from one library into a group in another library (both groups must be in the same library).

Command Parameters

subgroup-expression
Specifies one or more groups to be inserted into a second group (indicated by group-expression). A subgroup expression can be a group name, a wildcard expression, or a list of these separated by commas.

group-expression
Specifies the group into which subgroup-expression is to be inserted. A group expression can be a group name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The INSERT GROUP command inserts one or more groups into one or more other groups. Both groups must exist. When you use the INSERT GROUP command to insert group A into group B, the elements accessible through group B change as the contents of group A change. A group cannot be a member of itself; that is, it cannot be a subgroup of itself. For example, you cannot insert group A into group B if group A already contains group B.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/IF_ABSENT
Directs CMS to insert subgroup-expression into group-expression only if group-expression does not already contain it. If subgroup-expression already belongs to group-expression, CMS takes no action and does not return an error.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to GROUP
GROUP (D)
NOGROUP
NONE—equivalent to NOGROUP

You can specify either ALL, NONE, or the [NO]GROUP keyword.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

Example

```
CMS> INSERT GROUP USER_MANUAL CODE_AND_DOCS "user documentation"
%CMS-S-INSERTED, group DISKX:[PROJECT.CMSLIB]USER_MANUAL inserted into
group DISKX:[PROJECT.CMSLIB]CODE_AND_DOCS
```

This command inserts the group named USER_MANUAL into the group named CODE_AND_DOCS. As long as group USER_MANUAL belongs to group CODE_AND_DOCS, any changes to the contents of USER_MANUAL are reflected in the contents of CODE_AND_DOCS. Any element accessible through USER_MANUAL is also accessible through CODE_AND_DOCS.

MARK GENERATION

Marks each specified element generation for review and adds it to the review pending list.

Format

MARK GENERATION *element-expression "remark"*

Command Qualifiers	Defaults
/[NO]CONFIRM	/NOCONFIRM
/GENERATION[=generation-expression]	/GENERATION=1+
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL

Restrictions

- This command can be used only on element generations that do not already have reviews pending.

Command Parameters

element-expression
Specifies one or more elements or groups of elements whose generations are to be marked with pending review status. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The MARK GENERATION command changes the review status of the specified element generation from none to pending and inserts it into the review pending list. You can then review the element generation by using the REVIEW GENERATION command. Use one of the following commands to change the review status of the element generation: ACCEPT GENERATION, REJECT GENERATION, or CANCEL REVIEW. For more information, see Section 4.5.4.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/GENERATION[=generation-expression]
/GENERATION=1+ (D)
Specifies which generation of the element is to be marked as having review pending status. If you omit /GENERATION, CMS marks the most recent generation on the main line of descent.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)

Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, CLASS)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
CLASS (D)
NOCLASS
NONE—equivalent to (NOELEMENT, NOGROUP, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

Example

```
$  CMS MARK GENERATION/GENERATION=1X1 SPEC.COM
_Remark:  check this gen out before reinserting into class
%CMS-S-MARKED, generation 1X1 of element DISKX:[PROJECT.CMSLIB]EXAMPLE.SDML
marked for review
```

This command marks a specific generation for review. CMS adds the generation to the review pending list.

MODIFY CLASS

Changes the characteristics of a specified class (or classes).

Format

MODIFY CLASS *class-expression /qualifier "remark"*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/NOCONFIRM
/[NO]LOG	/LOG
/NAME=class-name	See text
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/[NO]READ_ONLY	See text
/REMARK="string"	See text

Restrictions

- You cannot modify a class that has read-only access. If a class has read-only access, you must change it to NOREAD_ONLY access to change the contents of the class or any other characteristics.
- You must specify one or more of the following qualifiers: /NAME, /[NO]READ_ONLY, or /REMARK.

Command Parameters

class-expression
Specifies the class being modified. A class expression can be a class name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null

remark ("") is logged. Note that this parameter and the string on the /REMARK qualifier are unrelated.

Description

The MODIFY CLASS command changes the characteristics of one or more classes. You can alter the following characteristics:

- The name of the class.
- The access to the class (READ_ONLY or NOREAD_ONLY). You cannot change the contents, the name, or the remark of a class that has been set to READ_ONLY.
- The creation remark that is associated with the class.

Use the SHOW CLASS command to display class characteristics.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/NAME=class-name

Specifies the new name for the class. The new class name cannot be the same as an existing class or group name. If a previously used class or group name has been removed with the DELETE CLASS or DELETE GROUP command, you can reuse that name. Wildcards and comma lists are not allowed.

If you specify the /NAME qualifier, you cannot use wildcards or a comma list in the class-expression parameter, nor can you use a wildcard for the /NAME qualifier. You cannot change the name of a class that has read-only access.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)

Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to CLASS
CLASS (D)
NOCLASS
NONE—equivalent to NOCLASS

You can specify either ALL, NONE, or the [NO]CLASS keyword.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/[NO]READ_ONLY

Establishes or alters the **read-only** attribute of a class. To change the characteristics of a read-only class, you must set the class to NOREAD_ONLY. NOREAD_ONLY is the default attribute of a class when it is created with the CREATE CLASS command.

/REMARK="string"

Specifies a new remark to be substituted for the creation remark that is associated with the class. You cannot change the remark of a class that has been set to READ_ONLY.

Examples

1.
```
CMS>  MODIFY CLASS PRE_RELEASE/READ_ONLY "freeze internal version"
%CMS-S-MODIFIED, class DISKX:[PROJECT.CMSLIB]PRE_RELEASE modified
```

   This command sets the class named PRE_RELEASE to READ_ONLY.

2.
```
CMS>  MODIFY CLASS PRE_RELEASE/NOREAD_ONLY/NAME=PRE_RELEASE_V3
_Remark:  include additional functions
%CMS-S-MODIFIED, class DISKX:[PROJECT.CMSLIB]PRE_RELEASE modified
```

   This example renames class PRE_RELEASE to PRE_RELEASE_V3. Because PRE_RELEASE had been set to READ_ONLY, it is necessary to use the /NOREAD_ONLY qualifier to modify the class.

MODIFY ELEMENT

Changes the characteristics of a specified element or elements.

Format

MODIFY ELEMENT *element-expression /qualifier "remark"*

Command Qualifiers	**Defaults**
/[NO]CONCURRENT	See text
/[NO]CONFIRM	/NOCONFIRM
/HISTORY="string"	See text
/NOHISTORY	
/[NO]LOG	/LOG
/NAME=element-name	See text
/NOTES="string"	See text
/NONOTES	
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/POSITION=n	See text
/[NO]REFERENCE_COPY	See text
/REMARK="string"	See text
/[NO]REVIEW	See text

Restrictions

- You cannot modify an element if it is set to read-only access.
- You can modify only the **reference copy**, **remark**, and **review** attributes of an element that has a generation reserved.
- If you specify /NOTES, you must also specify /POSITION on the same command line.
- You must specify only one or more of the following qualifiers:

  /[NO]CONCURRENT
  /[NO]HISTORY
  /NAME

```
/[NO]NOTES
/POSITION
/[NO]REFERENCE_COPY
/REMARK
/[NO]REVIEW
```

Command Parameters

element-expression
Specifies one or more elements to be modified. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged. Note that this parameter and the string on the /REMARK qualifier are unrelated.

Description

The MODIFY ELEMENT command changes the characteristics of one or more elements. You can alter the following characteristics:

- The **concurrent** attribute of the element
- The history string that is inserted in the element history when the element is reserved or fetched
- The element name
- The notes string and related **position** attribute
- The **reference copy** attribute of the element
- The creation remark that is associated with the element
- The **review** attribute

Use the SHOW ELEMENT command to display element characteristics.

If the **history**, **notes**, or **position** attribute is modified on an element that has the **reference copy** attribute, CMS creates an updated reference copy for the element.

Command Qualifiers

/[NO]CONCURRENT
Specifies whether this element can have multiple reservations. If you do not specify this qualifier, the existing concurrent access is not changed.

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/HISTORY="string"
Establishes the **history** attribute for the element. If an element has a **history** attribute, its history (which is similar to that produced by the ANNOTATE command) is included in the file when you retrieve it with the FETCH or RESERVE command.

The quoted string specifies the format of the history. The quoted string must contain the characters #H or #B (lowercase is allowed) and can contain other printing characters. To include a quotation mark in the output history string, type it twice (""). To include a number sign (#) in the output history string, type it twice (##). For a detailed explanation of the **history** attribute, see Section 4.5.

/NOHISTORY
Deletes any existing **history** attribute. If both /HISTORY and /NOHISTORY are omitted, any existing **history** attribute remains unchanged.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/NAME=element-name
Specifies the new name for the element. The new element name cannot be the same as an existing element name. Do not use the file name 00CMS. This name is reserved for CMS. If you specify the /NAME qualifier, you cannot use wildcards or a comma list in the element-name parameter, nor can you use a wildcard for the /NAME qualifier.

If an element is set to /REFERENCE_COPY, CMS creates a new reference copy of the element in the reference copy directory.

/NOTES="string"
Establishes the **notes** attribute for the element. If an element has a **notes** attribute, notes are appended to the lines of the file when it is retrieved by the FETCH or RESERVE command.

The quoted string specifies the format of the note. The quoted string can contain text or the characters #G, #g, or both. For a detailed explanation of the **notes** attribute, see Section 4.5.

/NONOTES
Cancels any current **notes** attribute and the corresponding **position** attribute.

If both /NOTES and /NONOTES are omitted, any existing **notes** attribute remains unchanged.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP)
ELEMENT (D)

NOELEMENT
GROUP (D)
NOGROUP
NONE—equivalent to (NOELEMENT, NOGROUP)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT and [NO]GROUP keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/POSITION=n

Establishes the **position** attribute; that is, the character position at which the note generated by the /NOTES qualifier is to begin on the line. A file that has a **position** attribute must have a **notes** attribute. Thus, the /POSITION qualifier can be used only if the element already has an established **notes** attribute, or the /NOTES qualifier is used on the same MODIFY ELEMENT command.

The value n is required and must be an integer in the range 1 to 511. If the length of the line is less than n, the note appears at position n. If the length of the line is greater than or equal to n, the note is placed at the next tab stop after the end of the line. (Tab stops are at position 9 and every 8 characters thereafter.)

If you omit /POSITION, any current **position** attribute remains unchanged.

/[NO]REFERENCE_COPY

Controls whether a new reference copy of an element is created or deleted. If you specify /REFERENCE_COPY, CMS creates a new reference copy of the element in the reference copy directory and updates the current reference copy directory whenever you create a new main-line generation of the element. If you specify /NOREFERENCE_COPY, CMS deletes the existing reference copy of the element. The reference copy directory must be established before you enter the MODIFY ELEMENT/REFERENCE_COPY command. Use the MODIFY LIBRARY/REFERENCE_COPY command to establish the reference copy directory.

/REMARK="string"

Specifies a new remark to be substituted for the creation remark that is associated with the element.

/[NO]REVIEW

Controls whether new generations of the element are marked for review. If you specify /REVIEW, new generations of the element are marked for review. If you specify /NOREVIEW, new generations are marked only if the reserved generation either is rejected or has a review pending. If you do not specify this qualifier, the existing **review** attribute is not changed.

To determine whether an element has the **review** attribute enabled, use the SHOW ELEMENT/FULL command.

Examples

1.
```
CMS> MODIFY ELEMENT INIT.FOR/NOCONCURRENT
_Remark: no more changes, other than those already discussed
%CMS-S-MODIFIED, element DISKX:[PROJECT.CMSLIB]INIT.FOR modified
```

This example sets the element INIT.FOR to NOCONCURRENT access. This means that only one person can reserve the element at a time.

2.
```
CMS> MODIFY ELEMENT/REVIEW EXAMPLE.SDML
_Remark: team should review all changes before performing build
%CMS-S-MODIFIED, element DISKX:[PROJECT.CMSLIB]EXAMPLE.SDML modified
```

This example marks the latest generation of the element EXAMPLE.SDML for review, and adds it to the review pending list. To display what generations are pending review, use the SHOW REVIEWS_PENDING command.

MODIFY GENERATION

Changes the characteristics of a specified generation (or generations).

Format

MODIFY GENERATION *element-expression /qualifier "remark"*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/NOCONFIRM
/GENERATION[=generation-expression]	/GENERATION=1+
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/REMARK="string"	See text

Restrictions

- You must specify one of the following qualifiers: /GENERATION or /REMARK.

Command Parameters

element-expression
Specifies one or more elements whose generations are to be modified. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged. Note that this parameter and the string on the /REMARK qualifier are unrelated.

Description

The MODIFY GENERATION command changes the remark that is associated with a particular generation of one or more elements. Use the SHOW GENERATION command to display generation characteristics.

This command does not change a generation's review status. See the ACCEPT GENERATION, CANCEL GENERATION, MARK GENERATION, REJECT GENERATION, and REVIEW GENERATION commands for more information.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/GENERATION[=generation-expression]
/GENERATION=1+ (D)
Specifies which generation of the element is to be modified. If you omit /GENERATION, CMS modifies the most recent generation on the main line of descent.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, CLASS)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
CLASS (D)
NOCLASS
NONE—equivalent to (NOELEMENT, NOGROUP, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/REMARK="string"
Specifies a new remark to be substituted for the creation (replacement) remark that is associated with the generation. This qualifier is required.

Example

```
CMS> MODIFY GENERATION/GENERATION=5/REMARK="Obsolete"
_Element expression:  SPEC.TXT
_Remark:  Marked obsolete
%CMS-S-MODIFIED, generation 5 modified
```

This command specifies a new remark to be substituted for the creation remark of generation 5 of element SPEC.TXT. You must also specify the element expression, and its associated remark, which is logged in the history file.

MODIFY GROUP

Changes the characteristics of the specified group (or groups).

Format

MODIFY GROUP *group-expression /qualifier "remark"*

Command Qualifiers	Defaults
/[NO]CONFIRM	/NOCONFIRM
/[NO]LOG	/LOG
/NAME=group-name	See text
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/[NO]READ_ONLY	See text
/REMARK="string"	See text

Restrictions

- You cannot change the attributes of a group that has been set to READ_ONLY. If a group has read-only access, you must change it to NOREAD_ONLY to change any other characteristics.
- You must specify one or more of the following qualifiers: /NAME, /[NO]READ_ONLY, or /REMARK.

Command Parameters

group-expression
Specifies the group to be modified. A group expression can be a group name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null

remark ("") is logged. Note that this parameter and the string on the /REMARK qualifier are unrelated.

Description

The MODIFY GROUP command changes the characteristics of one or more groups. You can alter the following characteristics:

- The name of the group.
- The access to the group (READ_ONLY or NOREAD_ONLY). You cannot change the contents of a group that has been set to READ_ONLY.
- The creation remark that is associated with the group.

Use the SHOW GROUP command to display group characteristics.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/NAME=group-name
Specifies the new name for the group. You cannot change the name of a group that has been set to READ_ONLY. The new group name cannot be the same as an existing group or class name. Wildcards and comma lists are not allowed. If you specify the /NAME qualifier, you cannot use wildcards or a comma list in the group name parameter, nor can you use a wildcard for the /NAME qualifier.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to GROUP
GROUP (D)
NOGROUP
NONE—equivalent to NOGROUP

You can specify either ALL, NONE, or the [NO]GROUP keyword.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/[NO]READ_ONLY
Establishes or alters the **read-only** attribute of a group. To change the characteristics of a READ_ONLY group, you must set the group to NOREAD_ONLY. NOREAD_ONLY is the default attribute of a group when it is created with the CREATE GROUP command.

/REMARK="string"
Specifies a new remark to be substituted for the creation remark that is associated with the group.

xample

```
CMS>  MODIFY GROUP TESTS/READ_ONLY "coordinate before changing contents"
%CMS-S-MODIFIED, group DISKX:[PROJECT.CMSLIB]TESTS modified
```

This command sets the group TESTS to READ_ONLY. Once the group is set to READ_ONLY, the contents cannot be changed.

MODIFY LIBRARY

Establishes or removes the connection between the current CMS library and a reference copy directory.

Format

MODIFY LIBRARY */qualifier "remark"*

Command Qualifiers	**Defaults**
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/REFERENCE_COPY=directory-specification	See text
/NOREFERENCE_COPY	
/REVISION_TIME[=option]	See text

Command Parameter

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The MODIFY LIBRARY command establishes or removes the connection between the current CMS library and a reference copy directory. The reference copy directory cannot be a CMS library. This command does not add files to or delete any files from a reference copy directory.
Once you establish a reference copy directory for a library, subsequent transactions that create new element generations on the main line of descent also update the reference copy directory (provided the element is set to /REFERENCE_COPY).

You must use the MODIFY ELEMENT/NOREFERENCE_COPY command on the elements in the library before you can use the MODIFY LIBRARY/NOREFERENCE_COPY command.

If you specify MODIFY LIBRARY/REFERENCE_COPY and the reference copy directory is already set, CMS first verifies all the files found in that directory. The contents of the directory must exactly correspond with the elements that have /REFERENCE_COPY set in the CMS library. If CMS finds discrepancies or if elements are set with /NOREFERENCE_COPY and there are existing reference copies for those elements, CMS advises you to use VERIFY/REPAIR.

Use the SHOW LIBRARY command to display library characteristics.

Command Qualifiers

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to OTHER
OTHER (D)
NOOTHER
NONE—equivalent to NOOTHER

You can specify either ALL, NONE, or the [NO]OTHER keyword.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/REFERENCE_COPY=directory-specification
/NOREFERENCE_COPY
Specifies a valid VMS directory to be used for reference copies of library elements. The directory cannot be a CMS library, nor can it be a subdirectory of a CMS library directory.

If you use the MODIFY LIBRARY command on a search list of more than one library, you should specify a reference copy directory for each library in the search list. Although CMS allows different libraries to be assigned the same reference copy directory, it is strongly recommended that you assign each CMS library its own unique reference copy directory. If you specify only one reference copy directory for more than one library, CMS uses one reference copy directory for the entire search list, not one reference copy directory for each library in the search list.

Use the /NOREFERENCE_COPY qualifier to remove the connection between the current CMS library and the current reference copy directory. Wildcards are not allowed.

/REVISION_TIME[=option]
Controls whether CMS uses the original file revision time or the file storage time when a file is retrieved from the CMS library. The options field can contain one of the following keywords:

ORIGINAL (D)
STORAGE_TIME

Use the ORIGINAL keyword to indicate that the original revision time of files placed in a CMS library should be restored unchanged upon their retrieval. This is the default behavior.

Use the STORAGE_TIME keyword to indicate that the time when a file was stored in a CMS library (through a CREATE ELEMENT or REPLACE transaction) should be substituted for its original revision time upon retrieval.

Example

```
CMS> MODIFY LIBRARY/REFERENCE_COPY=[WORK.REFCOPY] "current test area"
%CMS-S-MODIFIED, library [WORK.CODELIB] modified
```

This command establishes the reference copy directory [WORK.REFCOPY] for the current CMS library. In addition, to update the reference copy directory, an element must be set to /REFERENCE_COPY. If these two conditions are met, each transaction that creates a new main-line element generation also updates the reference copy directory.

REJECT GENERATION

Changes the review status of each specified element generation from pending to rejected and removes it from the review pending list.

Format

REJECT GENERATION *element-expression "remark"*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/NOCONFIRM
/GENERATION[=generation-expression]	See text
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL

Command Parameters

element-expression
Specifies one or more elements whose generations are to be rejected. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a character string to be associated with the element generation specified, to be logged in the history file with this command. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The REJECT GENERATION command changes the review status of each specified element generation from pending to rejected and removes it from the review pending list. You can use this command only on element generations that have reviews pending (see the description of the REVIEW GENERATION command for more information).

If you try to reserve a generation that has been rejected, CMS issues a message stating that the generation was rejected, and then prompts you for confirmation. Additionally, further generations created from a rejected generation are marked for review, regardless of the element's **review** attribute. See Section 4.5.4 for more information.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/GENERATION[=generation-expression]
Specifies a particular generation of the element to be rejected. If you omit /GENERATION, CMS rejects the most recently created generation with a review pending. You specify this qualifier only if more than one generation of an element is under review.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field

contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, CLASS)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
CLASS (D)
NOCLASS
NONE—equivalent to (NOELEMENT, NOGROUP, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

Example

```
$  CMS REJECT GENERATION EXAMPLE.SDML "don't change this until it is fixed"
%CMS-S-REJECTED, generation 1X1 of element DISKX:[PROJECT.CMSLIB]EXAMPLE.SDML
rejected
```

This command rejects the latest generation of EXAMPLE.SDML, which was on the review pending list. This generation of EXAMPLE.SDML remains rejected unless you specify a MARK GENERATION command.

REMARK

Places a remark in the library history.

Format

REMARK *"remark"*

Command Qualifiers	**Defaults**
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/UNUSUAL	See text

Description

The REMARK command adds a remark to the library history. When you let CMS prompt you for the remark, the length of the remark cannot exceed 254 characters. When you enter the remark on the command line, the length of the remark cannot exceed 256 characters. The remark is recorded in the library history in the following format:

```
date time username REMARK "remark"
```

For more information on remarks, see Section 10.2.2.

Command Qualifiers

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to OTHER
OTHER (D)
NOOTHER
NONE—equivalent to NOOTHER

You can specify either ALL, NONE, or the [NO]OTHER keyword.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/UNUSUAL
Specifies that the remark string placed in the history file be marked as an unusual occurrence, so that it appears marked with an asterisk in the output from SHOW HISTORY and is included in the output from a SHOW HISTORY/UNUSUAL command.

Example

```
CMS> REMARK "all transactions from this point use modules for new system"
%CMS-S-REMARK, remark added to history file
```

This command adds the remark enclosed in quotation marks to the library history.

REMOVE ELEMENT

Removes one or more elements from one or more groups.

Format

REMOVE ELEMENT *element-expression group-expression "remark"*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/NOCONFIRM
/IF_PRESENT	See text
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL

Restrictions

- You cannot remove elements from a group that has read-only access (see the description of the MODIFY GROUP command).

Command Parameters

element-expression
Specifies one or more elements to be removed from one or more groups. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas. When you use wildcard characters in the element expression, /IF_PRESENT is the default. (CMS does not return an error message if the group does not contain the element being removed.)

group-expression
Specifies the group from which one or more elements are to be removed. A group expression can be a group name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The REMOVE ELEMENT command removes one or more elements from one or more groups. The command does not delete the elements from the library, but there is no longer any association between the elements and the groups.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/IF_PRESENT
Directs CMS to remove the element from the group if it belongs to the group. If the element does not belong to the group, CMS takes no action and does not return an error. When you use wildcard characters in the element expression, /IF_PRESENT is the default.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
NONE—equivalent to (NOELEMENT, NOGROUP)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT and [NO]GROUP keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

Example

```
CMS> REMOVE ELEMENT *.* A2 "remove all elements from group"
%CMS-S-REMOVED, element DISKX:[PROJECT.CMSLIB]SEARCH.FOR removed from
group DISKX:[PROJECT.CMSLIB]A2
%CMS-S-REMOVED, element DISKX:[PROJECT.CMSLIB]ARGCHK.FOR removed from
group DISKX:[PROJECT.CMSLIB]A2
     .
     .
     .
```

This command removes all the elements from the group A2.

REMOVE GENERATION

Removes one or more element generations from one or more classes.

Format

REMOVE GENERATION *element-expression class-expression "remark"*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/NOCONFIRM
/GENERATION[=generation-expression]	See text
/IF_PRESENT	See text
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL

Restrictions

- You cannot remove a generation from a class with read-only access (see the description of the MODIFY CLASS command).

Command Parameters

element-expression
Specifies one or more generations of elements to be removed from one or more classes. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas. When you use wildcard characters in the element expression, /IF_PRESENT is the default. (CMS does not return an error message if the class does not contain a generation of the element.)

class-expression
Specifies the class from which the element generation is to be removed. The class must not have read-only access. A class expression can be a class name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The REMOVE GENERATION command removes an element generation from a class. The command does not delete the element generation from the library, but the element generation is no longer associated with the class.

To remove one element generation from a class and replace it with another generation of the same element, use the INSERT GENERATION command with the /SUPERSEDE qualifier.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/GENERATION[=generation-expression]
Directs CMS to remove a particular generation of an element from one or more classes in the library. The generation must be currently existing in the class. If you use a wildcard or a list of class names for the class expression, CMS deletes the particular generation from each specified class.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/IF_PRESENT
Directs CMS to remove any generation of the element that exists in the class. If the class does not contain a generation from the element, CMS takes no action and does not return an error. When you use wildcard characters in the element expression, /IF_PRESENT is the default.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, CLASS)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
CLASS (D)
NOCLASS
NONE—equivalent to (NOELEMENT, NOGROUP, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

Example

```
CMS> REMOVE GENERATION USER.DOC PRE_RELEASE_V3
_Remark:  internal documentation is online
%CMS-S-GENREMOVED, generation 2 of element DISKX:[PROJECT.CMSLIB]USER.DOC
removed from class DISKX:[PROJECT.CMSLIB]PRE_RELEASE_V3
```

This example removes generation 2 of USER.DOC from class PRE_RELEASE_V3.

REMOVE GROUP

Removes one or more groups from another group (or groups).

Format

REMOVE GROUP *group-expression1 group-expression2 "remark"*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/NOCONFIRM
/IF_PRESENT	See text
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL

Restrictions

- You cannot remove a group from a group with read-only access (see the description of the MODIFY GROUP command).

Command Parameters

group-expression1
Specifies one or more groups to be removed. Wildcards and a comma list are allowed. When you use wildcard characters or a comma list in the group name, /IF_PRESENT is the default. (CMS does not return an error message if group-expression2 does not contain group-expression1.)

group-expression2
Specifies one or more groups from which the groups in group-expression1 are to be removed. Wildcards and a comma list are allowed.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The REMOVE GROUP command removes a group from another group. The command does not delete the group from the library, but there is no longer any association between the two groups. Removing group A from group B means that the contents of group A are no longer accessible through group B.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/IF_PRESENT
Directs CMS to remove group-expression1 only if it belongs to group-expression2. If group-expression1 does not belong to group-expression2, CMS takes no action and does not return an error. When you use wildcard characters or a comma list in the group name, /IF_PRESENT is the default.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays

a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to GROUP
GROUP (D)
NOGROUP
NONE—equivalent to NOGROUP

You can specify either ALL, NONE, or the [NO]GROUP keyword.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

Example

```
CMS>  REMOVE GROUP A1 A2 "remove group from group"
%CMS-S-REMOVED, group DISKX:[PROJECT.CMSLIB]A1 removed from group
DISKX:[PROJECT.CMSLIB]A2
```

This command removes group A1 from group A2.

REPLACE

Returns each specified element reservation to the library and creates a new generation of the element.

Format

REPLACE *element-expression "remark"*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/NOCONFIRM
/GENERATION=generation-expression	See text
/IDENTIFICATION_NUMBER=n	See text
/IF_CHANGED	See text
/INPUT[=file-specification]	See text
/[NO]KEEP	/NOKEEP
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/[NO]RESERVE	/NORESERVE
/VARIANT=variant-letter	/NOVARIANT
/NOVARIANT	

Command Parameters

element-expression
Specifies one or more reserved generations of an element to be replaced. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas. If you specify more than one element (with either a group name or a wildcard expression), each file indicated by the element expression must exist in the same directory. When you use wildcards, CMS creates an input element list based on the list of element generations that you have reserved.

"remark"
Specifies a character string to be associated with the newly created generations, to be logged in the history file with this command. The remark is enclosed in quotation marks. If no remark was entered, then the remark from the corresponding reservation is used for the new generation and the replacement transaction in the history file.

Description

The REPLACE command transfers a file from your default directory to the current CMS library, thus creating a new generation. You can direct CMS to use a file other than the one located in your default directory by specifying the /INPUT qualifier. After the reserved generation is replaced, CMS deletes the file used to create the new generation (and any earlier versions of the file in the same directory). If you specify either the /KEEP or the /RESERVE qualifier, CMS does not delete the file. You cannot replace a reserved generation held by another user unless you hold BYPASS process privilege or unless you are granted BYPASS access to the element by an access control entry (see Section 7.1.2.2). After the replace transaction is completed, the reservation is ended.

The number of the new generation is the number of its predecessor with the rightmost level number increased by 1. For example, if you reserved generation 1A1, CMS would create generation 1A2 when you replaced it. CMS also stores the creation date and time, the revision date and time, and the file revision number of the file used to create the new generation. When you fetch or reserve a generation of an element, CMS restores the times and file revision number associated with the file used to create the element generation. You can also display this information by using the SHOW GENERATION/FULL command.

CMS reports an error if you try to create a generation that is already in the library (see the description of the /VARIANT qualifier).

The REPLACE command checks for other current reservations and concurrent replacements of the element, and whether you are replacing another user's reservation. If any of these situations occur, CMS prompts whether you want to proceed with the command. If you type NO or press RETURN or CTRL/Z, the command is not executed. If you type YES, CMS executes the command and records the transaction as an unusual occurrence.

If you have more than one reservation of an element, or if you are replacing a concurrent reservation made by another user (that is, if there is any ambiguity), you must specify the exact reservation to be replaced (see Section 4.3.3). You do this by using either the /GENERATION qualifier or the /IDENTIFICATION_NUMBER qualifier.

You can use /GENERATION as long as the concurrent reservations are not on the same generation. If you have more than one concurrent reservation for the same generation, you must identify the specific reservation to be replaced. Each reservation is assigned an identification number. Use the SHOW RESERVATIONS command to determine the identification number of each reservation. The identification number appears in parentheses at the beginning of each line. If you use the /IDENTIFICATION_NUMBER qualifier, you do not need to also use the /GENERATION qualifier; when both are used, CMS ignores the /GENERATION qualifier.

If the **reference copy** attribute is enabled for an element and REPLACE creates the new generation on the main line of descent, CMS creates a new reference copy in the reference copy directory for the element and deletes the old copy from the reference copy directory.

Replacing an Element with Defined Attributes

If you reserve a generation of an element with an embedded history and then replace it, the REPLACE command ignores the history; that is, CMS does not copy the history into your CMS library. If you add text to the file in or above the history (relative to #B), or in or below the history (relative to #H), the REPLACE command issues an error message and the command is not executed.

If you reserve a file with embedded notes and then replace it, the REPLACE command does not copy the notes to the CMS library. If, while editing the file, you insert text that looks like an embedded note, it is deleted when the file is replaced.

For more information about concurrent reservations and replacements, see Chapter 6. For detailed information on embedded histories and notes, see Section 4.5.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

The /NOCONFIRM qualifier does not override the confirmation prompt issued when you make a concurrent replacement or when you replace another user's reservation.

/GENERATION=generation-expression
Specifies which reserved generation of the element is to be replaced. If you have more than one reservation of the same element generation, you must use the /IDENTIFICATION_NUMBER qualifier to replace the reservation.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/IDENTIFICATION_NUMBER=n
Specifies which reservation is to be replaced. This qualifier is required when you have multiple reservations of the same generation of an element. /IDENTIFICATION_NUMBER can be used instead of /GENERATION when you have multiple reservations. Use the SHOW RESERVATIONS command to determine the identification number of each reservation. The identification number appears in parentheses at the beginning of each line.

/IF_CHANGED
Specifies that a new generation is to be created only if the input file is different from the generation that was reserved. CMS automatically creates a new generation, regardless of the existence of any differences.

CMS deletes the input file from the specified location after the new generation is created (unless you specify the /KEEP or /RESERVE qualifier).

/INPUT[=file-specification]
Specifies a file to be used as input for the replacement transaction. If you use the /INPUT qualifier but you do not supply a file specification, CMS searches your current default directory for a file with the same name as the element specified on the command line. When you specify /INPUT, CMS deletes the input file from the specified location after the new generation is created (unless you specify the /KEEP or /RESERVE qualifier).

CMS must be able to match the input element list with the list of elements indicated by the element expression parameter. Thus, if you use wildcards in the /INPUT file specification to generate more than one input file, you must also use wildcards in the element expression parameter.

/KEEP
/NOKEEP (D)
Controls whether the file used to create the new element generation is deleted from your directory. If you omit both /KEEP and /RESERVE, the files are deleted.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, CLASS)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
CLASS (D)
NOCLASS

NONE—equivalent to (NOELEMENT, NOGROUP, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/RESERVE
/NORESERVE (D)
Controls whether the new generation of the element created by the replacement is reserved. If you specify the /RESERVE qualifier, the generation is reserved and the element files are not deleted from your current default directory. The list of concurrent replacements is updated as if /RESERVE had been omitted. For information on concurrent reservations and replacements, see Chapter 6.

/VARIANT=variant-letter
/NOVARIANT (D)
Controls whether a variant generation is created. If you specify the /VARIANT=variant-letter qualifier, the number of the created generation is the predecessor's number, followed by the variant letter, followed by the number 1.

If two or more users have concurrently reserved the same element generation, the replaced generations cannot be on the same line of descent. Thus, one can be replaced as a main-line generation and the rest must be replaced as variants. For more information on creating variant generations, see Chapter 6.

Example

```
CMS>  REPLACE FILEIO.BLI
_Remark:  descriptor bug fixed
%CMS-S-GENCREATED, generation 14 of element DISKX:[PROJECT.CMSLIB]FILEIO.BLI
created
```

This command creates a new generation on the main line of descent of element FILEIO.BLI.

RESERVE

Retrieves a copy of each specified element generation from a CMS library and marks it as reserved.

Format

RESERVE *element-expression "remark"*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/NOCONFIRM
/GENERATION[=generation-expression]	/GENERATION=1+
/HISTORY="string"	See text
/NOHISTORY	
/[NO]LOG	/LOG
/MERGE=generation-expression	/NOMERGE
/NOMERGE	
/NOCONCURRENT	See text
/NOTES="string"	See text
/NONOTES	
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/OUTPUT[=file-specification]	See text
/NOOUTPUT	
/POSITION=column-number	See text

Command Parameters

element-expression
Specifies the element (or elements) from which a generation is to be reserved. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas. By default, CMS reserves the most recent generation on the main line of descent of each element designated by the element-expression.

"remark"
Specifies a character string to be associated with the reservation and logged in the history file with this command. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The RESERVE command sends a copy of the specified generation of each specified element to your current default directory (or to another location if you specified the /OUTPUT qualifier). Each element is marked as reserved. Usually, after you modify the file, you return your changes to the library with the REPLACE command. Alternatively, you can cancel the reservation with the UNRESERVE command.

You can reserve more than one generation of the same element if concurrent reservations are allowed (see Section 4.3.2). If a generation of an element is reserved by you or another user, or a generation of the element is under review or has been rejected, CMS displays the current reservations and review comments, and prompts whether you want to proceed with the command. If you type YES, you are added to the list of current reservers. The transaction is recorded as an unusual occurrence (see Chapter 9). If you type NO or press RETURN or CTRL/Z, no action is taken.

If a version of the output file exists in your default directory when you enter the RESERVE command, CMS notifies you. A new version is then created with the next higher version number.

When you retrieve a generation of an element from a CMS library, CMS restores the file creation and revision times. The file that is placed in your directory has the same creation and revision times as the file that was used to create the generation that you are reserving. If you specify /MERGE, the file placed in your default directory has the current creation and revision times.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

You cannot use /NOCONFIRM to override the prompt generated when you try to reserve a generation of an element that is already reserved by you or by another user.

/GENERATION[=generation-expression]
/GENERATION=1+ (D)
Specifies a particular generation of the element that is to be reserved. If you omit /GENERATION, CMS reserves the most recent generation on the main line of descent.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/HISTORY="string"
Specifies that history is to be included in the retrieved file. The quoted string specifies the format of the history. The quoted string must contain the characters #H or #B (lowercase is allowed) and can contain other printing characters. To include a quotation mark in the output history string, type it twice (""). To include a number sign (#) in the output history string, type it twice (##). For a detailed explanation of the **history** attribute, see Section 4.5.

/NOHISTORY
Prevents CMS from including the element history in the file. If you omit /NOHISTORY, and the retrieved element has the **history** attribute, CMS includes the element history in the output file. An element has the **history** attribute if the /HISTORY qualifier was specified on the CREATE ELEMENT or MODIFY ELEMENT command.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational

messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/MERGE=generation-expression
/NOMERGE (D)
Controls whether another generation of the element (called the merge generation) is to be merged with the generation that is being reserved (called the retrieved generation).

If you specify the /MERGE qualifier, CMS merges the lines of the two generations and delivers a single copy of the file to your default directory. The file that is placed in your directory has the current creation and revision times. The merge generation cannot be on the same line of descent as the retrieved generation. When there is a conflict between blocks of one or more lines, CMS includes the conflicting lines and flags the conflict.

For an explanation of how two generations are merged and how CMS treats conflicts between the generations, see Chapter 6.

/NOCONCURRENT
Specifies that the element cannot be reserved by another user while you have it reserved. You must replace or unreserve the element before others can reserve it. CMS allows concurrent reservations if the element has the **concurrent** attribute set (see Section 4.3.1).

/NOTES="string"
Specifies that notes are to be appended to the lines of the file as it is retrieved by the RESERVE operation. This qualifier overrides the element's **nonotes** attribute, if one was established.

The quoted string specifies the format of the note. The quoted string can contain text or the characters #G, #g, or both. If you specify /NOTES for an element that does not have the **notes** attribute enabled, then you must also specify /POSITION. For a detailed explanation of the **notes** attribute, see Section 4.5.

/NONOTES
Specifies that notes are not to be embedded in the output file. If you omit /NONOTES, and the retrieved element has the **notes** attribute, CMS embeds notes in the output file. An element has the **notes** attribute if the /NOTES qualifier was specified on the CREATE ELEMENT or MODIFY ELEMENT command.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)

Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, CLASS)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
CLASS (D)
NOCLASS
NONE—equivalent to (NOELEMENT, NOGROUP, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/OUTPUT[=file-specification]

Directs CMS to write output to the specified file. If you omit the /OUTPUT qualifier (or if you specify /OUTPUT but do not provide a file specification), CMS creates a file with the same name as the element in your default directory.

If you reserve more than one element (by specifying wildcards or a group name for the element expression parameter), and you do not specify wildcards in the output file specification, CMS creates successive versions of the file indicated by /OUTPUT.

/NOOUTPUT

Specifies that the generation is to be reserved, but that no output file is to be created.

/POSITION=column-number

Specifies the column in which the note is to be placed. The column number is required and must be an integer in the range 1 to 511. The **notes** attribute or the /NOTES qualifier is required with the /POSITION qualifier.

If the length of the line is less than the specified column number, the note appears at the column number. If the length of the line is greater than or equal to the column number, the note is placed at the next tab stop after the end of the line. (Tab stops are at positions 9 and every 8 characters thereafter.)

Examples

1.
```
CMS>  RESERVE FILEIO.BLI
_Remark:  fix temporary descriptor bug
%CMS-S-RESERVED, generation 13 of element DISKX:[PROJECT.CMSLIB]FILEIO.BLI
reserved
```

This command reserves generation 13 of the element FILEIO.BLI. When the element is replaced, a successor generation is created.

2.
```
CMS>  RESERVE SYNTAX.PAS
_Remark:  add syntax for RECORD declaration
Element SYNTAX.PAS currently reserved by:
  (1)  JERRYH     1     24-JUL-1990 16:17:45 "implement FOR loop syntax"
Proceed?  [Y/N] (N):  YES
%CMS-S-RESERVED, generation 1 of element DISKX:[PROJECT.CMSLIB]SYNTAX.PAS
reserved
```

This example creates a concurrent reservation for the element SYNTAX.PAS. Because you type YES in response to the Proceed prompt, generation 1 of the element is reserved.

3.
```
CMS>  RESERVE COPY.BLI/GENERATION=12/MERGE=11A1
_Remark: merging new I/O routines with library self checking
%CMS-W-MERGECONFLICT, 31 changes successfully merged with 3 conflicts
%CMS-S-RESERVED, generation 12 of element DISKX:[PROJECT.CMSLIB]COPY.BLI
reserved and merged with generation 11A1
```

This command merges generation 11A1 into generation 12 of the element COPY.BLI and reserves generation 12. The version that is delivered to the user directory contains the changes from both generations. Thirty-one changes were successfully merged. There were three conflicts between the two generations. These conflicts must be resolved by editing the file. For more information on merging, see Chapter 6.

RETRIEVE ARCHIVE

Retrieves one or more generations from one or more archive files.

Format

RETRIEVE ARCHIVE *file-expression*

Command Qualifiers	Defaults
/[NO]CONFIRM	/NOCONFIRM
/GENERATION[=generation-expression]	See text
/[NO]LOG	/LOG
/OUTPUT[=file-specification]	See text
/NOOUTPUT	

Command Parameter

file-expression
Specifies one or more archive files. A file expression can be a filename.type specification, a wildcard expression, or a list of these separated by commas.

Description

The RETRIEVE ARCHIVE command retrieves one or more generations of an element from one or more archive files. CMS retrieves the highest numbered generation from the archive file and places a copy of the generation in your default directory with the same file name and file type of the element whose generation was originally deleted (unless you specify another name or location with the /OUTPUT qualifier).

You can specify a particular generation to be retrieved with the /GENERATION qualifier. CMS creates one file for each retrieved generation. You do not need to have a library currently set to use this command, because the RETRIEVE ARCHIVE command does not interact with the CMS library.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/GENERATION[=generation-expression]
Specifies a particular generation to be retrieved from the archive file. If you omit /GENERATION, CMS retrieves the highest numbered generation in the archive file.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OUTPUT[=file-specification]
Directs CMS to write output to the specified file. If you omit the /OUTPUT qualifier (or if you specify /OUTPUT but do not provide a file specification), CMS creates a file with the same name as the archived element and the file type .CMS_ARCHIVE.

If you retrieve generations from more than one archive file and you do not specify wildcards in the output file specification, CMS creates successive versions of the file indicated by /OUTPUT.

Example

```
CMS>  RETRIEVE ARCHIVE/GENERATION=2A3 SAMPLE.CMS_ARCHIVE
%CMS-S-RETRIEVED, generation 2A3 of element DISKX:[PROJECT.CMSLIB]SAMPLE.PAS
retrieved from DISKX:[WORK]SAMPLE.CMS_ARCHIVE;1
```

This command retrieves generation 2A3 from the file SAMPLE.CMS_ARCHIVE in your default directory. CMS names generation 2A3 to its original element name SAMPLE.PAS and places it in your default directory.

REVIEW GENERATION

Associates a review comment with one or more specified element generations.

Format

REVIEW GENERATION *element-expression "remark"*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/NOCONFIRM
/GENERATION[=generation-expression]	See text
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL

Restrictions

- This command can be used only on element generations that have reviews pending (see Section 4.5.4 for more information).

Command Parameters

element-expression
Specifies one or more elements. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The REVIEW GENERATION command associates a review remark with the specified element generation. The review status of the generation must be pending. You can display the remarks associated with the generation by issuing the SHOW REVIEWS_PENDING command.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/GENERATION[=generation-expression]
Specifies the generation of the element with which to associate the review remark. If you omit /GENERATION, CMS uses the most recently created generation with a review pending. You specify this qualifier only if more than one generation of an element is under review.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)

Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, CLASS)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
CLASS (D)
NOCLASS
NONE—equivalent to (NOELEMENT, NOGROUP, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

Example

```
$  CMS REVIEW GENERATION EXAMPLE.SDML "looks ok to me -- JEFF"
%CMS-S-REVIEWED, generation 3 of element DISKX:[PROJECT.CMSLIB]EXAMPLE.SDML
reviewed
```

In this example, the latest generation of the element EXAMPLE.SDML is pending review; the REVIEW GENERATION command allows users to associate review comments with that generation. The generation can then be accepted, canceled, or rejected. Use the SHOW REVIEWS_PENDING command to display all generations under review in the library.

SET ACL

Manipulates the access control list (ACL) on various objects in the CMS library.

Format

SET ACL */OBJECT_TYPE=type object-expression "remark"*

Command Qualifiers	Defaults
/ACL[=(ace[,...])]	See text
/AFTER=ace	See text
/[NO]CONFIRM	/NOCONFIRM
/DEFAULT	See text
/DELETE	See text
/LIKE=object-specification	See text
/[NO]LOG	/LOG
/NEW	See text
/OBJECT_TYPE=type	See text
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/REPLACE=(ace[,...])	See text

Command Parameters

object-expression
Specifies one or more objects whose ACLs are to be modified. Wildcards and a comma list are allowed.

The object expression depends on the object type (see the /OBJECT_TYPE qualifier). For example, if the object type is CLASS, the object expression must be the name of a class in the CMS library. The same principle applies to elements and groups. However, if the object type is LIBRARY, the object expression must be one or more of the following keywords:

ELEMENT_LIST
CLASS_LIST
GROUP_LIST

HISTORY
LIBRARY_ATTRIBUTES

These keywords are referred to as object subtypes. You can abbreviate object subtypes. Wildcards are not allowed. See Chapter 7 for more information.

The object name can also be the name of a CMS command. If /OBJECT_TYPE is specified as COMMAND, SET ACL modifies the ACL on the given command. Commands that contain two words must be specified with an underscore, for example, INSERT_ELEMENT.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

Description

The SET ACL command is used to manipulate ACLs on various objects in the CMS library. ACLs are used to control access to individual CMS commands. ACLs are also used to control access to elements, groups, and classes, as well as on the lists containing these entities. An ACL can also be put on the entire library, and on the library history. For more information on using ACLs, see Chapter 7.

Command Qualifiers

/ACL[=(ace[,...])]
Specifies one or more access control entries (ACEs) to be modified. When no ACE is specified, the entire ACL is affected. Separate multiple ACEs with commas and enclose the list in parentheses. The specified ACEs are inserted at the beginning of the ACL unless the /AFTER qualifier is used.

/AFTER=ace
Indicates that all ACEs specified with the /ACL qualifier are added after the ACE specified with the /AFTER qualifier. By default, any ACEs added to the ACL are always placed at the top of the list.

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/DEFAULT
Creates an ACL for one or more specified objects as if the object were newly created. The /DEFAULT qualifier propagates the DEFAULT option ACEs in the ACL of the entity list to the ACL of the specified object. This qualifier can be used only with an object that is a library entity, that is, either an element, class, or group.

/DELETE
Indicates that the ACEs specified with the /ACL qualifier are to be deleted. If no ACEs are specified with the /ACL qualifier, the entire ACL is deleted. If the /ACL qualifier specifies an ACE that does not exist in the ACL of the specified object, you are notified that the ACE does not exist, and the delete operation continues on to the next ACE on the ACL, if any exists.

/LIKE=object-specification
Indicates that the ACL of the specified object is to replace the ACL of the object (or objects) specified with SET ACL. Any existing ACEs are deleted before the ACL specified by /LIKE is copied.

The type of the source and destination objects must be the same. No wildcard characters are allowed in the /LIKE parameter.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/NEW

Indicates that any existing ACEs in the ACL of the object specified with SET ACL are to be deleted. To use the /NEW qualifier, you must specify a new ACL or ACE with the /ACL qualifier.

/OBJECT_TYPE=type

Specifies the type of the object whose ACL is being modified. There is no default object type; therefore, this qualifier is required. The type must be one of the following keywords:

CLASS
ELEMENT
GROUP
LIBRARY
COMMAND

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)

Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, CLASS, OTHER)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
CLASS (D)
NOCLASS
OTHER (D)
NOOTHER
NONE—equivalent to (NOELEMENT, NOGROUP, NOCLASS, NOOTHER)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, [NO]CLASS, and [NO]OTHER keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/REPLACE=(ace[,...])
Deletes the ACEs specified with the /ACL qualifier and replaces them with those specified with /REPLACE. Any ACEs specified with the /ACL qualifier must exist and must be specified in the order in which they appear in the current ACL.

Examples

1.
```
CMS> SET ACL/OBJECT_TYPE=ELEMENT SAMPLE.PAS/ACL=(IDENTIFIER=WALLEN, -
_CMS> ACCESS=RESERVE+CONTROL) "setting up acl on element"
%CMS-S-MODACL, modified access control list for element
DISKX:[PROJECT.CMSLIB]SAMPLE.PAS
```

This command assigns an ACL on the element SAMPLE.PAS, specifying that the user holding the identifier WALLEN has RESERVE and CONTROL access to the element.

2.
```
CMS> SET ACL/OBJECT_TYPE=LIBRARY ELEMENT_LIST -
_CMS> /ACL=((IDENTIFIER=WALLEN,OPTIONS=DEFAULT,ACCESS=FETCH), -
_CMS> (IDENTIFIER=WALLEN,ACCESS=CREATE+CONTROL)) ""
%CMS-S-MODACL, modified access control list for subtype
DISKX:[PROJECT.CMSLIB]ELEMENT_LIST
```

This example shows how to assign two separate ACEs on the element list. The first ACE specifies a default ACE to be inherited by newly created elements in the library. The second ACE allows the user holding the identifier WALLEN to create elements in the library.

3.
```
CMS> SET ACL/OBJECT_TYPE=ELEMENT/DEFAULT SAMPLE.PAS ""
%CMS-S-MODACL, modified access control list for element
DISKX:[PROJECT.CMSLIB]SAMPLE.PAS

CMS> SHOW ACL/OBJECT_TYPE=ELEMENT SAMPLE.PAS

ACLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

SAMPLE.PAS
         (IDENTIFIER=[PROJECT,WALLEN],ACCESS=FETCH)
```

The SET ACL command causes the default ACE from the element list (see Example 2) to be placed on the existing element SAMPLE.PAS (new elements would inherit this default ACE automatically). The SHOW ACL command displays this ACE.

4.
```
CMS>  SET ACL/OBJECT_TYPE=CLASS BL1/ACL=(IDENTIFIER=[DEV,*]+LIBRARIAN, -
_CMS>  ACCESS=INSERT+REMOVE) ""
%CMS-S-MODACL, modified access control list for class DISKX:[PROJECT.CMSLIB]BL1
```

This command assigns an ACL allowing INSERT and REMOVE access to class BL1 for users in group DEV holding the LIBRARIAN identifier.

5.
```
CMS>  SET ACL/OBJECT_TYPE=CLASS/LIKE=BL1 BL2 ""
%CMS-S-MODACL, modified access control list for class DISKX:[PROJECT.CMSLIB]BL2
CMS>  SHOW ACL/OBJECT_TYPE=CLASS

ACLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

BL1
          (IDENTIFIER=[DEV,*]+LIBRARIAN,ACCESS=INSERT+REMOVE)

BL2
          (IDENTIFIER=[DEV,*]+LIBRARIAN,ACCESS=INSERT+REMOVE)
```

In this example, the /LIKE qualifier causes the ACL from the class BL1 (see Example 4) to be placed on the class BL2. The SHOW ACL command displays the ACL on both classes BL1 and BL2.

6.
```
CMS>  SET ACL/OBJECT_TYPE=ELEMENT SAMPLE.PAS -
_CMS>  /ACL=(IDENTIFIER=WALLEN,ACCESS=FETCH+CONTROL) ""
%CMS-S-MODACL, modified access control list for element
DISKX:[PROJECT.CMSLIB]SAMPLE.PAS

CMS>  SET ACL/OBJECT_TYPE=ELEMENT SAMPLE.PAS -
_CMS>  /ACL=(IDENTIFIER=BRADLEY,ACCESS=NONE) ""
%CMS-S-MODACL, modified access control list for element
DISKX:[PROJECT.CMSLIB]SAMPLE.PAS

CMS>  SHOW ACL/OBJECT_TYPE=ELEMENT SAMPLE.PAS

ACLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

SAMPLE.PAS
          (IDENTIFIER=[CMS,BRADLEY],ACCESS=NONE)
          (IDENTIFIER=[CMS,WALLEN],ACCESS=CONTROL+FETCH)

CMS>  SET ACL/OBJECT_TYPE=ELEMENT SAMPLE.PAS/AFTER=(IDENTIFIER=BRADLEY) -
_CMS>  /ACL=((IDENTIFIER=DAVIS,ACCESS=RESERVE+REPLACE), -
_CMS>  (IDENTIFIER=HENRY,ACCESS=MODIFY)) ""
%CMS-S-MODACL, modified access control list for element
DISKX:[PROJECT.CMSLIB]SAMPLE.PAS

CMS>  SHOW ACL/OBJECT_TYPE=ELEMENT SAMPLE.PAS

ACLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]
```

```
SAMPLE.PAS
         (IDENTIFIER=[CMS,BRADLEY],ACCESS=NONE)
         (IDENTIFIER=[CMS,DAVIS],ACCESS=REPLACE+RESERVE)
         (IDENTIFIER=[CMS,HENRY],ACCESS=MODIFY)
         (IDENTIFIER=[CMS,WALLEN],ACCESS=CONTROL+FETCH)
```

In this example, user WALLEN assigns an ACL giving himself FETCH and CONTROL access to SAMPLE.PAS. (Assume WALLEN has EXECUTE access to both the FETCH and SET ACL commands.) Wallen then assigns an ACE to user BRADLEY, restricting BRADLEY from any access to SAMPLE.PAS. By default, the ACE assigned to user BRADLEY is put at the beginning of the ACL.

WALLEN then allows REPLACE and RESERVE access to SAMPLE.PAS to user DAVIS, and MODIFY access to SAMPLE.PAS to user HENRY. The /AFTER qualifier causes the ACEs for DAVIS and HENRY to be placed after the ACE for user BRADLEY. The SHOW ACL command displays the order of the ACEs in the ACL.

7.
```
CMS> SET ACL/OBJECT_TYPE=ELEMENT SAMPLE.PAS -
_CMS> /ACL=((IDENTIFIER=BRADLEY),(IDENTIFIER=HENRY)) -
_CMS> /REPLACE=((IDENTIFIER=PROJ_MEMBERS,ACCESS=DELETE+MODIFY), -
_CMS> (IDENTIFIER=TALCOTT,ACCESS=NONE)) ""
%CMS-S-MODACL, modified access control list for element
DISKX:[PROJECT.CMSLIB]SAMPLE.PAS

CMS> SHOW ACL/OBJECT_TYPE=ELEMENT SAMPLE.PAS

ACLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

SAMPLE.PAS
         (IDENTIFIER=PROJ_MEMBERS,ACCESS=DELETE+MODIFY)
         (IDENTIFIER=[CMS,DAVIS],ACCESS=REPLACE+RESERVE)
         (IDENTIFIER=[CMS,TALCOTT],ACCESS=NONE)
         (IDENTIFIER=[CMS,WALLEN],ACCESS=CONTROL+FETCH)
```

In this example, the /REPLACE qualifier causes the ACEs belonging to user BRADLEY and user HENRY (see Example 6) to be replaced with new ACEs allowing DELETE and MODIFY access to SAMPLE.PAS for users with the PROJ_MEMBERS identifier and no access for user TALCOTT.

The SHOW ACL command displays the new order of the ACEs in the ACL.

8.
```
CMS> SET ACL/OBJECT_TYPE=ELEMENT SAMPLE.PAS/NEW -
_CMS> /ACL=((IDENTIFIER=WALLEN,ACCESS=MARK+REVIEW+CONTROL), -
_CMS> (IDENTIFIER=DICKAU,ACCESS=FETCH)) ""
%CMS-E-NOMODACL, error modifying access control list for element
DISKX:[PROJECT.CMSLIB]SAMPLE.PAS
-CMS-E-NOACCESS, no control access to element SAMPLE.PAS
```

In this example, user WALLEN uses the /NEW qualifier to delete the existing ACL on the element SAMPLE.PAS and specify a new ACL. The attempt is unsuccessful; although user WALLEN matches the ACE (with the [CMS,WALLEN] identifier) containing FETCH and CONTROL access to SAMPLE.PAS, he also matches the ACE (with the PROJ_MEMBERS identifier) containing DELETE and MODIFY access to SAMPLE.PAS. The ACE with the PROJ_MEMBERS identifier is matched *before* the ACE with the [CMS,WALLEN] identifier is reached. Thus, since PROJ_MEMBERS does not have the required CONTROL access to SAMPLE.PAS, WALLEN receives an error.

9.
```
CMS> SHOW ACL/OBJECT_TYPE=ELEMENT INIT.FOR

ACLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

INIT.FOR
         (IDENTIFIER=[CMS,WALLEN],ACCESS=CONTROL+BYPASS+REPLACE)

CMS> SET ACL/OBJECT_TYPE=ELEMENT INIT.FOR/NEW -
_CMS> /ACL=((IDENTIFIER=WALLEN,ACCESS=MODIFY+CONTROL), -
_CMS> (IDENTIFIER=PROJ_MEMBERS,ACCESS=NONE)) ""
%CMS-S-MODACL, modified access control list for element
DISKX:[PROJECT.CMSLIB]INIT.FOR

CMS> SHOW ACL/OBJECT_TYPE=ELEMENT INIT.FOR

ACLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

INIT.FOR
         (IDENTIFIER=[CMS,WALLEN],ACCESS=CONTROL+MODIFY)
         (IDENTIFIER=PROJ_MEMBERS,ACCESS=NONE)
```

In this example, WALLEN enters SHOW ACL to display any existing ACL on the element INIT.FOR. WALLEN successfully uses the /NEW qualifier to assign himself an ACE containing MODIFY and CONTROL access to INIT.FOR, and uses the ACCESS=NONE clause to delete any existing access for INIT.FOR from users holding the PROJ_MEMBERS identifier.

10.
```
CMS>  SET ACL/OBJECT_TYPE=ELEMENT INIT.FOR/DELETE/ACL=(IDENTIFIER=PROJ_MEMBERS) ""
%CMS-S-MODACL, modified access control list for element
DISKX:[PROJECT.CMSLIB]INIT.FOR

CMS>  SHOW ACL/OBJECT_TYPE=ELEMENT INIT.FOR

ACLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

INIT.FOR
         (IDENTIFIER=[CMS,WALLEN],ACCESS=CONTROL+MODIFY)

CMS>  SET ACL/OBJECT_TYPE=ELEMENT INIT.FOR/DELETE ""
%CMS-S-MODACL, modified access control list for element
DISKX:[PROJECT.CMSLIB]INIT.FOR

CMS>  SHOW ACL/OBJECT_TYPE=ELEMENT INIT.FOR

ACLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

INIT.FOR
```

In this example, the /DELETE qualifier is used to delete the existing ACL on the element INIT.FOR for users holding the PROJ_MEMBERS identifier. Then, WALLEN enters the /DELETE qualifier on the remaining ACE. Because no ACEs were specified with the /ACL qualifier, CMS deletes the entire ACL by default.

SET LIBRARY

Enables access to an existing CMS library (or libraries). Subsequent CMS commands automatically refer to the libraries identified by this command.

Format

SET LIBRARY *directory-specification[,...]*

Command Qualifiers	**Defaults**
/AFTER[=directory-specification]	See text
/BEFORE[=directory-specification]	See text
/[NO]LOG	/LOG
/[NO]VERIFY	/VERIFY

Command Parameter

directory-specification
Specifies one or more existing CMS libraries. The directory that is used as the CMS library cannot be your current default directory. The directory specification must conform to VMS conventions; it can also be a logical name. If you specify more than one VMS directory, you must separate the directory specifications with commas. Wildcards are not allowed.

Description

The SET LIBRARY command enables access to an existing CMS library or list of libraries. Subsequent CMS commands automatically refer to the library (or libraries) identified in the SET LIBRARY command. The SET LIBRARY command defines logical names beginning with CMS$ that allow CMS commands to refer implicitly to the library. You should not define logical names beginning with CMS$ because this prefix is reserved for CMS.

The SET LIBRARY command performs some consistency checks on the directory to verify that it is a valid CMS library (unless you specify the /NOVERIFY qualifier). If the library is not valid, you receive an error message.

The library must have been created with the CREATE LIBRARY command. During each session in which you want to use your CMS library, you must use the SET LIBRARY command before you access a library. The command is not required, however, if you have just created a library (see the description of the CREATE LIBRARY command) because the CREATE LIBRARY command performs an implicit SET LIBRARY.

You create a search list by specifying multiple libraries on the SET LIBRARY command. This enables you to manipulate several libraries with one command. You must include commas between the directory specifications.

Command Qualifiers

/AFTER[=directory-specification]
Instructs CMS to insert new libraries into the existing library search list (that you previously specified by using a comma list with the CREATE LIBRARY or SET LIBRARY command) immediately following the existing specified directory. If you omit the directory specification, CMS adds the libraries to the end of the list. You cannot specify both /AFTER and /BEFORE on the same command line. By default, the SET LIBRARY command's library list supersedes any existing search list.

/BEFORE[=directory-specification]
Instructs CMS to insert new libraries into the existing library search list (that you previously specified by using a comma list with the CREATE LIBRARY or SET LIBRARY command) immediately in front of the existing specified directory. If you omit the directory specification, CMS adds the libraries to the front of the list. You cannot specify both /AFTER and /BEFORE on the same command line. By default, the SET LIBRARY command's library list supersedes any existing search list.

/LOG (D)
/NOLOG

Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/VERIFY (D)
/NOVERIFY

Allows you to set a library without performing the locking and verification process CMS normally performs. This speeds up the SET LIBRARY operation and allows a CMS library to be set even if the library is locked by another user. However, if the library needs recovery, the condition is not detected until another transaction is attempted.

Examples

1.
```
CMS>  SET LIBRARY [WORK.CMSLIB]
%CMS-S-LIBIS, CMS library is DISKX:[WORK.CMSLIB]
```

   This command sets the CMS library to the existing library [WORK.CMSLIB].

2.
```
CMS>  SET LIBRARY [WORK.CMLSIB],[TEST.CMSLIB]
%CMS-I-LIBIS, library is DISK$:[WORK.CMSLIB]
%CMS-I-LIBINSLIS, library DISK$:[TEST.CMSLIB] inserted at end of library
list
%CMS-S-LIBSET, library set
-CMS-I-SUPERSEDE, library list superseded
```

   This command sets (or resets, if there was an existing library or libraries) the current library to contain two libraries.

3.
```
CMS> SET LIBRARY [PROJECT.CMSLIB]/AFTER=[WORK.CMSLIB]
%CMS-I-LIBINSLIS, library DISK$:[PROJECT.CMLSIB] inserted after
DISKX:[WORK.CMSLIB]
%CMS-S-LIBSET, library set
```

Assuming you set your library as in Example 2, this command directs CMS to insert the library [PROJECT.CMSLIB] after the library [WORK.CMSLIB]. The library list now contains the three libraries [WORK.CMSLIB], [PROJECT.CMSLIB], and [TEST.CMSLIB], in that order. Use the SHOW LIBRARY command to display the library search list.

SET NOLIBRARY

Removes one or more libraries from the current library search list.

Format

SET NOLIBRARY *[directory-specification[,...]]*

Command Qualifiers	**Defaults**
/[NO]LOG	/LOG

Command Parameter

directory-specification
Specifies one or more existing CMS libraries in the current library search list. The directory specification must conform to VMS conventions; it can also be a logical name that translates to a search list. If you specify more than one VMS directory, you must separate the directory specifications with commas. Wildcards are not allowed. If you do not supply a directory specification, CMS removes all libraries from the current library search list.

Description

The SET NOLIBRARY command removes one or more libraries from the current library search list. For more information on search lists, see Chapter 3. If all libraries are removed from the list, the logical name CMS$LIB is deassigned.

Command Qualifiers

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays

a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

Example

```
CMS>  SET NOLIBRARY
%CMS-W-UNDEFLIB, library is undefined
```

This example removes all existing libraries from the current library search list.

SHOW ACL

Displays the access control list (ACL) associated with the specified object (or objects).

Format

SHOW ACL *object-expression /OBJECT_TYPE=type*

Command Qualifiers	**Defaults**
/[NO]APPEND	/NOAPPEND
/OBJECT_TYPE = type	See text
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/OUTPUT[=file-specification]	/OUTPUT=SYS$OUTPUT

Command Parameter

object-expression
Specifies the CMS library object whose ACL is to be displayed. Wildcards and a comma list are allowed.

The object name depends on the object type (see the /OBJECT_TYPE qualifier). For example, if the object type is CLASS, the object name is the name of a class in the CMS library. The same principle applies to elements and groups. If the object type is LIBRARY, the object expression must be a list of one or more of the following keywords:

ELEMENT_LIST
CLASS_LIST
GROUP_LIST
HISTORY
LIBRARY_ATTRIBUTES

These keywords are called object subtypes. You can abbreviate object subtypes. Wildcards are not allowed.

The object name can also be the name of a CMS command. If /OBJECT_TYPE is specified as COMMAND, SHOW ACL displays the ACL for the given command.

Description

The SHOW ACL command displays the ACL associated with the specified object (or objects).

Command Qualifiers

/APPEND
/NOAPPEND (D)
Controls whether CMS appends the command output to an existing file, or creates a new file. If you specify /APPEND and the output file does not exist, CMS creates a new file. If you do not provide a file specification (see the description for /OUTPUT), the output is appended to SYS$OUTPUT.

/OBJECT_TYPE=type
Specifies the type of the object whose ACL is to be displayed. There is no default object type; therefore, this qualifier is required. The object type can be one of the following keywords:

ELEMENT
CLASS
GROUP
LIBRARY
COMMAND

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, CLASS, OTHER)
ELEMENT (D)
NOELEMENT

GROUP (D)
NOGROUP
CLASS (D)
NOCLASS
OTHER (D)
NOOTHER
NONE—equivalent to (NOELEMENT, NOGROUP, NOCLASS, NOOTHER)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, [NO]CLASS, and [NO]OTHER keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/OUTPUT[=file-specification]
/OUTPUT=SYS$OUTPUT (D)
Directs CMS to write output to the specified file, except for any warning and error messages, which are written to SYS$OUTPUT and SYS$ERROR. CMS creates a new file if you do not specify /APPEND. If you omit the /OUTPUT qualifier (or if you specify /OUTPUT but do not provide a file specification), CMS directs output to the default output device (SYS$OUTPUT). If you omit either the file name or the file-type component, CMS supplies the missing component from the default specification.

Example

```
CMS> SHOW ACL/OBJECT_TYPE=ELEMENT SAMPLE.PAS

ACLs in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

SAMPLE.PAS
           (IDENTIFIER=[PROJECT,WALLEN],ACCESS=FETCH)
```

This command displays the ACE on element SAMPLE.PAS.

SHOW ARCHIVE

Displays information about one or more archive files.

Format

SHOW ARCHIVE *file-expression*

Command Qualifiers	**Defaults**
/[NO]APPEND	/NOAPPEND
/BRIEF	See text
/FULL	See text
/INTERMEDIATE	See text
/OUTPUT[=file-specification]	/OUTPUT=SYS$OUTPUT

Command Parameter

file-expression
Specifies the name of the archive file. The file expression can be a file-name.type specification, a wildcard expression, or a list of these separated by commas.

Description

The SHOW ARCHIVE command displays information about the contents of one or more specified archive files.

Command Qualifiers

/APPEND
/NOAPPEND (D)
Controls whether CMS appends the command output to an existing file, or creates a new file. If you specify /APPEND and the output file does not exist,

CMS creates a new file. If you do not provide an output file specification (see the description for /OUTPUT), the output is appended to SYS$OUTPUT.

/BRIEF
/FULL
/INTERMEDIATE (D)
The /BRIEF qualifier displays the name of the element, the generations archived into this file, the name of the person who archived the file, the date and time, the remark entered on the DELETE GENERATION command, and the name of the library in which the original element resided. The /FULL qualifier displays complete generation file information for each archived generation. The /INTERMEDIATE qualifier displays the generation history for the archived generations.

/OUTPUT[=file-specification]
/OUTPUT=SYS$OUTPUT (D)
Directs CMS to write output to the specified file, except for any warning and error messages, which are written to SYS$OUTPUT and SYS$ERROR. CMS creates a new file if you do not specify /APPEND. If you omit the /OUTPUT qualifier (or if you specify /OUTPUT but do not provide a file specification), CMS directs output to the default output device (SYS$OUTPUT). If you omit either the file name or the file-type component, CMS supplies the missing component from the default specification.

Examples

1.
```
CMS>  SHOW ARCHIVE/BRIEF SAMPLE.CMS_ARCHIVE
25-JAN-1990 17:08:47 JONES DISKX:[WORK.CMSLIB] SAMPLE.PAS(2A1 through 2A3) "delete
the variant range and archive the deleted generations"
```

This command displays information about the archive file, consisting of the date and time entered with the DELETE GENERATION command, the name of the person who archived the file, the library in which the original element resided, the generations that were archived into the file, and the remark entered on the DELETE GENERATION command.

```
2.  CMS> SHOW ARCHIVE/FULL SAMPLE.CMS_ARCHIVE

    25-JAN-1990 17:08:47 JONES DISKX:[WORK.CMSLIB] SAMPLE.PAS(2A1 through 2A3) "delete
    the variant range and archive the deleted generations"
        2A1     22-JAN-1990 14:27:41 JONES "reserved concurrently to add routines"
            File creation:      22-JAN-1990 14:27
            File revision:      24-JAN-1990 17:12 (1)
            Record format:      Variable length
            Record attributes:  Carriage return carriage control
            Review status:      None

        2A2     22-JAN-1990 14:36:56 JONES "another routine added"
            File creation:      22-JAN-1990 14:36
            File revision:      22-JAN-1990 18:01 (1)
            Record format:      Variable length
            Record attributes:  Carriage return carriage control
            Review status:      None

        2A3     22-JAN-1990 14:45:12 JONES "last one I promise"
            File creation:      22-JAN-1990 14:45
            File revision:      27-JAN-1990 08:30 (1)
            Record format:      Variable length
            Record attributes:  Carriage return carriage control
            Review status:      None
```

This command displays complete generation file information for each archived generation in the archive file.

SHOW CLASS

Displays information about one or more classes in a CMS library.

Format

SHOW CLASS *[class-expression]*

Command Qualifiers	**Defaults**
/[NO]APPEND	/NOAPPEND
/BRIEF	See text
/[NO]CONTENTS	/NOCONTENTS
/FULL	See text
/INTERMEDIATE	See text
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/OUTPUT[=file-specification]	/OUTPUT=SYS$OUTPUT

Command Parameter

class-expression
Specifies one or more classes to be listed. If you do not specify a class name, CMS lists all classes in the library. A class expression can be a class name, a wildcard expression, or a list of these separated by commas.

Description

The SHOW CLASS command lists the names of all established classes in alphabetical order, along with the associated creation remarks.

Command Qualifiers

/APPEND
/NOAPPEND (D)
Controls whether CMS appends the command output to an existing file, or creates a new file. If you specify /APPEND and the output file does not exist, CMS creates a new file. If you do not provide an output file specification (see the description for /OUTPUT), the output is appended to SYS$OUTPUT.

/BRIEF
/FULL
/INTERMEDIATE (D)
The /BRIEF qualifier displays only the class names. The /FULL qualifier displays the name, creation remark, and **read-only** attribute (if established) for each class. The /INTERMEDIATE qualifier displays the name and creation remark for each class.

/CONTENTS
/NOCONTENTS (D)
Controls whether CMS identifies the element generations that belong to each class. If you specify /CONTENTS, CMS displays the class name and the creation remark, along with the element name and generation number for each generation in the class.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to CLASS
CLASS (D)
NOCLASS
NONE—equivalent to NOCLASS

You can specify either ALL, NONE, or the [NO]CLASS keyword.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/OUTPUT[=file-specification]
/OUTPUT=SYS$OUTPUT (D)
Directs CMS to write output to the specified file, except for any warning and error messages, which are written to SYS$OUTPUT and SYS$ERROR. CMS creates a new file if you do not specify /APPEND. If you omit the /OUTPUT qualifier (or if you specify /OUTPUT but do not provide a file specification), CMS directs output to the default output device (SYS$OUTPUT). If you omit either the file name or the file-type component, CMS supplies the missing component from the default specification.

Examples

1.
```
CMS> SHOW CLASS

Classes in CMS Library DISKX:[RELEASE.CMSLIB]

INTERNAL_RELEASE   "for internal use only"
MESSAGETEST     "filter evolution checks"
PASCAL_CLASS    "PASCAL tests"
```

   This command displays the class name and the creation remark for each class in the library.

2.
```
CMS> SHOW CLASS/BRIEF

Classes in CMS Library DISKX:[RELEASE.CMSLIB]

INTERNAL_RELEASE       MESSAGETEST          PASCAL_CLASS
```

   This command limits the output to class names only.

SHOW ELEMENT

Displays information about one or more elements in a CMS library.

Format

SHOW ELEMENT *[element-expression]*

Command Qualifiers	**Defaults**
/[NO]APPEND	/NOAPPEND
/BRIEF	See text
/FORMAT="string"	/NOFORMAT
/NOFORMAT	
/FULL	See text
/INTERMEDIATE	See text
/[NO]MEMBER	/NOMEMBER
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/OUTPUT[=file-specification]	/OUTPUT=SYS$OUTPUT

Command Parameter

element-expression
Specifies one or more elements to be listed. If you do not supply an element expression, CMS lists all the elements in the library. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas.

Description

The SHOW ELEMENT command lists the name of each specified element in alphabetical order, along with the remark logged at the time the element was created or modified. You can also specify qualifiers that provide information about element attributes, concurrent access, and the groups to which the element belongs.

Command Qualifiers

/APPEND
/NOAPPEND (D)
Controls whether CMS appends the command output to an existing file, or creates a new file. If you specify /APPEND and the output file does not exist, CMS creates a new file. If you do not provide an output file specification (see the description for /OUTPUT), the output is appended to SYS$OUTPUT.

/BRIEF
/FULL
/INTERMEDIATE (D)
The /BRIEF qualifier displays only the element names. The /FULL qualifier displays the name, creation remark, and the attributes in effect for the specified elements. The /INTERMEDIATE qualifier displays the name and creation remark associated with the element.

/FORMAT="string"
/NOFORMAT (D)
Controls whether the output of the SHOW ELEMENT command is formatted. You can use the /FORMAT qualifier in combination with the /OUTPUT qualifier to set up a command file. With this command file, you can execute a CMS command or a DCL command on a specified set of elements (such as all elements in a group).

The format string can contain printing characters; within the format string, CMS recognizes #E (and #e) as the element format parameter. For each line of output (one line per element), CMS displays the format string and replaces each occurrence of #E (or #e) with the element name. To include a number sign in the output line, type it twice (##). When you specify /FORMAT, CMS does not generate the heading normally produced by the SHOW ELEMENT command.

To set up a command file, you specify a format string consisting of a command, including the dollar sign ($) prompt and the element format parameter (for example, /FORMAT="$ CMS FETCH #E"). Use the /OUTPUT qualifier to direct the output to a command file. When you execute the SHOW ELEMENT command with these qualifiers, CMS creates a command file containing a list of FETCH commands that use each element in the denoted set as parameters.

/MEMBER
/NOMEMBER (D)
Lists the element name, creation remark, and the names of any groups to which the element belongs.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
NONE—equivalent to (NOELEMENT, NOGROUP)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT and [NO]GROUP keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/OUTPUT[=file-specification]
/OUTPUT=SYS$OUTPUT (D)
Directs CMS to write output to the specified file, except for any warning and error messages, which are written to SYS$OUTPUT and SYS$ERROR. CMS creates a new file if you do not specify /APPEND. If you omit the /OUTPUT qualifier (or if you specify /OUTPUT but do not provide a file specification), CMS directs output to the default output device (SYS$OUTPUT). If you omit either the file name or the file-type component, CMS supplies the missing component from the default specification.

Examples

1.
   ```
   CMS> SHOW ELEMENT/BRIEF

   Elements in CMS Library DISKX:[PROJECT.CMSLIB]

   CCM.FOR              CCM1.FOR             COPYTEST.FOR
   ```

   This command limits the output to element names only.

2.
   ```
   CMS> SHOW ELEMENT/FORMAT="$ CMS FETCH #E"/OUTPUT=FETCH.COM
   ```

   This command produces a file named FETCH.COM that contains a FETCH command for each element in the library. The contents of a file produced by this command might look like the following example:

   ```
   $ CMS FETCH INIT.FOR
   $ CMS FETCH INITX.FOR
   $ CMS FETCH MSGDOC.FOR
   $ CMS FETCH OUTPUT.FOR
   $ CMS FETCH SEARCH.FOR
   ```

SHOW GENERATION

Displays information about one or more element generations in a CMS library.

Format

SHOW GENERATION *[element-expression]*

Command Qualifiers	**Defaults**
/[NO]ANCESTORS	/NOANCESTORS
/[NO]APPEND	/NOAPPEND
/BRIEF	See text
/[NO]DESCENDANTS	/NODESCENDANTS
/FORMAT="string"	/NOFORMAT
/FROM=generation-expression	/FROM=1
/FULL	See text
/GENERATION[=generation-expression]	See text
/INTERMEDIATE	/See text
/[NO]MEMBER	/NOMEMBER
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/OUTPUT[=file-specification]	/OUTPUT=SYS$OUTPUT

Restrictions

- You cannot specify the /ANCESTORS and /DESCENDANTS qualifiers on the same command line.
- You cannot specify the /DESCENDANTS and /FROM qualifiers on the same command line.
- If you specify the /FROM qualifier, you must also specify the /ANCESTORS qualifier on the same command line.

SHOW GENERATION

Command Parameter

element-expression
Specifies one or more elements. For each element, CMS lists the transaction record of the most recent generation on the main line of descent. If you do not supply an element expression, CMS lists a transaction record for each element in the library. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas.

Description

The SHOW GENERATION command lists information about a specific generation of an element or a group of elements. This command displays the transaction record of the latest generation on the main line of descent. You can also use SHOW GENERATION to display the element name and generation number of all element generations belonging to a specified group or class.

Command Qualifiers

/ANCESTORS
/NOANCESTORS (D)
Displays the transaction records of the specified element generation and all of its ancestors. The transaction records are listed in reverse chronological order. If you do not specify a particular generation, the list begins with the latest generation on the main line of descent.

The ancestors of a main-line generation are all the preceding generations back to the first generation of an element. The ancestors of a variant-line generation are all preceding generations on the variant line of descent, and any generations back to the first generation on the main line.

/APPEND
/NOAPPEND (D)
Controls whether CMS appends the command output to an existing file, or creates a new file. If you specify /APPEND and the output file does not exist, CMS creates a new file. If you do not provide an output file specification (see the description for /OUTPUT), the output is appended to SYS$OUTPUT.

/BRIEF
/FULL
/INTERMEDIATE (D)
The /BRIEF qualifier displays only the element names and generation numbers for each specified generation. The /FULL qualifier displays standard CMS transaction information (the element name, generation number, date, time, user, and remark), and also produces information about the file creation and revision date and time, the file revision number, and the record format and attributes. The /INTERMEDIATE qualifier displays the element name, generation number, date, time, user, and remark associated with the transaction that created the generation.

/DESCENDANTS
/NODESCENDANTS (D)
Displays the transaction records of the specified element generation and of all its descendants. The transaction records are listed in reverse chronological order. If you do not specify a particular generation, the list begins with generation 1.

The descendants of a generation consist of all the successor generations, including those on variant lines of descent. Thus, you can use this command to determine whether any variant lines of descent exist for a particular element.

/FORMAT="string"
/NOFORMAT (D)
Controls whether the output of the SHOW GENERATION command is formatted. You can use the /FORMAT qualifier in combination with the /OUTPUT qualifier to set up a command file. With this command file, you can execute a CMS command or a DCL command on a specified set of elements (such as all elements in a class).

The format string can contain printing characters; within the format string, CMS recognizes #E (and #e) as the element format parameter, and #G (and #g) as the generation number format parameter. For each line of output (one line per generation), CMS displays the format string and replaces each occurrence of #E or #G with the element name or generation number, respectively. To include a number sign in the output line, type it twice (##). When you specify the /FORMAT qualifier, CMS does not generate the heading normally produced by the SHOW GENERATION command.

To set up a command file, you specify a format string consisting of a command, including the dollar sign ($) prompt and a format parameter (#E or #G) (for example, /FORMAT="$ CMS FETCH #E/GENERATION=#G"). Use the /OUTPUT qualifier to direct the output to a command file. When you execute the SHOW GENERATION command with these qualifiers, CMS creates a command file containing a list of FETCH commands that use each element in the denoted set as parameters.

/FROM=generation-expression
/FROM=1 (D)
Specifies the generation that begins the list of ancestors. You must specify the /ANCESTORS and /FROM qualifiers on the same command.

/GENERATION[=generation-expression]
Specifies the generation about which you want information. When you use the /GENERATION qualifier with the element-name parameter, the transaction record of the indicated generation is displayed.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/MEMBER
/NOMEMBER (D)
Lists the element name, generation number, and the names of any classes to which the element generation belongs.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, CLASS)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
CLASS (D)
NOCLASS
NONE—equivalent to (NOELEMENT, NOGROUP, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/OUTPUT[=file-specification]
/OUTPUT=SYS$OUTPUT (D)
Directs CMS to write output to the specified file, except for any warning and error messages, which are written to SYS$OUTPUT and SYS$ERROR. CMS creates a new file if you do not specify /APPEND. If you omit the /OUTPUT qualifier (or if you specify /OUTPUT but do not provide a file specification), CMS directs output to the default output device (SYS$OUTPUT). If you omit either the file name or the file-type component, CMS supplies the missing component from the default specification.

Examples

1.
```
CMS> SHOW GENERATION/BRIEF

Element generations in CMS Library DISKX:[TAYLOR.CMSLIB]

CCM.FOR/2             CCM1.FOR/1           COPYTEST.FOR/2
```

This command directs CMS to display only the element name and the number of the latest main-line generation for each element. Because no element is specified in the command line, CMS displays information about all elements in the library.

2.
```
CMS> SHOW GENERATION CCM.FOR

Element generations in CMS Library DISKX:[TAYLOR.CMSLIB]

CCM.FOR       2       7-DEC-1990 14:15:51 SMITH "header changed"
```

This command displays the element name, generation number, date, time, and remark associated with the latest main-line generation for element CCM.FOR.

3.
```
CMS> SHOW GENERATION/FULL CCM.FOR
Element generations in CMS Library DISKX:[TAYLOR.CMSLIB]

CCM.FOR      2       6-MAR-1990 17:34:04 SMITH "header changed"
        File creation:       6-MAR-1990 17:24
        File revision:       6-MAR-1990 17:24 (1)
        Record format:      Variable length
        Record attributes:  Carriage return carriage control
```

This command produces information about the file creation and revision date and time, the file revision number, and the record format and attributes, in addition to the standard CMS transaction information (element name, generation number, date, time, user, and remark). This additional information describes the file that was used to create the particular element generation, in this case generation 2 of CCM.FOR.

4.
```
CMS> SHOW GENERATION/GENERATION=RELEASE5 -
_CMS> /FORMAT="$ CMS FETCH #E/GENERATION=#G"/OUTPUT=FETCH_CLASS.COM
```

This example produces a file named FETCH_CLASS.COM that contains a FETCH command for each element that belongs to the class named RELEASE5. The FETCH command retrieves the element of the correct generation from the RELEASE5 class. The contents of a file produced by this command might look like the following:

```
$ CMS FETCH INIT.FOR/GENERATION=6
$ CMS FETCH OUTPUT.FOR/GENERATION=7
$ CMS FETCH SEARCH.FOR/GENERATION=3
```

SHOW GROUP

Displays information about one or more groups in a CMS library.

Format

SHOW GROUP *[group-expression]*

Command Qualifiers	Defaults
/[NO]APPEND	/NOAPPEND
/BRIEF	See text
/CONTENTS[=n]	/NOCONTENTS
/NOCONTENTS	
/FULL	See text
/INTERMEDIATE	See text
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/OUTPUT[=file-specification]	/OUTPUT=SYS$OUTPUT

Command Parameter

group-expression
Specifies the group to be listed. If you do not supply a group expression, CMS lists all groups in the library. A group expression can be a group name, a wildcard expression, or a list of these separated by commas.

Description

The SHOW GROUP command lists the names of all established groups in alphabetical order, along with the remark logged at the time each group was created or modified.

Command Qualifiers

/APPEND
/NOAPPEND (D)
Controls whether CMS appends the command output to an existing file, or creates a new file. If you specify /APPEND and the output file does not exist, CMS creates a new file. If you do not provide an output file specification (see the description for /OUTPUT), the output is appended to SYS$OUTPUT.

/BRIEF
/FULL
/INTERMEDIATE (D)
The /BRIEF qualifier displays only the group names. The /FULL qualifier displays the group name, creation remark, and **read-only** attribute (if established) for each group. The /INTERMEDIATE qualifier displays the name and creation remark associated with each group.

/CONTENTS[=n]
/NOCONTENTS (D)
Lists the group name and creation remark, along with the names of any elements or groups contained within the specified group.

You can specify an integer value (n) that directs CMS to display nested groups up or down to and including the level indicated by n. When you specify /CONTENTS without a value, CMS displays one level of contents. You can also specify /CONTENTS=ALL to display all levels of contained groups.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to GROUP
GROUP (D)
NOGROUP
NONE—equivalent to NOGROUP

You can specify either ALL, NONE, or the [NO]GROUP keyword.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/OUTPUT[=file-specification]
/OUTPUT=SYS$OUTPUT (D)
Directs CMS to write output to the specified file, except for any warning and error messages, which are written to SYS$OUTPUT and SYS$ERROR. CMS creates a new file if you do not specify /APPEND. If you omit the /OUTPUT qualifier (or if you specify /OUTPUT but do not provide a file specification), CMS directs output to the default output device (SYS$OUTPUT). If you omit either the file name or the file-type component, CMS supplies the missing component from the default specification.

Examples

1.
```
CMS> SHOW GROUP

Groups in DEC/CMS Library DISKX:[PROJECT.CMSLIB]

ARTWORK   "graph display"
OUT_DEFS  "output definition files"
TEST_ART  "tests for graphs"
```

This command displays the group name and the creation remark associated with the group, by default. Because no group name is specified on the command line, CMS lists all groups in the library.

2.
```
CMS> SHOW GROUP ARTWORK/CONTENTS

Groups in DEC/CMS Library DISKX:[WORK.CMSV3.CALL.TEST.CMSLIB]

ARTWORK    "graph display"
        5666.ARTWRK
        CHAP_ART.TXT
        FIG3.FIG
        MODPARAM5.SDML
        TEST_ART
```

This command produces a list of elements and groups that are members of group ARTWORK. The group TEST_ART was inserted into ARTWORK with the INSERT GROUP command; thus, as the contents of TEST_ART change, the elements accessible in the group ARTWORK change.

SHOW HISTORY

Displays a chronological list of transactions performed on a CMS library. You can limit the number of transactions that are displayed by specifying different parameters and qualifiers on the command.

Format

SHOW HISTORY *[object-expression]*

Command Qualifiers	**Defaults**
/[NO]APPEND	/NOAPPEND
/BEFORE=date-time	See text
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/OUTPUT[=file-specification]	/OUTPUT=SYS$OUTPUT
/SINCE=date-time	See text
/[NO]TRANSACTIONS=(keyword,...)	/TRANSACTIONS=ALL
/[NO]UNUSUAL	/NOUNUSUAL
/USER=username	See text

Command Parameter

object-expression
Specifies an object about which history information is to be displayed. The object expression can be one or more class names, element names, group names, a wildcard expression, or a list of any of these, separated by commas. If you do not specify an object expression, CMS lists all classes, elements, and groups in the library.

Description

The SHOW HISTORY command lists information about CMS library transactions. CMS records all transactions that update the library. The following transactions are not logged in the library history:

ANNOTATE	SET LIBRARY
DIFFERENCES	SHOW commands
FETCH (no remark)	VERIFY (no qualifiers)
RETRIEVE ARCHIVE	

The SHOW HISTORY command displays a chronological list of transactions. The transaction records are displayed in the following format:

date time username command remark

A space separates each item in the transaction record. Each item is explained as follows.

date

Displays the date the command was entered, in the following format:

[d]d-mmm-yyyy

For example:

```
9-JUN-1990
```

time

Displays the time the command was entered, in the following format:

hh:mm:ss

For example:

```
17:12:12
09:02:33
```

username

Displays the user name of the person who entered the command.

command
Identifies the command that was entered. The string “CMS” does not appear as part of the command. The command name, option name (if any), and any parameters are displayed. If the command operates on a particular generation of an element, the element name is followed by the generation number enclosed in parentheses; for example, SEMANTICS.PAS(3). Qualifiers that indicate some modification of the library are logged; for example, the /NAME qualifier with a MODIFY command.

“remark”
Specifies a character string that was logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is displayed.

The following example shows one transaction record:

```
11-AUG-1990 11:54:15 TAYLOR REPLACE SYNTAX.PAS(3) "RECORD and FOR loop syntax compatible'
```

This record shows that on August 11, 1990, at 11:54, user TAYLOR entered the REPLACE command on SYNTAX.PAS, which created generation 3.

For any command that caused CMS to record an unusual occurrence, an asterisk (*) is displayed in the first column.

Note that because transaction records for elements, groups, or classes deleted from a library are retained (see DELETE ELEMENT, DELETE GROUP, and DELETE CLASS), SHOW HISTORY can display records for elements and classes that do not currently exist. If a deleted name is reused, SHOW HISTORY does not distinguish between the old and the new histories.

Command Qualifiers

/APPEND
/NOAPPEND (D)
Controls whether CMS appends the command output to an existing file, or creates a new file. If you specify /APPEND and the output file does not exist, CMS creates a new file. If you do not provide an output file specification (see the description for /OUTPUT), the output is appended to SYS$OUTPUT.

/BEFORE=date-time
Lists all history information prior to a specified date. By default, the time is the current date and time. The time value can be an absolute, delta, or combination time value, or one of the following keywords: TODAY, TOMORROW, or YESTERDAY.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to OTHER
OTHER (D)
NOOTHER
NONE—equivalent to NOOTHER

You can specify either ALL, NONE, or the [NO]OTHER keyword.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/OUTPUT[=file-specification]
/OUTPUT=SYS$OUTPUT (D)
Directs CMS to write output to the specified file, except for any warning and error messages, which are written to SYS$OUTPUT and SYS$ERROR. CMS creates a new file if you do not specify /APPEND. If you omit the /OUTPUT qualifier (or if you specify /OUTPUT but do not provide a file specification), CMS directs output to the default output device (SYS$OUTPUT). If you omit either the file name or the file-type component, CMS supplies the missing component from the default specification.

/SINCE=date-time
Specifies that only those history entries dated on or after the given time are to be displayed. The time value can be an absolute, delta, or combination time value, or one of the following keywords: TODAY, TOMORROW, or YESTERDAY. When the /SINCE qualifier is omitted, all transactions that occurred since the library was created are displayed. If you specify the /SINCE qualifier but do not specify a value, CMS defaults to /SINCE=TODAY.

/[NO]TRANSACTIONS=(keyword,...)
/TRANSACTIONS=ALL (D)
Displays all transaction records generated by a specific command. You can specify the following keywords with this qualifier:

ACCEPT	FETCH	REMOVE
ALL	INSERT	REPLACE
CANCEL	MARK	RESERVE
COPY	MODIFY	REVIEW
CREATE	REJECT	SET
DELETE	REMARK	UNRESERVE
		VERIFY

If you specify more than one keyword, you must enclose the keyword list in parentheses. The /TRANSACTIONS qualifier directs CMS to list transaction records for only the listed keywords. The /NOTRANSACTIONS qualifier directs CMS to list transaction records for all keywords except the listed keywords.

/UNUSUAL
/NOUNUSUAL (D)
Controls whether transactions recorded as unusual occurrences are displayed. If you specify /UNUSUAL, only unusual transactions are displayed. /NOUNUSUAL displays all transactions, including unusual occurrences. The following CMS transactions are defined as unusual occurrences:

- A RESERVE command specifying an element that is already reserved
- A concurrent replacement
- A VERIFY/RECOVER command
- A VERIFY/REPAIR command
- A CONVERT LIBRARY command
- A REMARK/UNUSUAL command

/USER=username
Lists the transactions executed by the specified user.

Example

```
CMS>  SHOW HISTORY
History of CMS Library DISKX:[PROJECTLIB]

 15-JAN-1990 08:45:15 SNOW CREATE LIBRARY [PROJECTLIB] "code library"
 25-JAN-1990 10:23:42 SNOW CREATE ELEMENT STRING.LIB(1) "string defs"
 25-JAN-1990 13:37:57 SCHULTZ CREATE ELEMENT SIGNAL.LIB(1) "signaling defs"
 25-JAN-1990 14:09:37 PADDOCK CREATE ELEMENT CONVRT.BLI(1) "gen conversion util"
 26-JAN-1990 13:59:16 PADDOCK DELETE ELEMENT CONVRT.BLI "wrong format"
 27-JAN-1990 10:51:05 PADDOCK CREATE ELEMENT CNVNUM.BLI(1) "conversion routines"
 10-FEB-1990 10:51:48 SNOW CREATE ELEMENT SYSUTL.BLI(1) "general utilities"
 21-FEB-1990 11:11:55 SNOW RESERVE STRING.LIB(1) "change definitions"
 23-FEB-1990 11:22:31 SNOW REPLACE STRING.LIB(2) "added characters"
 23-FEB-1990 11:23:41 PADDOCK RESERVE CNVNUM.BLI(1) "change constant values"
 23-FEB-1990 12:15:21 SNOW FETCH STRING.LIB(1) "check differences"
 23-FEB-1990 12:20:42 PADDOCK REPLACE CNVNUM.BLI(2) "constant values changed"
          .
          .
          .
```

This command shows part of the early history of a library.

SHOW LIBRARY

Displays the directory specification of the current CMS library. SHOW LIBRARY also checks whether the directory is a valid CMS library (unless you specify the /BRIEF qualifier).

Format

SHOW LIBRARY

Command Qualifiers	Defaults
/[NO]APPEND	/NOAPPEND
/BRIEF	See text
/FULL	See text
/INTERMEDIATE	See text
/OUTPUT[=file-specification]	/OUTPUT=SYS$OUTPUT

Description

The SHOW LIBRARY command identifies the current CMS library. If you have not designated a CMS library for the current session (with SET LIBRARY or CREATE LIBRARY), this command issues an error message.

Command Qualifiers

/APPEND
/NOAPPEND (D)
Controls whether CMS appends the command output to an existing file, or creates a new file. If you specify /APPEND and the output file does not exist, CMS creates a new file. If you do not provide an output file specification (see the description for /OUTPUT), the output is appended to SYS$OUTPUT.

/BRIEF
/FULL
/INTERMEDIATE (D)
The /BRIEF qualifier displays the directory specification (and does not lock the library) even if the default library is locked by another user. However, if the library needs recovery, it is not detected until another operation is performed. The /FULL qualifier displays the current library directory specification, the number of elements, groups, classes, reservations, concurrent replacements, reviews pending, and the current reference copy directory specification, if any. The /INTERMEDIATE qualifier displays the directory specification of the current CMS library.

/OUTPUT[=file-specification]
/OUTPUT=SYS$OUTPUT (D)
Directs CMS to write output to the specified file, except for any warning and error messages, which are written to SYS$OUTPUT and SYS$ERROR. CMS creates a new file if you do not specify /APPEND. If you omit the /OUTPUT qualifier (or if you specify /OUTPUT but do not provide a file specification), CMS directs output to the default output device (SYS$OUTPUT). If you omit either the file name or the file-type component, CMS supplies the missing component from the default specification.

Examples

1.
```
CMS> SHOW LIBRARY
Your CMS library list consists of:
  DISKX:[PROJECT.CMSLIB]
```

This command displays the device and directory where the current CMS library is located.

2. `CMS>  SHOW LIBRARY/FULL`

```
Your CMS library list consists of:

  DISKX:[CODE.CMSLIB]
      and contains

        224 elements
          4 groups
          7 classes
          6 reservations
          0 concurrent replacements
          5 reviews pending

    REFERENCE_COPY directory is DISKX:[CODE.REQUIRE]
```

This command directs CMS to display additional information about the contents of the library.

SHOW RESERVATIONS

Displays all current reservations and concurrent replacements in effect at the time the command is issued.

Format

SHOW RESERVATIONS *[element-expression]*

Command Qualifiers	**Defaults**
/[NO]APPEND	/NOAPPEND
/BEFORE=date-time	See text
/CONCURRENT	See text
/GENERATION[=generation-expression]	See text
/NOREPLACEMENTS	See text
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/OUTPUT[=file-specification]	/OUTPUT=SYS$OUTPUT
/SINCE=date-time	See text
/USER=username	See text

Command Parameter

element-expression
Specifies one or more elements for which CMS displays current reservation and concurrent replacement information. If you do not specify an element expression, information is displayed for all elements. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas.

SHOW RESERVATIONS

Description

The SHOW RESERVATIONS command provides information about current reservations and concurrent replacements. For each element that has one or more generations reserved, the output consists of the element name and the following:

- For each reservation—identification number, user, reserved generation number, date, time, and remark
- For each concurrent replacement—user, replaced generation number, date, time, and remark

The elements are listed in alphabetical order.

Command Qualifiers

/APPEND
/NOAPPEND (D)
Controls whether CMS appends the command output to an existing file, or creates a new file. If you specify /APPEND and the output file does not exist, CMS creates a new file. If you do not provide an output file specification (see the description for /OUTPUT), the output is appended to SYS$OUTPUT.

/BEFORE=date-time
Lists current reservations that were executed prior to the specified time. The date value can be an absolute, delta, or combination time value, or one of the following keywords: TODAY, TOMORROW, or YESTERDAY. By default, CMS displays all existing reservations.

/CONCURRENT
Lists only concurrent reservations and concurrent replacements.

/GENERATION[=generation-expression]
Lists reservations for a specific generation or set of generations in a class.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/NOREPLACEMENTS
Indicates that concurrent replacements are not to be displayed.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)

Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, CLASS)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
CLASS (D)
NOCLASS
NONE—equivalent to (NOELEMENT, NOGROUP, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/OUTPUT[=file-specification]
/OUTPUT=SYS$OUTPUT (D)

Directs CMS to write output to the specified file, except for any warning and error messages, which are written to SYS$OUTPUT and SYS$ERROR. CMS creates a new file if you do not specify /APPEND. If you omit the /OUTPUT qualifier (or if you specify /OUTPUT but do not provide a file specification), CMS directs output to the default output device (SYS$OUTPUT). If you omit either the file name or the file-type component, CMS supplies the missing component from the default specification.

/SINCE=date-time

Lists current reservations that were executed after the specified time. The time value can be an absolute, delta, or combination time value, or one of the following keywords: TODAY, TOMORROW, or YESTERDAY.

/USER=username

Lists current reservations that were executed by the specified user. If you do not specify this qualifier, CMS lists reservations held by all users.

Example

```
CMS>  SHOW RESERVATIONS

Reservations in CMS Library DISKX:[PROJECT.CMSLIB]

AUDIT.FOR
    (2)    STEVEW     8      25-APR-1990 09:46:38 "add checks for file attributes"
  Concurrent replacements
           STEVEM     9      16-MAY-1990 16:01:30 "change %ascid & non-picrefs
  to pic"
GETCMD.FOR
    (1)    STEVEM     15     17-MAY-1990 21:28:34 "fix access violation"
INIT.FOR
    (1)    SMITH      13     15-MAY-1990 15:09:33 "context reference through add_field"
LSTUTL.FOR
    (1)    STEVEM     5      17-MAY-1990 21:43:10 "bug fixes"
```

This command lists the element name, the identification number of the reservation, the user who has it reserved, the generation number, and the date, time, and remark associated with the reservation for each element. In addition, CMS reports that the element AUDIT.FOR has been concurrently replaced, creating generation 9. As a result, user STEVEW must create a variant when he replaces the element.

SHOW REVIEWS_PENDING

Displays a list of element generations that currently have review pending status.

Format

SHOW REVIEWS_PENDING *[element-expression]*

Command Qualifiers	Defaults
/[NO]APPEND	/NOAPPEND
/BEFORE=date-time	See text
/GENERATION[=generation-expression]	See text
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/OUTPUT[=file-specification]	/OUTPUT=SYS$OUTPUT
/SINCE=date-time	See text
/USER=username	See text

Command Parameter

element-expression
Specifies one or more elements that are to be searched for generations with pending reviews. By default, all elements are searched. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas.

Description

The SHOW REVIEWS_PENDING command provides information about element generations currently marked for review. For each element with generations under review, the output consists of the element name, user, generation number, date, time, replacement remark, and any associated review comments that were established with the REVIEW GENERATION command. The elements are listed in alphabetical order.

Command Qualifiers

/APPEND
/NOAPPEND (D)
Controls whether CMS appends the command output to an existing file, or creates a new file. If you specify /APPEND and the output file does not exist, CMS creates a new file. If you do not provide an output file specification (see the description for /OUTPUT), the output is appended to SYS$OUTPUT.

/BEFORE=date-time
Specifies that only generations placed under review before the indicated date and time are to be displayed. By default, all generations with reviews pending are displayed. The date value can be an absolute, delta, or combination time value, or one of the following keywords: TODAY, TOMORROW, or YESTERDAY.

/GENERATION[=generation-expression]
Specifies that only reviews pending for generations matching the generation expression are to be displayed. By default, all generations with reviews pending are displayed.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, CLASS)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
CLASS (D)
NOCLASS
NONE—equivalent to (NOELEMENT, NOGROUP, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

/OUTPUT[=file-specification]
/OUTPUT=SYS$OUTPUT (D)
Directs CMS to write output to the specified file, except for any warning and error messages, which are written to SYS$OUTPUT and SYS$ERROR. CMS creates a new file if you do not specify /APPEND. If you omit the /OUTPUT qualifier (or if you specify /OUTPUT but do not provide a file specification), CMS directs output to the default output device (SYS$OUTPUT). If you omit either the file name or the file-type component, CMS supplies the missing component from the default specification.

/SINCE=date-time
Specifies that only generations placed under review after the indicated date and time are to be displayed. By default, all generations with reviews pending are displayed. The date value can be an absolute, delta, or combination time value, or one of the following keywords: TODAY, TOMORROW, or YESTERDAY.

/USER=username
Specifies that only reviews pending for generations created by the indicated user are to be displayed. By default, all generations with reviews pending are displayed.

Example

```
$  CMS SHOW REVIEWS_PENDING

Reviews pending in DEC/CMS Library DISKX:[WORK.CMSLIB]

EXAMPLE.SDML
    DAVIS     3      22-JAN-1990 10:40:26 "example is now complete"
    Review comments
       EVEDITOR      23-JAN-1990 14:41:54 "this looks wonderful!"
       JONES         25-JAN-1990 11:30:02 "agreed...let's submit it"
```

```
4777TIM_TST.BAS
    KENW      6      04-FEB-1990 09:22:36 "we need to verify this program"
```

This command shows all generations of elements in the library that are currently on the review pending list.

SHOW VERSION

Displays the version number of the CMS system currently in use.

Format

SHOW VERSION

Command Qualifiers	**Defaults**
/[NO]APPEND	/NOAPPEND
/OUTPUT[=file-specification]	/OUTPUT=SYS$OUTPUT

Description

The SHOW VERSION command shows the version number of the current CMS system.

Command Qualifiers

/APPEND
/NOAPPEND (D)
Controls whether CMS appends the command output to an existing file, or creates a new file. If you specify /APPEND and the output file does not exist, CMS creates a new file. If you do not provide an output file specification (see the description for /OUTPUT), the output is appended to SYS$OUTPUT.

/OUTPUT[=file-specification]
/OUTPUT=SYS$OUTPUT (D)
Directs CMS to write output to the specified file, except for any warning and error messages, which are written to SYS$OUTPUT and SYS$ERROR. CMS creates a new file if you do not specify /APPEND. If you omit the /OUTPUT qualifier (or if you specify /OUTPUT but do not provide a file specification), CMS directs output to the default output device (SYS$OUTPUT). If you omit either the file name or the file-type component, CMS supplies the missing component from the default specification.

SHOW VERSION

Example

```
CMS>  SHOW VERSION
DEC/CMS Version V3.3
```

This command shows the current version of CMS.

UNRESERVE

Cancels one or more reservations of a generation of an element.

Format

UNRESERVE *element-expression "remark"*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/NOCONFIRM
/[NO]DELETE[=file-specification]	/NODELETE
/GENERATION=generation-expression	See text
/IDENTIFICATION_NUMBER=n	See text
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL

Command Parameters

element-expression
Specifies one or more elements whose reservations are to be canceled. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas.

"remark"
Specifies a character string to be logged in the history file with this command, usually used to explain why the command was entered. The remark is enclosed in quotation marks. If no remark was entered, a null remark ("") is logged.

UNRESERVE

Description

The UNRESERVE command cancels an existing reservation of a generation of an element. You cannot unreserve a generation held by another user unless you hold BYPASS privilege or unless you are granted BYPASS access to the element by an access control entry (see Section 7.3).

If you have more than one reservation of an element, or if you are replacing a concurrent reservation made by another user (that is, if there is any ambiguity), you must specify the exact reservation to be canceled (see Section 4.3.3). You do this by using either the /GENERATION qualifier or the /IDENTIFICATION_NUMBER qualifier.

You can use /GENERATION as long as the concurrent reservations are not on the same generation. If you have more than one concurrent reservation for the same generation, you must identify the specific reservation to be canceled. Each reservation is assigned an identification number. Use the SHOW RESERVATIONS command to determine the identification number of each reservation. The identification number appears in parentheses at the beginning of each line. If you use the /IDENTIFICATION_NUMBER qualifier, you do not need to also use the /GENERATION qualifier; when both are used, CMS ignores the /GENERATION qualifier.

Command Qualifiers

/CONFIRM
/NOCONFIRM (D)
Controls whether CMS prompts you for confirmation before each transaction.

When you specify /CONFIRM and run CMS in interactive mode, CMS prompts you for confirmation. If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

The /NOCONFIRM qualifier does not override the confirmation prompt issued when you unreserve a concurrent reservation or when you cancel another user's reservation.

/DELETE[=file-specification]
/NODELETE (D)
Controls whether all versions of the unreserved file are deleted. If you omit /DELETE, the files are not deleted. If you specify /DELETE without a file specification, all versions of the unreserved files are deleted from your current directory. The file specification allows you to specify a different location and the name of the file to be deleted.

/GENERATION=generation-expression
Specifies which reserved generation of the element is to be unreserved. If you have more than one reservation of the same element generation, you must use the /IDENTIFICATION_NUMBER qualifier to unreserve the reservation.

You can specify a generation indirectly by using a class name, the plus operator, the semicolon, or relative generation offsets. See Section 10.2.5.

/IDENTIFICATION_NUMBER=n
Specifies which reservation is to be unreserved. This qualifier is required when you have multiple reservations of the same generation of an element. /IDENTIFICATION_NUMBER can be used instead of /GENERATION when you have multiple reservations. Use the SHOW RESERVATIONS command to determine the identification number of each reservation. The identification number appears before the user name.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, CLASS)
ELEMENT (D)

NOELEMENT
GROUP (D)
NOGROUP
CLASS (D)
NOCLASS
NONE—equivalent to (NOELEMENT, NOGROUP, NOCLASS)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]CLASS keywords.

CMS automatically performs occlusion for all objects; that is, CMS selects only the first occurrence of a specified object.

Examples

1.
```
CMS> UNRESERVE FILEIO.BLI
_Remark:  problem elsewhere
%CMS-S-UNRESERVED, element DISKX:[PROJECT.CMSLIB]FILEIO.BLI unreserved
```

This command cancels the existing reservation of the element FILEIO.BLI.

2.
```
$ SET PROCESS/PRIVILEGE=BYPASS
$ CMS UNRESERVE SAMPLE.TXT "unreserving on behalf of SMITH"
Element A.TXT currently reserved by:
    (1)    SMITH     1      17-FEB-1990 13:43:52 "sample module for program"
Unreserve (1) SAMPLE.TXT generation 1, held by SMITH? [Y/N] (N): Y
%CMS-S-UNRESERVED, element DISKX:[SMITH.WORK.CMSLIB]SAMPLE.TXT unreserved
```

This example shows the cancellation of another user's existing reservation (through the use of BYPASS privilege) of the latest generation of the element SAMPLE.TXT. If concurrent reservations of element SAMPLE.TXT existed, you would need to specify the exact reservation to be canceled by using either the /GENERATION or /IDENTIFICATION qualifier.

VERIFY

Performs a series of checks on your CMS library to confirm that the library structure and library files are in a valid form.

Format

VERIFY *[element-expression]*

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/CONFIRM
/[NO]LOG	/LOG
/OCCLUDE[=option,...]	/OCCLUDE=ALL
/[NO]RECOVER	/NORECOVER
/[NO]REPAIR	/NOREPAIR

Restrictions

- You cannot specify the /RECOVER and /REPAIR qualifiers on the same command line.
- You cannot specify an element expression parameter when you use the /RECOVER qualifier.

Command Parameter

element-expression
Specifies one or more elements to be verified. If you do not supply an element expression, CMS verifies every file in the library. You cannot specify an element expression parameter if you use the /RECOVER qualifier. An element expression can be an element name, a group name, a wildcard expression, or a list of these separated by commas.

VERIFY

Description

The VERIFY command performs a series of consistency checks on your library. If you issue VERIFY under normal conditions, the command executes successfully, indicating that the information in your library is correct. However, if the data in the library is invalid, the command returns an error message indicating that there is an error in the verification of the library. You can use the /RECOVER and /REPAIR qualifiers on the VERIFY command to correct some of the errors discovered by VERIFY.

Recovery and repair transactions are marked as unusual occurrences in the library history. For more information about the VERIFY transaction, see Section 9.2.

Command Qualifiers

/CONFIRM (D)
/NOCONFIRM
Controls whether CMS prompts you for confirmation prior to deleting any invalid reference copies during a VERIFY/REPAIR operation. In some cases, however, if CMS finds invalid reference copies (for example, if there is one valid reference copy and the remaining reference copies are invalid), it automatically deletes the invalid copies without prompting for confirmation.

If you type YES, ALL, TRUE, or 1, CMS executes the transaction. If you type NO, QUIT, FALSE, 0, or press RETURN or CTRL/Z, no action is performed. If you type any other character, CMS continues to prompt until you type an acceptable response.

CMS does not prompt for confirmation in batch mode.

/LOG (D)
/NOLOG
Controls whether CMS displays success and informational messages on the default output device. If the command executes successfully, CMS displays a success message. If you specify /NOLOG, success and informational messages are suppressed. Any warning, error, or fatal error messages are displayed regardless of whether /LOG or /NOLOG is specified.

/OCCLUDE[=option,...]
/OCCLUDE=ALL (D)
Controls whether CMS selects the first instance of the specified object, or all instances of the specified object in the library search list. The options field contains one or more keywords associated with the name of the object. The options field can consist of the following keywords:

ALL—equivalent to (ELEMENT, GROUP, OTHER)
ELEMENT (D)
NOELEMENT
GROUP (D)
NOGROUP
OTHER (D)
NOOTHER
NONE—equivalent to (NOELEMENT, NOGROUP, NOOTHER)

You can specify either ALL or NONE, or any combination of the [NO]ELEMENT, [NO]GROUP, and [NO]OTHER keywords.

If you do not specify an element expression on the VERIFY command, the default is /OCCLUDE=NONE. If you do specify an element expression, the default is /OCCLUDE=ALL.

/RECOVER
/NORECOVER (D)
Controls whether the VERIFY command cancels an incomplete transaction. You use the /RECOVER qualifier when a transaction with the library is incomplete and the rollback mechanism does not automatically cancel the command (see Chapter 9). For example, you must use the /RECOVER qualifier if the VMS system fails while a CMS command is updating the library.

If a CMS command is terminated and the library is left in an inconsistent state, CMS recognizes that the last transaction was incomplete and automatically initiates command rollback (see Section 9.1) to return the library to a valid format.

If you have set up a restrictive protection scheme and a system failure occurs during a CMS transaction and leaves your library in an inconsistent state, the VERIFY/RECOVER command should be executed by the same person who was using CMS at the time of the system failure or by a person with sufficient privileges.

/REPAIR
/NOREPAIR (D)
Controls whether or not the VERIFY command repairs a file or files in the CMS library. You should use the /REPAIR qualifier if VERIFY issues a message concerning one of the following conditions:

- Element data files in the library were not closed by CMS.
- The checksum of elements in the library is invalid.
- Generations in the library have an invalid maximum record size.
- A data block was not found on pass 1.
- The reference copy for an element is missing.
- A reference copy is found for an element with the /NOREFERENCE_COPY qualifier.
- There are duplicate reference copies for an element.
- The reference copy of an element is invalid.

If reference copies need repairing, VERIFY/REPAIR creates or deletes files as necessary to correct the information in the reference copy directory.

If a file was not closed by CMS, VERIFY/REPAIR repairs the VAX Record Management Services (RMS) file header so that the file can be successfully verified. If the checksum of a file does not correspond to the contents of the file, VERIFY/REPAIR recalculates the checksum so that the library can be verified.

If any of these conditions exist, data may have been changed in the library by methods other than the normal updating of the library with CMS commands. For example, a file may have been opened and modified with a text editor. You may want to find out why the files could not be verified. See Chapter 9 for more information.

Example

```
CMS> VERIFY
%CMS-I-VERCLS, class list verified
%CMS-I-VERCMD, command list verified
%CMS-I-VERELE, element list verified
%CMS-I-VERGRP, group list verified
%CMS-I-VERRES, reservation list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERFRE, internal free space list verified
%CMS-I-VERARC, archive control block verified
%CMS-I-VER2, internal contiguous space verified
%CMS-I-VERCON, control file verified
%CMS-I-VEREDF, element DISKX:[PROJECT.CMSLIB]INIT.FOR verified
%CMS-I-VEREDF, element DISKX:[PROJECT.CMSLIB]MSGDOC.FOR verified
%CMS-I-VEREDF, element DISKX:[PROJECT.CMSLIB]OUTPUT.FOR verified
%CMS-I-VEREDFS, element data files verified
%CMS-I-VERIFIED, library DISKX:[PROJECT.CMSLIB] verified
```

This example indicates that the library is in a consistent format. If VERIFY reports errors, see the description of the error message in Appendix B.

Appendix A

Summary of CMS Interface Functional Mappings

This table displays how each of the CMS interfaces are functionally mapped into each other.

DCL Command	Callable Routine	DECwindows Menu Item
ACCEPT GENERATION	CMS$REVIEW_GENERATION	Maintenance.Review.Accept
ANNOTATE	CMS$ANNOTATE	Data.Annotate
CANCEL REVIEW	CMS$REVIEW_GENERATION	Maintenance.Review.Cancel
CONVERT LIBRARY	N/A	N/A
COPY ELEMENT	CMS$COPY_ELEMENT	Maintenance.Copy.Element
CREATE CLASS	CMS$CREATE_CLASS	Maintenance.Create.Class
CREATE ELEMENT	CMS$CREATE_ELEMENT	Data.Create.Element
CREATE GROUP	CMS$CREATE_GROUP	Maintenance.Create.Group
CREATE LIBRARY	CMS$CREATE_LIBRARY	Library.Create
DELETE CLASS	CMS$DELETE_CLASS	Maintenance.Delete.Class
DELETE ELEMENT	CMS$DELETE_ELEMENT	Maintenance.Delete.Element
DELETE GENERATION	CMS$DELETE_GENERATION	Maintenance.Delete.Generation
DELETE GROUP	CMS$DELETE_GROUP	Maintenance.Delete.Group
DELETE HISTORY	CMS$DELETE_HISTORY	Maintenance.Delete.History
DIFFERENCES	CMS$DIFFERENCES	Data.Differences
FETCH	CMS$FETCH	Data.Fetch
INSERT ELEMENT	CMS$INSERT_ELEMENT	Maintenance.Insert.Element
INSERT GENERATION	CMS$INSERT_GENERATION	Maintenance.Insert.Generation
INSERT GROUP	CMS$INSERT_GROUP	Maintenance.Insert.Group

DCL Command	Callable Routine	DECwindows Menu Item
MARK GENERATION	CMS$REVIEW_GENERATION	Maintenance.Review.Mark
MODIFY CLASS	CMS$MODIFY_CLASS	Maintenance.Modify.Class
MODIFY ELEMENT	CMS$MODIFY_ELEMENT	Maintenance.Modify.Element
MODIFY GENERATION	CMS$MODIFY_GENERATION	Maintenance.Modify.Generation
MODIFY GROUP	CMS$MODIFY_GROUP	Maintenance.Modify.Group
MODIFY LIBRARY	CMS$MODIFY_LIBRARY	Maintenance.Modify.Library
REJECT GENERATION	CMS$REVIEW_GENERATION	Maintenance.Review.Reject
REMARK	CMS$REMARK	Maintenance.Remark
REMOVE ELEMENT	CMS$REMOVE_ELEMENT	Maintenance.Remove.Element
REMOVE GENERATION	CMS$REMOVE_GENERATION	Maintenance.Remove.Generatior
REMOVE GROUP	CMS$REMOVE_GROUP	Maintenance.Remove.Group
REPLACE	CMS$REPLACE	Data.Replace
RESERVE	CMS$FETCH	Data.Reserve
RETRIEVE ARCHIVE	CMS$RETRIEVE_ARCHIVE	N/A
REVIEW GENERATION	CMS$REVIEW_GENERATION	Maintenance.Review.Comment
SET ACL	CMS$SET_ACL	Maintenance.Set.ACL
SET LIBRARY	CMS$SET_LIBRARY	Library.Open
SET NOLIBRARY	CMS$SET_NOLIBRARY	Library.Close
SHOW ACL	CMS$SHOW_ACL	View.Expand.ACLs
SHOW ARCHIVE	CMS$SHOW_ARCHIVE	N/A
SHOW CLASS	CMS$SHOW_CLASS	View.Class
SHOW ELEMENT	CMS$SHOW_ELEMENT	View.Element
SHOW GENERATION	CMS$SHOW_GENERATION	View.Expand.Children
SHOW GROUP	CMS$SHOW_GROUP	View.Group
SHOW HISTORY	CMS$SHOW_HISTORY	View.History
SHOW LIBRARY	CMS$SHOW_LIBRARY	N/A
SHOW RESERVATIONS	CMS$SHOW_RESERVATIONS	View.Reservation
SHOW REVIEWS_PENDING	CMS$SHOW_REVIEWS_PENDING	View.Review
SHOW VERSION	CMS$SHOW_VERSION	Help.About
UNRESERVE	CMS$UNRESERVE	Data.Unreserve
VERIFY	CMS$VERIFY	Library.Verify

Appendix B
Error Messages

This appendix lists the diagnostic messages produced by CMS. The messages are accompanied by explanations and suggested actions to recover from errors.

B.1 Message Display

Messages are displayed on the current device identified by the logical name SYS$OUTPUT. For an interactive user, this device is a terminal; for batch job users, it is the batch job log file. If the logical device SYS$ERROR is different from SYS$OUTPUT, the system writes warning, error, and fatal error messages to that device as well.

B.1.1 Severity Levels

The severity level of a message is included in the status message. Success and informational messages inform you that CMS has performed your request.

Warning messages indicate that the command may have performed some, but not all, of your request, and that you may need to verify command or program output.

Error messages indicate that CMS is unable to perform the requested function, and that you must correct the problem and enter the command again.

Fatal messages indicate that CMS has discovered inconsistencies in itself or the CMS library, and that normal processing cannot continue.

B.1.2 Library Directory Specifications

CMS displays the library directory specification for each status message containing a CMS object. These include:

- Elements
- Groups
- Classes
- The library (and its object subtypes)

This allows you to determine the exact library to which an object belongs when multiple libraries are involved in the operation. For example:

```
$ CMS SET LIBRARY [WORK.CMSLIB],[PROJ.CMSLIB]
.
.
.
$ CMS RESERVE EXAMPLE.TXT/OCCLUDE=NOELEMENT ""
%CMS-S-RESERVED, generation 1 of element DISKX:[WORK.CMSLIB]EXAMPLE.TXT reserved
%CMS-S-RESERVED, generation 1 of element DISKX:[PROJ.CMSLIB]EXAMPLE.TXT reserved
```

B.1.3 Secondary Messages

Often, CMS displays more than one message as a result of the execution of a command. When this occurs, CMS displays the primary message that indicates the status return, followed by one or more secondary messages that provide specific information. For example:

```
$ CMS FETCH SRC.C ""
%CMS-E-NOFETCH, error fetching element SRC.C
-CMS-E-NOTFOUND, element SRC.C not found
```

Primary messages contain the percent sign (%) prefix; secondary messages contain the hyphen (-) prefix.

B.2 CMS Messages

ABSTIM, 'qualifier' time value must be absolute

Error: You must use an absolute time value for the /BEFORE and /SINCE qualifiers.

User Action: Correct the time value and enter the command again.

ACCEPTANCES, 'count' generation(s) accepted

Success: CMS has successfully marked the indicated number of generations as accepted and removed them from the review pending list.

User Action: None.

ACCEPTED, generation 'gen-number' of element 'name' accepted

Success: CMS has successfully marked the indicated generation of the specified element as accepted and removed it from the review pending list.

User Action: None.

ACCVIORD, access violation reading routine argument at virtual address 'address'
ACCVIOWT, access violation referencing routine argument at virtual address 'address'

Fatal: You have passed an unreadable or unwritable argument to a CMS routine.

User Action: Check the number of arguments in the call to CMS. Also check to make sure you are using the correct passing mechanism, and that you are passing the arguments in the correct order.

ALPHACHAR, in a class name the first character must be alphabetic

Error: You have specified a class name that does not begin with a letter.

User Action: Correct the class name and enter the command again.

ALRDYEXISTS, 'name' is already a class name
ALRDYEXISTS, 'name' is already an element name
ALRDYEXISTS, 'name' is already a group name
ALRDYEXISTS, 'name' is already a reference copy name

Error: You have specified a group, element, or class name that is already being used in the library, or a reference copy name that already exists in the reference copy directory. All these names must be unique.

User Action: Use a different name and enter the command again.

ALRDYINCLS, class 'name' already contains a generation from element 'name'

Error: You have tried to insert an element generation into a class that already contains a generation from that element. Classes can contain only one generation from a particular element.

User Action: If you want to insert the new generation into the class, you must first remove the old generation from the class. (See the description of the REMOVE GENERATION command or the description of the /ALWAYS qualifier for the INSERT GENERATION command.)

ALRDYINGRP, element is already in group
ALRDYINGRP, group is already in group

Error: The specified element or group already belongs to the specified containing group.

User Action: None.

ALRDYMARKED, generation 'gen-number' of 'name' already marked for review

Error: You have tried to mark a generation for review that is already marked.

User Action: None.

ANNOTATED, element 'name' annotated

Success: CMS has successfully created an annotated listing file of the indicated element.

User Action: None.

ANNOTATIONS, 'count' annotation(s) completed

Success: CMS has successfully created annotated listings for the indicated number of elements.

User Action: None.

ARGCONFLICT, argument conflict (routine and file both specified)

Error: There are several CMS routines that do not accept both a routine and a file for input or for output.

User Action: Check the arguments that you have specified in your calls to CMS.

ARGCOUNTERR, incorrect number of arguments

Error: You have passed an incorrect number of arguments to the CMS$GET_STRING routine. The CMS$GET_STRING routine expects two arguments, one for the string identifier and a corresponding string to be filled in by CMS.

User Action: Check the arguments you have specified in the routine call.

AUTOREC, attempting automatic VERIFY/RECOVER

Informational: CMS has detected that recovery is necessary. An automatic VERIFY/RECOVER transaction will be attempted. If you do not have the required access privileges, automatic recovery will fail.

User Action: None.

AUTORECSUC, automatic VERIFY/RECOVER was successful

Informational: CMS has successfully recovered your library with an automatic VERIFY/RECOVER operation.

User Action: None.

BADBUG, there is an unrecoverable bug in CMS or something it calls

Fatal: A rollback attempt to recover has failed.

User Action: This is not a situation that a user can resolve. If you have a Self-Maintenance Software Agreement, submit a Software Performance Report.

BADCALL, invalid call to routine 'name'

Fatal: CMS has detected something wrong with one or more of the arguments passed to a CMS routine. This message can be followed by CMS$_INVFETDB, CMS$_INVLENGTH, CMS$_INVLIBDB, CMS$_INVRDARG, CMS$_INVSTRDES, CMS$_INVWRTARG, CMS$_MAXARG, CMS$_MINARG, or other error messages.

User Action: See the descriptions of the secondary messages for more information. Do not use SS$_CONTINUE to continue processing when CMS returns CMS$_BADCALL.

BADCRC, incorrect checksum in element 'name'

Warning: An element in your library contains an incorrect checksum.

User Action: Use the VERIFY/REPAIR command to recalculate the checksum.

BADFORMAT, invalid format combination

Error: CMS has detected an invalid format argument passed to a CMS routine. Only one data-format type can be specified, and only one data-partition type can be specified. The combination of data format and data partition must be valid.

User Action: Correct the mistake in the format information passed.

BADLIB, there is something wrong with your library

Fatal: CMS has discovered an inconsistency or error in your CMS library.

User Action: A BADLIB message indicates an inconsistency in your library that CMS cannot fix. You must use a backup copy of your library to be sure that you are using a consistent database.

BADCRETIME, file 'name' was created after the last successful transaction
BADLENSTR, 'structure' block length is 'count'; it should be 'count'
BADLSTSTR, 'object' list does not hold this 'object'
BADORDSTR, 'object' block 'name' is out of order

Error: CMS discovered an error while verifying your library.

User Action: See the descriptions of the primary messages for more information.

BADPTR, cannot finish reading 'object' list; bad pointer 'address'
BADTYPSTR, 'object' block type is 'number' ; it should be 'number'
BADVERSTR, 'object' block version is 'number' ; it should be 'number'
BCKPTRSTR, 'object' back pointer is 'address' ; previous block is 'address'

Error: CMS discovered an error while verifying your library.

User Action: See the descriptions of the primary messages for more information.

BADREF, reference copy for element 'name' is bad

Error: The reference copy for the indicated element is incorrect.

User Action: Use VERIFY/REPAIR to re-create the reference copy for the element.

BADVERSION, file 'name' has a bad version

Error: There is a duplicate element data file in the library, and CMS is unable to determine which of the two files is the valid element.

User Action: Examine the contents of the element data files to see if one of them is incorrect (perhaps it has no control records, or it is an incomplete file). Delete the incorrect file and enter the command again.

BUG, there is something wrong with CMS or something it calls

Fatal: CMS has discovered a bug or inconsistency in itself.

User Action: This is not a situation that a user can resolve. If you have a Self-Maintenance Software Agreement, submit a Software Performance Report.

CANCELATIONS, 'count' reviews canceled

Success: The indicated number of generations marked for review has been successfully removed from the review pending list.

User Action: None.

CANCELED, review of generation 'gen-number' of element 'name' canceled

Success: The indicated generation of the specified element marked for review has been successfully removed from the review pending list.

User Action: None.

CNTSTR, 'object' count is 'count'; it should be 'count'

Error: This is a secondary message to BADLIB.

User Action: This message indicates a situation that CMS cannot fix. You must use a backup copy of your library. See Chapter 9 for more information.

COMPARED, 'file-spec' and 'file-spec' compared

Success: CMS has successfully compared the indicated files.

User Action: None.

CONFLICTS, 'count' merge conflict(s)

Warning: CMS has detected one or more conflicts during the merge transaction. The merge conflicts are flagged with asterisks in the output file.

User Action: Edit the output file to resolve the conflicts and delete the conflict asterisks. See Chapter 6 for more information.

CONTROLC, operation aborted by CTRL/C

Warning: You pressed CTRL/C to abort an operation in progress.

User Action: Reenter the command to start the operation from the beginning.

CONVERTED, Version 2 library converted to Version 3 format

Success: CMS has successfully converted a Version 2 library to Version 3.0 format. Once you have converted the library, you can use CMS Version 3.0 to manipulate it.

User Action: None.

CONVERTLIB, 'directory' is a Version 'number' library and cannot be used without conversion

Error: You have tried to set your library to a directory that may contain library control structures created by a previous version of CMS. Before you can use the library, you must convert it.

User Action: Use the CONVERT LIBRARY command to convert a Version 2 library to Version 3. If the library is a Version 1 library, you should try to use the CMSCONVERTLIB program that was installed on your system with Version 2; however, conversion of Version 1 libraries is no longer supported.

CONVNOTNEC, library 'name' is already in the current format

Error: You have attempted to convert a library that is already in the current format. No conversion is necessary.

User Action: None.

COPIED, element 'name' copied to 'name'

Success: CMS has successfully copied the indicated element.

User Action: None.

COPIES, 'count' copies completed

Success: CMS has successfully copied the indicated number of elements.

User Action: None.

CREATED, library 'directory-spec' created
CREATED, class 'name' created
CREATED, element 'name' created
CREATED, group 'name' created

Success: CMS has successfully created the indicated library, class, element, or group.

User Action: None.

CREATED, element 'name' created, generation 1 reserved

Success: CMS has successfully created the indicated element and reserved the first generation.

User Action: None.

CREATES, 'count' creations completed

Success: CMS has successfully created one or more classes, elements, or groups.

User Action: None.

DEFAULTDIR, default directory cannot be a CMS library

Error: You cannot set your CMS library to your default directory. Also, you cannot specify your default directory as a parameter to the CREATE LIBRARY command, or set default to a CMS library when attempting to enter a CMS command.

User Action: Change your default directory so that it is different from your CMS library and enter the command again.

DELETED, class 'name' deleted
DELETED, element 'name' deleted
DELETED, group 'name' deleted

Success: CMS has successfully deleted the indicated class, element, or group.

User Action: None.

DELETIONS, 'count' deletion(s) completed

Success: CMS has successfully deleted the indicated number of classes, elements, or groups.

User Action: None.

DIFFERENT, files are different

Informational: CMS has detected differences between the compared files.

User Action: None.

DUPEDF, 'name' is a duplicate element data file

Error: VERIFY has detected a duplicate element data file.

User Action: Use VERIFY/REPAIR to remove the extra file.

DUPREF, 'name' is a duplicate reference copy

Error: VERIFY has detected a duplicate reference copy.

User Action: Use VERIFY/REPAIR to remove the extra file.

EDFMISS, element data file 'name' is missing

Error: An element name listed in the control file does not have a corresponding file in the library.

User Action: Use a backup tape to obtain the most recent copy of the element file.

ELEEXISTS, 'name' is already an element name

Error: You have specified an element name that is already being used in the library. Element names must be unique.

User Action: Use a different name and enter the command again.

ELEMULTRES, element 'name' is reserved more than once by you

Error: CMS cannot determine which reservation you want to replace or unreserve.

User Action: Remove the ambiguity by using either the /IDENTIFICATION_NUMBER or the /GENERATION qualifier. If you have multiple reservations on the same generation, you must use /IDENTIFICATION_NUMBER. To determine the correct identification number, use SHOW RESERVATIONS.

ELEXPIGN, element expression ignored

Warning: You have supplied an element expression for a VERIFY/RECOVER transaction.

User Action: Reenter the command without the element expression.

ENDPTRSTR, 'object' end pointer is 'address' ; last block is 'address'

Error: This is secondary to BADLIB; it indicates an inconsistency in the data base that CMS cannot fix.

User Action: Use a backup copy of the library. See Chapter 9 for more information.

EOF, end of file

Warning: The end of file was encountered. This message is used for communication between CMS and user-supplied callback routines.

User Action: None.

ERRACCEPTANCES, 'count' generation(s) accepted and 'count' error(s) occurred

Error: CMS has successfully marked the indicated number of generations for acceptance and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRANNOTATIONS, 'count' element(s) annotated and 'count' error(s) occurred

Error: CMS has successfully annotated the indicated number of elements and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRCANCELATIONS, 'count' review(s) canceled and 'count' error(s) occurred

Error: CMS has successfully removed the indicated number of the generations marked for review from the review pending list and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRCLOSE, error closing 'filename'

Error: CMS was unable to close the indicated file.

User Action: See the descriptions of the secondary messages for more information. If the problem involves any library files, use VERIFY.

ERRCOPIES, 'count' element(s) copied and 'count' error(s) occurred

Error: CMS has successfully copied the indicated number of elements and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRCREATES, 'count' creations completed and 'count' error(s) occurred

Error: CMS has successfully completed the indicated number of create transactions and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRDELETIONS, 'count' deletion(s) completed and 'count' error(s) occurred

Error: CMS has successfully deleted the indicated number of elements, groups, or classes and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRELEHIS, error in element history data

Error: CMS encountered erroneous data in the element history. The element history is included in the output file when you fetch or reserve an element that has the **history** attribute defined.

User Action: Examine the file in your default directory (do not edit the library element file) to see if there is any non-history data in the element history.

ERREMOVALS, 'count' removal(s) completed and 'count' error(s) occurred

Error: CMS has successfully removed the indicated number of elements or groups and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERREPLACEMENTS, 'count' element(s) replaced and 'count' error(s) occurred

Error: CMS has successfully replaced the indicated number of elements and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRESERVATIONS, 'count' element(s) reserved and 'count' error(s) occurred

Error: CMS has successfully reserved the indicated number of elements and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRETRIEVALS, 'count' generation(s) retrieved and 'count' error(s) occurred

Error: CMS has successfully retrieved the indicated number of generations and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRFETCHES, 'count' element(s) fetched and 'count' error(s) occurred

Error: CMS has successfully fetched the indicated number of elements and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRGENDELETIONS, 'count' element generations deleted and 'count' error(s) occurred

Error: CMS has successfully deleted the indicated number of element generation ranges, and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRHISLINE, missing beginning (ending) history control line in file 'filename'

Error: CMS cannot find the top or bottom control line of the files history data.

User Action: Edit the file and restore the missing history control line. The beginning and ending control lines should be both identical and the same as those surrounding the original replacement history text.

ERRINSERTIONS, 'count' insertion(s) completed and 'count' error(s) occurred

Error: CMS has successfully inserted the indicated number of elements, groups, or generations, and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRMARKS, 'count' generation(s) marked and 'count' error(s) occurred

Error: CMS has successfully marked the indicated number of generations for review and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRMODACLS, 'count' access control list(s) modified and 'count' error(s) occurred

Error: CMS has successfully modified the indicated number of access control lists and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRMODIFIES, 'count' modification(s) completed and 'count' error(s) occurred

Error: CMS has successfully completed the indicated number of modifications and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRPAREXP, error parsing 'type' expression

Error: You specified a class, element, or group with illegal syntax.

User Action: Correct the expression and enter the command again. See the descriptions of the secondary messages for more information.

ERRREJECTIONS, 'count' generation(s) rejected and 'count' error(s) occurred

Error: CMS has successfully rejected the indicated number of element generations and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRREVIEWS, 'count' generation(s) reviewed with 'count' error(s)

Error: CMS has successfully associated a review comment with the indicated number of generations and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRUNRESERVES, 'count' element(s) unreserved with 'count' error(s)

Error: CMS has successfully canceled the reservations for the indicated number of elements and encountered one or more errors during the transaction.

User Action: See the descriptions of the secondary messages for more information.

ERRVER2, internal contiguous space verified with errors
ERRVERARC, archive control block verified with errors
ERRVERCLS, class list verified with errors
ERRVERCMD, command list verified with errors

Error: CMS encountered one or more errors while verifying your library.

User Action: Use VERIFY/REPAIR. If it is unable to correct the problem, use a backup copy of your library. See Chapter 9 for more information.

ERRVERCON, control file verified with errors
ERRVEREDFS, element data files verified with errors
ERRVERELE, element list verified with errors
ERRVERFRE, internal free space list verified with errors

Error: CMS encountered one or more errors while verifying your library.

User Action: Use VERIFY/REPAIR. If it is unable to correct the problem, restore a backup copy of your library. See Chapter 9 for more information.

ERRVERGRP, group list verified with errors
ERRVERREFS, reference copies verified with errors
ERRVERRES, reservation list verified with errors
ERRVERSTR, internal string list structure verified with errors

Error: CMS encountered one or more errors while verifying your library.

User Action: Use VERIFY/REPAIR. If it is unable to correct the problem, restore a backup copy of your library. See Chapter 9 for more information.

EXCLUDE, object excluded from consideration

Success: Return this value from a callback routine to indicate that the current piece of data is to be excluded from the action of the CMS routine. (See the *VAX DEC/Code Management System Callable Routines Reference Manual* for more information about the CMS$_EXCLUDE return code.)

User Action: None.

EXIT, EXIT can be used only to leave subsystem level

Error: You entered CMS EXIT from DCL command level.

User Action: None.

FETCHED, generation 'gen-number' of element 'name' fetched
FETCHED, generation 'gen-number' of element 'name' fetched and merged with 'gen-number'

Success: CMS has successfully fetched the specified element generation from the library. CMS also returns this message when you execute a FETCH/MERGE transaction.

User Action: None.

FETCHES, 'count' element(s) fetched

Success: CMS has successfully fetched 'count' elements.

User Action: None.

FILEXISTS, file already exists, 'file-spec' created

Informational: There is already a version of the indicated file in the directory that CMS is using as output for a command.

User Action: None.

FILINUSE, file 'filename' in use

Informational: CMS is attempting to open a file that is currently locked by another user. Under these circumstances, CMS will wait and then try again.

User Action: None.

FIXCRC, checksum of element 'name' repaired

Informational: CMS has repaired the checksum of the indicated element.

User Action: None.

FIXHDR, file header of element data file 'name' repaired

Informational: CMS has repaired the file header of the indicated element.

User Action: None.

GENCREATED, generation 'gen-number' of element 'name' created
GENCREATED, generation 'gen-number' of element 'name' created and reserved

Success: CMS has successfully created the indicated element generation in the library.

User Action: None.

GENDELETED, 'count' generation(s) of element 'name' deleted

Success: CMS has successfully deleted the indicated number of generations of the specified element.

User Action: None.

GENDELETIONS, 'count' element generations deleted

Success: CMS has successfully deleted generations of the indicated number of elements.

User Action: None.

GENEXISTS, generation 'gen-number' already exists

Error: You have tried to execute a replacement transaction that would create a generation that already exists in the library.

User Action: To save the changes you have made to the file, you must create a variant generation.

GENINSERTED, generation 'gen-number' of element 'name' inserted into class 'name'

Success: CMS has successfully inserted the indicated element generation into the specified class.

User Action: None.

GENMULTRES, generation 'gen-exp' of element 'name' is reserved more than once by you

Error: CMS cannot determine which reservation you want to replace or unreserve.

User Action: Remove the ambiguity by using the /IDENTIFICATION_NUMBER qualifier. To determine the correct identification number, use SHOW RESERVATIONS.

GENNOINSERT, error inserting 'name' into class 'name'

Error: CMS cannot insert the element generation into the class.

User Action: See the descriptions of the secondary messages for more information.

GENNOREMOVE, error removing 'name' from class 'name'

Error: CMS cannot remove the element generation from the class.

User Action: See the descriptions of the secondary messages for more information.

GENNOTANC, generation 'gen-number' is not an ancestor of 'gen-number'

Error: You have specified a generation that is either a descendant of, or not on the same line of descent as, another generation.

User Action: Correct the generation number and enter the command again.

GENNOTFOUND, generation 'gen-number' of 'name' not found

Error: CMS was unable to find the indicated element generation in the library.

User Action: Check to see that you are requesting a valid element generation.

GENNOTRES, generation 'gen-exp' is not reserved by you

Error: You have specified a generation number that does not correspond to a reservation belonging to you.

User Action: Check your reservation generation and enter the command again.

GENRECSIZE, generation 'name' has 'size'-byte records; 'name' has 'size'-byte records

Error: CMS cannot merge generations with fixed-length records if the records are not of the same length.

User Action: Reserve the generation with the smaller records and convert those records to the larger size. Then replace the generation and attempt the merge again, this time using the newly created generation.

GENREMOVED, generation 'gen-number' of element 'name' removed from class 'name'

Success: CMS has successfully removed the indicated element generation from the class that you specified.

User Action: None.

GENRESREV, generation reserved or marked for review

Error: A deletion range cannot contain generations that are reserved or marked for review.

User Action: If the generation is reserved, either replace or unreserve it. If the generation is marked for review, accept or reject it or cancel the review. Then enter the command again.

GENTOODEEP, generation edits are nested too deeply

Error: The generation edits contained in the element data file are nested more deeply than the maximum allowed (400). This is a fairly rare occurrence caused by the method CMS uses to store generation edits, and is outside of the user's control.

User Action: Use the DELETE GENERATION command to delete old or unneeded generations, then enter the command again. Alternatively, you can use the reserved generation to create a new element. The generation contained in the old element can then be unreserved.

HASFILES, directory 'directory-spec' contains files

Informational: You have tried to create a CMS library in a directory that already contains files.

User Action: You must specify an empty directory or subdirectory as the parameter to the CREATE LIBRARY command.

HASMEMBERS, class 'name' contains one or more generations
HASMEMBERS, group 'name' contains one or more elements

Error: The class or group you specified contains one or more elements. A class or a group must be empty before you can delete it.

User Action: Remove any elements that belong to the group or class (see the description for REMOVE ELEMENT, REMOVE GENERATION, or REMOVE GROUP), then enter the command again.

HISNOTSTM, history file record format is not stream_lf

Error: The 00CMS.HIS file must be in stream_lf (line feed) format. Note that if you edit the history file, the format will no longer be stream_lf.

User Action: Use VERIFY/REPAIR.

HISTDEL, 'number' history records deleted

Success: CMS has successfully deleted part or all of your library history. By default, the deleted history is placed in the file HISTORY.DMP in your default directory.

User Action: None.

IDENTICAL, files are identical

Success: The two files compared by CMS are identical.

User Action: None.

IDENTNOTRES, reservation 'integer' is not reserved by you

Error: You have specified a reservation identification number that does not correspond to a reservation belonging to you.

User Action:

ILLACT, illegal review action

Error: This is a return status from the callable routine. You have specified a value for the review action that does not correspond to any known actions.

User Action: See the description of CMS$REVIEW_GENERATION in the *VAX DEC/Code Management System Callable Routines Reference Manual* and correct the action value.

ILLARCREC, illegal record in archive file

Error: The archive file contains a record that has an invalid format, indicating corruption of the archive file.

User Action: Restore a new copy of the archive file from backup and enter the command again.

ILLCHAR, illegal character in 'string'

Error: You have specified a character that is not allowed in this context.

User Action: Correct the syntax and enter the command again.

ILLCLSNAM, illegal class name

Error: You have specified a class name incorrectly. Class names can consist of alphanumeric characters (A–Z, a–z, 0–9), hyphens (-), underscores (_), dollar signs ($), and a period (.).

User Action: Correct the class name syntax and enter the command again.

ILLCONREC, illegal control record in library element
ILLDATREC, illegal data record format in element file

Fatal: CMS has encountered an illegal control record in a library element.

User Action: These messages indicate that, through some means other than CMS, the element file has become corrupted. Under these circumstances, you should substitute a recent backup copy of the element file to insure the integrity of the data. See Chapter 9 for more information.

ILLEGALDEV, illegal device name

Error: You have included an illegal device specification in a directory or file specification.

User Action: Correct the device specification and enter the command again.

ILLELENAM, illegal element name

Error: You have specified an illegal element name.

User Action: Correct the syntax and enter the command again.

ILLELEXP, 'string' is an illegal element expression

Error: You have specified an illegal element expression.

User Action: Correct the syntax and enter the command again.

ILLFORMAT, illegal /FORMAT qualifier string

Error: You have specified an illegal format string. In a format string, the number sign (#) indicator must be followed by a second number sign or the letter E.

User Action: Correct the format string and enter the command again.

ILLGEN, illegal generation expression

Error: You have specified an illegal generation expression.

User Action: Correct the generation expression and enter the command again.

ILLGRPNAM, illegal group name

Error: You have specified the group name incorrectly. Group names can consist of alphanumeric characters (A–Z, a–z, 0–9), hyphens (-), underscores (_), and dollar signs ($).

User Action: Correct the syntax of the group name and enter the command again.

ILLHIST, illegal history string

Fatal: The history string for an element in your library contains more than one history format parameter (#H or #B).

User Action: Use the MODIFY ELEMENT/HISTORY command to establish the correct history string.

ILLNAME, 'name' is for CMS use only

Error: You have used a name that is reserved for CMS use.

User Action: See the Command Dictionary for information on name limitations.

ILLNOTE, illegal notes string

Fatal: The notes string for an element in your library contains more than one notes format parameter (#G).

User Action: Use the MODIFY ELEMENT/NOTES command to establish the correct notes string for the element.

ILLOBJTYP, illegal object type

Error: You have specified an illegal value for the object type.

User Action: See the *VAX DEC/Code Management System Callable Routines Reference Manual* for legal object type values.

ILLPAR, illegal format parameter in /HISTORY string
ILLPAR, illegal format parameter in /NOTES string

Error: You have given an illegal parameter in the value specified for the /HISTORY or /NOTES qualifier.

User Action: Correct the syntax and enter the command again.

ILLPOSVAL, /POSITION value must be from 1 to 511

Error: You have specified a position value that is not within the allowed range.

User Action: Correct the position value and enter the command again.

ILLREFDIR, reference copy directory cannot be a CMS library

Error: You cannot direct CMS to use a CMS library as a reference copy directory.

User Action: Make sure that you are specifying a non-library directory.

ILLRMK, illegal remark

Error: You have specified an illegal remark. CMS does not accept control characters in the remark. You can use only printable characters in a remark text string.

User Action: Correct the remark and enter the command again.

ILLSEQ, illegal sequence value

Fatal: The element file is identified as sequenced, but the format of the file is incorrect.

User Action: Substitute a backup copy of the element file. See Chapter 9 for more information.

ILLSUBTYP, illegal object subtype: 'subtype' in SET ACL or SHOW ACL command

Error: You have specified an illegal object subtype.

User Action: Change the object subtype to one of the allowable keywords.

User Action: See Chapter 7 for allowable keywords.

ILLVAR, illegal variant specification

Error: You have specified an illegal value for the /VARIANT qualifier (the value must be a single alphabetic character).

User Action: Correct the syntax and enter the command again.

INCRANGSPEC, incomplete range specification

Error: A deletion range must have both beginning and ending generations specified with /FROM (or /AFTER) and /TO (or /BEFORE).

User Action: Correct the syntax and enter the command again.

INSERTED, element 'name' inserted into group 'name'
INSERTED, group 'name' inserted into group 'name'

Success: CMS has successfully inserted the indicated element or group into the group that you specified.

User Action: None.

INSERTIONS, 'count' insertion(s) completed

Success: CMS has successfully inserted the indicated number of generations or elements.

User Action: None.

INUSE, library 'directory-spec' is in use, please wait

Informational: The library is currently being used by someone else.

User Action: None. You will need to wait until the other transaction is finished. CMS will automatically continue execution of your command as soon as the library is free. It is not necessary to reenter the command or press CTRL/C.

INVFETDB, invalid fetch data block passed to CMS

Error: You have specified an invalid fetch data block. This message is always preceded by CMS$_BADCALL.

User Action: Check to make sure that you are using the correct passing mechanism for the fetch data block argument in the call to CMS.

INVFIXMRS, generation has fixed-length records with maximum record size of zero

Error: CMS has found a generation created from a file with fixed-length records that has a stored maximum record size of zero bytes. Because RMS does not allow the creation of a file with fixed-length records of length zero, CMS will be unable to fetch or reserve this generation.

User Action: Use VERIFY/REPAIR.

INVLENGTH, invalid library data block length

Fatal: This message indicates an incorrect library data block length. This message is always preceded by CMS$_BADCALL.

User Action: Check to see that you have included a call to CMS$SET_LIBRARY before calls to other CMS routines.

INVLIBDB, invalid library data block passed to CMS

Error: You have specified an invalid library data block. This message is always preceded by CMS$_BADCALL.

User Action: Check to see that you have included a call to CMS$SET_LIBRARY before calls to other CMS routines.

INVSTRDES, invalid string descriptor at virtual address 'address'

Fatal: You have passed an incorrect argument to a CMS routine. This message is always preceded by CMS$_BADCALL.

User Action: Check the number of arguments in the call to CMS. Also check to make sure you are using the correct passing mechanism, and that you are passing the arguments in the correct order.

ISMEMBER, element belongs to a group
ISMEMBER, group belongs to a group
ISMEMBER, generation belongs to a class

Error: You cannot delete an element that belongs to a group or a class.

User Action: You must remove the element from any groups or classes (or remove the group from any groups) that it belongs to before you can delete it.

ISRESERVED, a generation of this element is reserved

Error: You can only modify the review attribute, reference copy attribute, and remark field of an element that is reserved. You cannot delete an element that is reserved.

User Action: Replace the element or cancel the reservation before continuing.

LIBALRINLIS, library 'directory-spec' is already in the library list

Error: A library can appear only once in the library list; you cannot enter it twice.

User Action: Use the SET NOLIBRARY command to remove the library from the list before reentering it in its new position.

LIBINSLIS, library 'directory-spec' inserted...

Informational: The indicated directory was inserted into the library list.

User Action: None.

LIBIS, library is 'directory-spec'

Informational: Your CMS library is set to the indicated directory.

User Action: None.

LIBLISMOD, library list modified

Informational: The CMS library list was modified.

User Action: None.

LIBLISNOTMOD, library list not modified

Informational: The CMS library list was not modified.

User Action: None.

LIBNOTINLIS, library 'directory-spec' not found in the library list

Error: The indicated library was not found in the library list.

User Action: None.

LIBREMLIS, library 'directory-spec' removed from the library list

Informational: The indicated directory was removed from the library list.

User Action: None.

LIBSET, library set

Success: CMS has successfully set your library. This message is returned from CMS$SET_LIBRARY; however, CMS does not pass this message to the callback message routine.

User Action: None.

MARKED, generation 'gen-number' of element 'name' marked for review

Success: CMS has successfully marked the indicated generation of the specified element for review and placed it on the review pending list.

User Action: None.

MARKS, 'count' generation(s) marked for review

Success: CMS has successfully marked the indicated number of generations for review and placed them on the review pending list.

User Action: None.

MAXARG, routine called with 'count' arguments; maximum allowed is 'count'

Fatal: You have tried to pass too many arguments to a CMS routine. This message is always preceded by CMS$_BADCALL.

User Action: Remove any extra arguments from the routine call.

MERGECONFLICT, 'count' changes successfully merged with 'count' conflicts

Warning: CMS has encountered conflicts during the merge transaction. The merge conflicts are flagged with asterisks in the output file.

User Action: Edit the output file to resolve the conflicts and delete the conflict asterisks. See Chapter 6 for more information.

MERGECOUNT, 'count' changes successfully merged with no conflicts

Informational: CMS did not encounter any conflicts during the merge transaction.

User Action: None.

MERGED, generations 'gen-exp' and 'gen-exp' merged

Informational: CMS merged the indicated elements during an annotate transaction.

User Action: None.

MINARG, routine called with 'count' arguments; minimum required is 'count'

Fatal: You have not provided enough arguments for a CMS routine. This message is always preceded by CMS$_BADCALL.

User Action: Check the number of arguments in the calls to CMS.

MISBLKSTR, a 'number' block was not hit during pass 1

Error: CMS has detected an error in the database. This condition will not prevent the proper functioning of the library, but its cause should be investigated.

User Action: Use VERIFY/REPAIR.

MISMATCON, control records in library element are mismatched

Fatal: CMS has encountered an error in the library element format.

User Action: Substitute a recent backup copy of the element. See Chapter 9 for more information.

MODACL, modified access control list for class 'name'
MODACL, modified access control list for element 'name'
MODACL, modified access control list for group 'name'
MODACL, modified access control list for class list

Success: CMS has successfully modified the access control list for the indicated library object.

User Action: None.

MODACL, modified access control list for element list
MODACL, modified access control list for group list
MODACL, modified access control list for history access
MODACL, modified access control list for library access

Success: CMS has successfully modified the access control list for the indicated library object.

User Action: None.

MODACL, modified access control list for 'name' command

Success: CMS has successfully modified the access control list for the indicated library object.

User Action: None.

MODACLS, 'count' access control list(s) modified

Success: CMS has successfully modified the access control lists of the indicated number of library objects.

User Action: None.

MODIFIED, class 'name' modified
MODIFIED, element 'name' modified
MODIFIED, group 'name' modified
MODIFIED, library 'directory-spec' modified

Success: CMS has successfully modified the attributes of the indicated class, element, group, or library.

User Action: None.

MODIFICATIONS, 'count' modification(s) completed

Success: CMS has successfully modified the attributes of the indicated number of classes, elements, or groups.

User Action: None.

MSSBLKSTR, 'count' 'type-number' type blocks found on pass 1, and 'count' blocks found on pass 2

Error: CMS has encountered an internal error in the library control structure.

User Action: Use VERIFY/REPAIR. If it is unable to correct the problem, use a backup copy of your library. See Chapter 9 for more information.

MULTCALL, multiple calls made to CMS$PUT_STRING

Warning: A callback routine has invoked the CMS$PUT_STRING routine more than once during a single execution of the callback routine.

User Action: Check to make sure that the callback routine is doing what you intend. If CMS$PUT_STRING is called more than once during a single invocation of a callback routine, the string buffer is overwritten with each call to CMS$PUT_STRING.

MULTPAR, multiple format parameters in /HISTORY string
MULTPAR, multiple format parameters in /NOTES string

Error: CMS has encountered more than one format parameter in a history or notes string.

User Action: Remove the extra format parameter and enter the command again.

MUSTBEDIR, 'string' must be a directory specification

Error: CMS expects a directory specification where you have used the indicated string.

User Action: Correct the parameter and enter the command again.

MUSTBEPOS, position value must be a positive integer
MUSTBEPOS, width value must be a positive integer

Error: You cannot use negative width or position values.

User Action: Correct the parameter and enter the command again.

MUTEXC, 'qualifier' and 'qualifier' are mutually exclusive qualifiers

Error: You cannot specify the indicated qualifiers on the same command line.

User Action: Enter the command again using only one of the qualifiers.

NEEDNUMBER, generation number must end with an integer

Error: You have specified a generation number in an illegal format.

User Action: Correct the generation number and enter the command again.

NEEDPERIOD, element name must include or end with a period

Error: The element name you have specified does not contain a period.

User Action: Correct the element name parameter and enter the command again.

NETNOTALL, network access not allowed

Error: You cannot use CMS to access libraries on remote nodes.

User Action: None.

NOACCEPT, error accepting 'name'

Error: CMS was unable to remove the generation of the specified element from the review pending list and mark it as accepted.

User Action: See the descriptions of the secondary messages for more information.

NOACCESS, no 'type' access to class 'name'
NOACCESS, no 'type' access to element 'name'
NOACCESS, no 'type' access to group 'name'
NOACCESS, no 'type' access to class list

Error: The access control list for the specified object does not give you the required access to perform the operation.

User Action: Use the SET ACL command to modify the access control list of the appropriate object.

NOACCESS, no 'type' access to element list
NOACCESS, no 'type' access to group list
NOACCESS, no 'type' access to history
NOACCESS, no 'type' access to library

Error: The access control list for the specified object does not give you the required access to perform the operation.

User Action: Use the SET ACL command to modify the access control list of the appropriate object.

NOACCESS, no 'type' access to 'name' command

Error: The access control list for the specified object does not give you the required access to perform the operation.

User Action: Use the SET ACL command to modify the access control list of the appropriate object.

NOACE, specified access control entry does not exist

Warning: The access control entry specified on the following line does not exist in the access control list.

User Action: Specify an existing access control entry and enter the command again.

NOANNOTATE, error annotating 'name'

Error: CMS was unable to annotate the indicated element.

User Action: See the descriptions of the secondary messages for more information.

NOBACKUP, no backup for data file 'name'

Fatal: CMS is unable to recover the library because the required file is missing. If a transaction is stopped so that the library contains a new element file that is incomplete, the earlier version of the file is required for recovery. If someone has purged the CMS library, or if some other action has been taken that deletes the earlier, complete version of the library file, CMS cannot recover the library.

User Action: Use a backup copy of the element file. See Chapter 9 for more information.

NOCANCEL, error canceling review of 'name'

Error: CMS was unable to remove the generation of the specified element from the review pending list and cancel the review.

User Action: See the descriptions of the secondary messages for more information.

NOCHANGES, no changes

Informational: You have replaced an element that has not been modified since you reserved it. CMS creates a new generation that is identical to the generation that was reserved.

User Action: None.

NOCLOSE, error closing file 'name'

Fatal: CMS was unable to close the indicated file.

User Action: You must close the file using a non-CMS procedure, then use VERIFY.

NOCLS, no classes found

Warning: CMS did not find any classes that matched the input specification.

User Action: None.

NOCMD, no commands found

Warning: There are no CMS commands that match the input specification.

User Action: None.

NOCOMMALIST, comma list not allowed in this context: 'string'

Error: You cannot use a comma list when modifying the name of a class, element, or group.

User Action: You must issue the command for each object one at a time.

NOCOMPARE, error comparing files

Error: CMS was unable to execute the DIFFERENCES command.

User Action: See the descriptions of the secondary messages for more information.

NOCONCUR, element 'name' is set to NOCONCURRENT access

Error: If an element is set to NOCONCURRENT, only one person can reserve it at a time.

User Action: To reserve the element, you must set it to /CONCURRENT or wait until it is no longer reserved.

NOCONRES, element 'name' is reserved with NOCONCURRENT access

Error: You have attempted to reserve an element that is already reserved; in this case, the reservation transaction specified NOCONCURRENT access for the duration of the reservation.

User Action: You must wait until the element is replaced, or until the reservation is canceled.

NOCONVERT, error converting Version 2 library to Version 3 format

Error: CMS was unable to convert your Version 2 library.

User Action: Check any additional messages. Before you use CONVERT LIBRARY again, delete any new files that may have been created in the new directory; you must supply an empty directory, which CONVERT LIBRARY can use for the Version 3.0 library.

NOCOPY, error copying 'name'

Error: CMS was unable to copy the indicated element.

User Action: See the descriptions of the secondary messages for more information.

NOCREATE, error creating library 'directory-spec'
NOCREATE, error creating class 'name'
NOCREATE, error creating element 'name'
NOCREATE, error creating group 'name'

Error: CMS was unable to create the indicated library, class, element, or group.

User Action: See the descriptions of the secondary messages for more information.

NODEFACL, no default access control list associated with object type

Error: The /DEFAULT qualifier can be used only with object types ELEMENT, CLASS, and GROUP.

User Action: Specify a supported object type and enter the command again.

NODELETE, error deleting class 'name'
NODELETE, error deleting element 'name'
NODELETE, error deleting group 'name'
NODELETE, error deleting history records

Error: CMS was unable to delete the indicated class, element, group, or history records.

User Action: See the descriptions of the secondary messages for more information.

NODELETIONS, no 'name' deletions performed

Error: CMS was unable to delete the indicated items.

User Action: See the descriptions of the secondary messages for more information.

NODELGEN1, you cannot delete generation 1

Error: You cannot delete generation 1 of an element.

User Action: Remove generation 1 from the deletion range specification and retry the operation.

NOELE, no elements found

Warning: The current library does not contain any elements that match the element name specified on the command line.

User Action: None.

NOELEENT, file 'filename' does not have an element entry

Error: CMS has detected a file in your library that does not have a corresponding entry in the control file. This file has been put there by means other than CMS. If you enter the VERIFY/REPAIR command on the library in this state, CMS deletes the extraneous file.

User Action: Remove the file from the library directory.

NOFETCH, error fetching element 'name'

Error: CMS was unable to fetch the indicated element. This message can be displayed during a fetch transaction, and also during a differences transaction. CMS executes a fetch transaction to use a library file in a differences transaction.

User Action: See the descriptions of the secondary messages for more information.

NOFILE, no input file found

Error: This message is displayed if CMS is unable to find an input file for the primary input stream for a differences transaction.

User Action: Check the arguments in the call to CMS.

NOGENDELETED, no generations of element 'name' deleted

Error: CMS encountered an error while deleting generations of the indicated element.

User Action: See the descriptions of the secondary messages for more information.

NOGENS, no generations found in the deletion range

Error: The specified deletion range does not contain any generations.

User Action: None.

NOGRP, no groups found

Warning: The current library does not contain any groups that match the group name specified in the command line.

User Action: None.

NOHIS, no history records found

Warning: The current library does not contain any records of transactions associated with the specified object.

User Action: None.

NOHISNOTES, history and notes will not be included in output file

Informational: CMS will not provide history or notes information in the output file because the generation being fetched or reserved has fixed-length records.

User Action: To see the history and notes information for the generations you are retrieving, use the ANNOTATE command.

NOHISPAR, no history parameter specified in history string

Error: You did not include #H or #B in the history string.

User Action: Correct the history string and enter the command again.

NOINSERT, error inserting 'name' into group 'name'

Error: CMS was unable to insert the indicated element or group into the indicated group.

User Action: See the descriptions of the secondary messages for more information.

NOINPUT, no input data supplied by callback routine

Error: Your callback routine has not provided CMS with input.

User Action: Check to see that the callback routine contains a call to the CMS$PUT_STRING routine.

NOMARK, error marking 'name'

Error: CMS was unable to mark a generation of the specified element for review.

User Action: See the descriptions of the secondary messages for more information.

NOMODACL, error modifying access control list for class 'name'
NOMODACL, error modifying access control list for element 'name'
NOMODACL, error modifying access control list for group 'name'
NOMODACL, error modifying access control list for class list

Error: CMS was unable to modify the access control list for the indicated library object.

User Action: See the descriptions of the secondary messages for more information.

NOMODACL, error modifying access control list for element list
NOMODACL, error modifying access control list for group list
NOMODACL, error modifying access control list for history access
NOMODACL, error modifying access control list for library access

Error: CMS was unable to modify the access control list for the indicated library object.

User Action: See the descriptions of the secondary messages for more information.

NOMODACL, error modifying access control list for 'name' command

Error: CMS was unable to modify the access control list for the indicated library object.

User Action: See the descriptions of the secondary messages for more information.

NOMODARG, arguments do not specify any modifications to 'object'

Error: You have not provided the necessary arguments to change the characteristics of the specified element, group, class, or library.

User Action: Add a qualifier to your command line specifying a modification and enter the command again. If you are using a CMS$MODIFY_xxx routine, check the arguments in the call to CMS.

NOMODIFY, error modifying class 'name'
NOMODIFY, error modifying element 'name'
NOMODIFY, error modifying group 'name'
NOMODIFY, error modifying library 'directory-spec'

Error: CMS was unable to modify the attributes for the indicated class, element, group, or library.

User Action: See the descriptions of the secondary messages for more information.

NOOBJ , no objects found

Warning: The current library does not contain any objects that match the objects you specified.

User Action: None.

NOOBJTYP, no object type specified

Error: You have not specified an object type.

User Action: Use the /OBJECT_TYPE qualifier to specify the object type.

NORECOVER, error recovering library

Error: CMS is unable to recover your library.

User Action: Use a backup copy of the library. See Chapter 9 for more information.

NOREF, error referencing ′directory-spec′

Error: CMS is unable to use the indicated directory.

User Action: See the descriptions of the secondary messages for more information.

NOREFDIR, no reference copy directory set for current library

Error: You have tried to give an element the reference copy attribute in a library that does not have an associated reference copy directory.

User Action: Either specify a reference copy directory for the library using the MODIFY LIBRARY command, or do not specify the reference copy attribute for the element.

NOREFELE, reference copy found for element with the /NOREFERENCE_COPY qualifier.

Error: CMS has discovered a reference copy for an element with the /NOREFERENCE_COPY qualifier.

User Action: Use VERIFY/REPAIR to delete the unwanted reference copy.

NOREJECT, error rejecting 'name'

Error: CMS was unable to remove the generation of the specified element from the review pending list and mark it as canceled.

User Action: See the descriptions of the secondary messages for more information.

NOREMARK, error adding remark to library

Error: CMS was unable to enter your remark in the library history.

User Action: See the descriptions of the secondary messages for more information.

NOREMOVAL, error removing 'name' from group 'name'

Error: CMS was unable to remove the indicated element or group from the indicated group.

User Action: See the descriptions of the secondary messages for more information.

NOREPAIR, error repairing library

Error: CMS was unable to repair your library.

User Action: Use a backup copy of your library. See Chapter 9 for more information.

NOREPCMD, command 'command-name' not repaired

Error: CMS was unable to add the indicated command to the library's list of commands.

User Action: See descriptions of the secondary messages for more information.

NOREPEDF, cannot repair 'name'

Error: CMS is unable to repair the indicated element data file.

User Action: Use a backup copy of the element file. See Chapter 9 for more information.

NOREPGENMRS, maximum record size of generation 'number' of element 'name' not repaired

Error: CMS was unable to repair the maximum record size stored for the indicated generation.

User Action: Either correct the conditions indicated by the secondary messages and use VERIFY/REPAIR again, or delete the element and create it again.

NOREPLACE, error replacing 'name'

Error: CMS was unable to replace the indicated element in your library.

User Action: See the descriptions of the secondary messages for more information.

NOREPREF, cannot repair reference copy for 'name'

Error: CMS is unable to repair the indicated reference copy.

User Action: Use a backup copy of the reference copy. See Chapter 9 for more information.

NOREPRO, cannot replace a read-only element

Error: The protection associated with the element file has been changed since you reserved it.

User Action: Change the file protection to allow both read and delete access.

NORES, no reservations found

Warning: None of the elements in your library are reserved.

User Action: None.

NORESERVATION, error reserving 'name'

Error: CMS was unable to reserve the indicated element.

User Action: See the descriptions of the secondary messages for more information.

NORESNOCON, cannot reserve element 'name' as NOCONCURRENT (other reservers)

Error: You cannot specify the /NOCONCURRENT qualifier when you reserve an element that is already reserved by someone else.

User Action: Reserve the element concurrently, or wait until the element is replaced or the reservation is canceled.

NORESRO, cannot reserve a read-only element

Error: The element file does not allow the required access for you to reserve the element.

User Action: Change the access so that you have both read and delete access.

NORETRIEVE, error retrieving generation 'number' from archive 'file-spec'

Error: CMS was unable to retrieve the indicated generation from the indicated archive file.

User Action: See the descriptions of the secondary messages for more information.

NOREV, no reviews pending

Warning: None of the generations specified have reviews pending.

User Action: None.

NOREVIEW, error reviewing 'name'

Error: CMS was unable to associate a review remark with the associated generation of the specified element.

User Action: See the descriptions of the secondary messages for more information.

NOREVPEND, generation 'gen-number' of 'name' does not have a review pending

Error: The indicated generation of the specified element has no reviews pending.

User Action: None.

NOREVSPEND, no reviews pending for element 'name'

Error: The indicated element has no generations with reviews pending.

User Action: None.

NORMAL, normal successful completion

Success: This code indicates the successful completion of a user-supplied routine that is invoked from a callable CMS routine.

User Action: None.

NOSINCE, error executing /SINCE operation

Error: CMS is unable to display the library history in the method indicated by /SINCE.

User Action: See the descriptions of the secondary messages for more information.

NOSRCHLST, search lists are not allowed in this context

Error: You cannot use a VMS search list as your default library and set your CMS library search list to a single library. Also, you must not define CMS$LIB for your own purposes; the CMS$LIB logical name is reserved for CMS use only.

User Action: None.

NOSUPERSEDE, element 'name' is not in class 'name', cannot supersede

Error: You cannot use /SUPERSEDE if the class does not already contain a generation from the specified element.

User Action: See the description of the /ALWAYS, /IF_ABSENT, and /SUPERSEDE qualifiers for the INSERT GENERATION command in the Command Dictionary.

NOTBYCMS, data file 'name' not closed by CMS

Error: Someone has used something other than CMS to access your CMS library.

User Action: Use VERIFY/REPAIR.

NOTCOMPLETED, last transaction was not completed

Error: The last library transaction was interrupted before completion. Because an incomplete transaction leaves your library in a potentially inconsistent state, you cannot manipulate the library until you have recovered the library.

User Action: Use VERIFY/RECOVER.

NOTCMSLIB, 'directory-spec' is not a CMS library

Error: You have specified a directory that is not a valid CMS library.

User Action: Correct the directory specification and enter the command again.

NOTCRELIB, first history record is not a CREATE LIBRARY transaction

Warning: The first history record of every history file should be a CREATE LIBRARY transaction. It is likely that the history file has been edited. This condition does not prevent CMS from using the file.

User Action: None.

NOTDIRDES, 'generation' is not a direct descendant of 'generation'

Error: The first generation indicated must be a descendant on the same line of descent as the second generation indicated.

User Action: None.

NOTESVALREQ, /NOTES value required for the position attribute

Warning: If you establish the **position** attribute for an element, you must also supply a notes string.

User Action: Establish a notes string for the element.

NOTFOUND, class 'name' not found
NOTFOUND, element 'name' not found
NOTFOUND, group 'name' not found
NOTFOUND, reservations for 'name' not found

Error: CMS was unable to find the indicated class, element, group, or reservation.

User Action: None.

NOTLOGGED, 'command-name' commands are not logged in the library history

Informational: You have attempted to show the history of ANNOTATE, DIFFERENCES, or SHOW commands. CMS does not log these commands in the library history; however, CMS does not prevent you from including them in the syntax of the SHOW HISTORY command.

User Action: None.

NOTNOREF, not all elements are set to /NOREFERENCE_COPY

Error: You cannot issue the MODIFY LIBRARY/NOREFERENCE_COPY command unless all elements are set with /NOREFERENCE_COPY.

User Action: Use the MODIFY ELEMENT/NOREFERENCE_COPY command on all elements with the reference copy attribute and issue the MODIFY LIBRARY command again. Use the SHOW ELEMENT/FULL command to list the reference copy attribute of your elements.

NOTRESBYOU, element 'name' is not reserved by you

Error: Unless you have BYPASS access, you cannot replace an element that you have not reserved.

User Action: None. See Chapter 7 for more information.

NOTSET, library not set

Error: CMS was unable to set your library to the specified directory.

User Action: Make sure that the directory is a valid library (or an empty directory for CREATE LIBRARY) and enter the command again.

NOTTHERE, element is not in class
NOTTHERE, element is not in group
NOTTHERE, group is not in group

Error: CMS is unable to execute a command such as REMOVE GENERATION, REMOVE ELEMENT, or REMOVE GROUP because the element or group you specified does not exist in the specified location.

User Action: Use the /CONTENTS qualifier with the SHOW CLASS or SHOW GROUP command.

NOUNRESERVE, error unreserving 'name'

Error: CMS was unable to unreserve the indicated element.

User Action: See the descriptions of the secondary messages for more information.

NOVERIFY, error verifying library

Error: CMS was unable to verify your library.

User Action: See Chapter 9 for more information.

NOWLDCARD, wildcards are not allowed in this context: 'string'

Error: You cannot use wildcard characters in the indicated context.

User Action: Correct the syntax and enter the command again.

NULLARG, required argument is null

Fatal: The CMS routine call requires an argument that you have not supplied.

User Action: Check to see that you have specified all the necessary arguments, and that you have supplied placeholders where required.

NULLSTR, null string is not allowed in this context: 'string'

Error: You have specified a null string.

User Action: Check that you have supplied the correct arguments, and that you have used placeholders where necessary.

ONEPERIOD, only one period is allowed in an element name

Error: The element name that you specified contains more than one period.

User Action: Make sure that the element name contains only one period and enter the command again.

OPENARC, error opening archive file 'file-spec'

Error: CMS was unable to open the indicated archive file for input or output, depending on the operation.

User Action: See the descriptions of the secondary messages for more information.

OPENIN, error opening 'file-spec' as input
OPENIN1, error opening 'file-spec' as input
OPENIN2, error opening 'file-spec' as input

Error: CMS was unable to open the indicated file. OPENIN1 and OPENIN2 are used to identify the first and second file specifications given as arguments to DIFFERENCES.

User Action: See the descriptions of the secondary messages for more information.

OPENOUT, error opening 'file-spec' for output

Error: CMS was unable to open the indicated file for output.

User Action: See the descriptions of the secondary messages for more information.

OVERDRAFT, disk quota exceeded; using overdraft for 'file-spec'

Informational: You have exceeded your disk quota during a CMS transaction. In this case, CMS uses overdraft quota to finish the transaction.

User Action: Reset your quota or remove some files so that you can continue.

POSVALREQ, /POSITION value required for the notes attribute

Error: If you establish the **notes** attribute for an element, you must also supply a position value.

User Action: Establish a position value for the element.

PROCEEDING, proceeding...

Informational: The library is now available for your transaction.

User Action: None.

QUALCONFLICT, conflicting qualifiers

Error: You have specified conflicting qualifiers on the same command line.

User Action: Correct the syntax and enter the command again.

READERR, error reading from 'filename'
READIN, error reading 'filename'

Error: CMS was unable to read the indicated file.

User Action: See the descriptions of the secondary messages for more information.

READONLY, class 'name' is set to READ_ONLY
READONLY, group 'name' is set to READ_ONLY

Error: You cannot modify the characteristics of a class or group that is set to READ_ONLY.

User Action: Change the class or group to NOREAD_ONLY, then modify the other characteristics.

RECGRP, inserting 'name' into 'name' would create a recursive group

Error: You cannot insert one group into another if there is already a connection between the two groups.

User Action: Check to see that you are establishing the group hierarchy that you intend.

RECNOTNEC, recovery is not necessary; the library is in a safe state

Error: You have entered VERIFY/RECOVER for a functioning library.

User Action: None.

RECOVERED, library 'directory-spec' recovered

Success: CMS has successfully recovered your library.

User Action: None.

REFMISMAT, element has /REFERENCE_COPY attribute, library does not

Error: The element you specified has the /REFERENCE_COPY attribute enabled, but there is no reference copy directory set for the library. This situation can occur when using a library that was created with CMS V2, since that version did not enforce any correlation between the attributes.

User Action: Use VERIFY/REPAIR or MODIFY ELEMENT /NOREFERENCE_COPY to correct the mismatched attributes.

REFMISS, reference copy for element 'name' is missing

Error: An element with the /REFERENCE_COPY attribute does not have a corresponding reference copy.

User Action: Use VERIFY/REPAIR to create a new reference copy for the element.

REFREPAIR, reference copies must be repaired—use VERIFY/REPAIR

Warning: This message appears after MODIFY LIBRARY has been used to change the reference copy directory, if the contents of the directory do not agree with the contents of the CMS library. CMS must repair the files in the reference copy directory before further operations can be done.

User Action: Either use VERIFY/REPAIR or use MODIFY LIBRARY to change the reference copy directory. See Chapter 3 and the description of the VERIFY command in the Command Dictionary for more information.

REJECTED, generation 'gen-number' of element 'name' rejected

Success: CMS has successfully marked a generation as rejected and removed it from the review pending list.

User Action: None.

REJECTIONS, 'count' generation(s) rejected

Success: CMS has successfully marked the indicated number of generations as rejected and removed them from the review pending list.

User Action: None.

REMARK, remark added to history file

Success: CMS has successfully added your remark to the history file.

User Action: None.

REMOVALS, 'count' removals completed

Success: CMS has successfully completed the indicated number of removals from one or more classes or groups.

User Action: None.

REMOVED, element 'name' removed from group 'name'
REMOVED, group 'name' removed from group 'name'

Success: CMS has successfully removed the indicated element or group from the indicated containing group.

User Action: None.

REPAIRED, library 'directory-spec' repaired

Success: CMS has successfully repaired your library.

User Action: None.

REPCMD, command 'command-name' repaired

Success: CMS has successfully added 'command-name' to the library's list of commands.

User Action: None.

REPDEL, extraneous file 'name' deleted

Informational: VERIFY/REPAIR has deleted a file from the library. The file did not have a corresponding entry in the CMS control file. See the description of NOELEENT for more information.

User Action: None.

REPEDF, element 'name' repaired

Informational: CMS has repaired the indicated element data file.

User Action: None.

REPGENMRS, maximum record size of generation 'number' of element 'name' repaired to 'size' bytes

Informational: CMS has repaired the stored maximum record size for the indicated generation.

User Action: None.

REPLACEMENTS, 'count' element(s) replaced

Success: CMS has successfully replaced the indicated number of elements in the library.

User Action: None.

REPREF, reference copy for element 'name' repaired

Informational: CMS has repaired the indicated reference copy.

User Action: None.

RESERVATIONS, 'count' element(s) reserved

Success: CMS has successfully reserved the indicated number of elements.

User Action: None.

RESERVED, generation 'gen-number' of element 'name' reserved
RESERVED, generation 'gen-number' of element 'name' reserved and merged with 'gen-number'

Success: CMS has successfully reserved the indicated element generation.

User Action: None.

RETRIEVALS, 'count' generation(s) retrieved

Success: CMS has successfully retrieved the indicated number of generations.

User Action: None.

RETRIEVED, generation 'gen-number' of element 'name' retrieved from 'archive-file'

Success: CMS has successfully retrieved the indicated element generation.

User Action: None.

REVIEWED, generation 'gen-number' of element 'name' reviewed

Success: CMS has successfully associated a review remark with the indicated generation of the specified element.

User Action: None.

REVIEWS, 'count' generation(s) reviewed

Success: CMS has successfully associated a review remark with the indicated number of generations.

User Action: None.

REVPENDING, a generation of this element has a review pending

Error: You cannot delete an element if one of its generations has a review pending.

User Action: Accept or reject the generation, or cancel the review; then enter the command again.

SAMELINE, generations 'gen-exp' and 'gen-exp' are on the same line of descent

Error: You cannot merge two element generations that are on the same line of descent.

User Action: Make sure that you are specifying the correct generations.

SEQFAIL, illegal sequence value in library element

Error: A sequence value field in a record of an element file has been corrupted.

User Action: Use a backup copy of the element. See Chapter 9 for more information.

SEQUENCED, normal successful completion—sequenced open

Success: A CMS$FETCH_OPEN transaction was completed successfully.

User Action: None.

SIZEMISMAT, cannot merge generations with different-sized records

Error: CMS cannot merge two generations with fixed-length records if the records are not of the same length.

User Action: Reserve the generation with the smaller records and convert those records to the larger size. Then replace the generation and attempt the merge again, this time using the newly created generation.

STARTHIS, library history starts at 'date'

Error: You have specified a /BEFORE date prior to the date of the CREATE LIBRARY transaction for this library.

User Action: Correct the date and enter the command again.

STOPPED, wildcard action terminated

Success: You have terminated a wildcard action from a user-supplied routine that is invoked by a callable CMS routine.

User Action: None.

SUPERSEDED, library list superseded

Informational: The previously existing library list has been superseded by the library or libraries you specified.

User Action: None.

SYSTIMDIF, last transaction executed at 'time'

Error: This message is secondary to SYSTIMERR.

User Action: See the explanation of SYSTIMERR.

SYSTIMERR, system time has been set incorrectly

Error: CMS has detected an error in system time. CMS displays this message if the current system time is less than the time of the last transaction. One of three situations may have occurred: the system time was inadvertently set back; at some point the system was set ahead and then later corrected (and any transactions that were executed during this time period may appear to have occurred in the future); or there is a difference between the times of two nodes on a cluster.

User Action: Check to see that the system time is correct. If your system is part of a cluster, check to see that the system times for all the nodes on the cluster are correct. Correct the system times and use VERIFY/REPAIR if necessary. (See Chapter 9 for more information.)

TIMEORDER, /BEFORE and /SINCE time values cannot be resolved

Error: You have specified an incorrect sequence of time values for this combination of qualifiers. (The /BEFORE value cannot indicate a time that occurs prior to the /SINCE value.)

User Action: Check to make sure that you are entering the correct values and enter the command again.

TOOLONG, 'string' is too long; maximum is 'count' character(s)

Error: You have provided a character string that is longer than allowed in this situation.

User Action: Correct the syntax and enter the command again.

TOOMANYLIBS, too many libraries in library list

Error: You have attempted to specify more than 128 libraries in a library list.

User Action: Delete some libraries from the library list and enter the command again.

TRYAGNLAT, library locked; try again later

Error: The library has been locked long enough to cause CMS to exit.

User Action: Wait until the library is unlocked and enter the command again.

UNDEFLIB, library is undefined

Warning: Your CMS library is undefined.

User Action: Make sure you are specifying a valid CMS library.

UNFOUT, unformatted output cannot be specified for an output file

Error: If you provide an output file for the CMS$DIFFERENCES routine, you cannot also direct CMS to provide unformatted output.

User Action: Check to see that you are passing the correct arguments to CMS.

UNRESERVED, element 'name' unreserved

Success: CMS has successfully canceled the reservation for the indicated element.

User Action: None.

UNRESERVES, 'count' element(s) unreserved

Success: CMS has successfully canceled the indicated number of reservations.

User Action: None.

UNSUPFRMT, 'filename' is in an unsupported format

Error: CMS does not support the file organization, record format, or record attributes of the indicated file.

User Action: See Chapter 3 for information about supported file formats.

USERECOVER, use VERIFY/RECOVER

Error: Your library is in an inconsistent state.

User Action: Use VERIFY/RECOVER. See the description of the VERIFY command in the Command Dictionary for more information.

USEREPAIR, use VERIFY/REPAIR

Error: This message appears when your library contains one or more files that have an invalid checksum or do not have a checksum, or when someone has accessed your library using means other than CMS.

User Action: Use VERIFY/REPAIR. See the description of the VERIFY command in the Command Dictionary for more information.

USERERR, error returned from a user-supplied routine

Error: A callback routine that you have supplied has returned an error to CMS.

User Action: According to situation.

USESETLIB, use SET LIBRARY

Error: Your CMS library is undefined.

User Action: Use the SET LIBRARY command to access a library.

VARINRANGE, range has variants

Error: A deletion range cannot contain the root of a variant line of descent.

User Action: Delete the variant lines of descent explicitly, then delete the original range.

VARLETTER, variant letter is misplaced in generation number

Error: You have incorrectly specified a generation number.

User Action: Correct the generation number syntax and enter the command again.

VER2, internal contiguous space verified
VERARC, archive control block verified
VERCLS, class list verified
VERCMD, command list verified

Informational: CMS has successfully verified the indicated portion of the library.

User Action: None.

VERCON, control file verified

Informational: CMS has successfully verified the indicated portion of the library.

User Action: None.

VEREDF, element 'name' verified

Informational: CMS has verified the indicated element data file.

User Action: None.

VEREDFERR, element 'name' verified with errors

Error: CMS has encountered errors during the verification transaction.

User Action: See the descriptions of the secondary messages for more information.

VEREDFS, element data files verified

Informational: CMS has successfully verified the elements in your library.

User Action: None.

VERELE, element list verified
VERFRE, internal free space list verified
VERGRP, group list verified

Informational: These messages are displayed during the verification of the library data structures.

User Action: None.

VERIFIED, library 'directory-spec' verified

Success: CMS has successfully verified your library.

User Action: None.

VERLMTERR, 'directory-spec' requires a higher version limit

Error: Any directory that you intend to use as a CMS library must have a version limit of at least 2.

User Action: Change the version limit of the directory and enter the command again.

VERREF, reference copy for element 'name' verified

Informational: CMS has verified the indicated reference copy.

User Action: None.

VERREFERR, reference copy for 'name' verified with errors

Error: CMS has encountered errors during the verification transaction. This message may appear during a VERIFY operation.

User Action: See the descriptions of the secondary messages for more information.

VERREFERRW, reference copy for 'name' verified with errors

Warning: CMS has encountered errors during the verification transaction. This message may appear during a MODIFY LIBRARY operation.

User Action: See the descriptions of the secondary messages for more information.

VERREFS, reference copies verified

Informational: CMS has successfully verified the reference copies for the elements in your library.

User Action: None.

VERRES, reservation list verified
VERSTR, internal string list structure verified

Informational: These messages are displayed during the verification of the library data structures.

User Action: None.

WAITING, library 'directory-spec' is still in use
WAITING, file 'name' is still in use

Informational: The INUSE message displayed earlier is still in effect. Another user is still accessing the library.

User Action: None. CMS automatically continues execution of your command as soon as the library is free. It is not necessary to reenter the command or press CTRL/C.

WILDCONFLICT, wildcard conflict between element name and /INPUT

Error: You have provided a wildcard input file specification without the corresponding element name wildcard.

User Action: Correct the syntax and enter the command again.

WILDNEEDED, Wildcard needed: 'string'

Error: You have used a comma list in a context where a wildcard is necessary to resolve an expression, for example, in MODIFY ELEMENT/NAME or COPY ELEMENT.

User Action: Correct the syntax and enter the command again.

WILDVER, version number wildcard not allowed on /INPUT

Error: CMS does not allow version number wildcards in element creation or replacement transactions.

User Action: Correct the syntax and enter the command again.

WRITEERR, error writing to 'filename'

Error: CMS was unable to write to the indicated file.

User Action: See the descriptions of the secondary messages for more information. If the problem involves any library files, use the VERIFY command.

ZEROADD, address of zero required in call frame entry

Error: Your call to the CMS$UNRESERVE routine does not contain the required address of zero in the call frame.

User Action: Correct the arguments in the call.

ZLENBLK, zero-length block found during pass 2

Error: CMS has discovered a error in the control file.

User Action: Use a backup copy of the library. See Chapter 9 for more information.

Appendix C

CMS Library Storage Method

This appendix contains information that may be useful to you as you build and use your CMS library. In general, project planning has the greatest impact on how you can best use CMS. Each project has its own characteristics that determine how you should organize your library.

CMS stores the entire text of the first generation of an element. This file is called a delta file. Each time you replace an element, CMS determines what has been changed in the element files, and, to save storage space, stores only the new and changed lines of successive generations. (See Chapter 4 for more information.) To estimate the required storage for the library, you should allow three times the amount of disk space that you would normally allow for one copy of all project files.

The following example shows a file in a CMS library:

```
    .
    .
    .
(1,2,3)         APPLES
(1,2,3)         BANANAS
(1,2,3)         CHERRIES
(1)             POOCHES
(2)             PAUNCHES
(3)             PEACHES
(1,2,3)         ELDERBERRIES
    .
    .
    .
```

This example shows that each data item is numbered according to the element generations in which the item appears. The line POOCHES (occurring in the first generation) has been changed to PAUNCHES in the second generation, and changed again to PEACHES in the third generation.

CMS can provide a complete copy of any of the three generations whenever necessary. (It can also produce an annotated copy showing all changes in each generation and identify the users who made those changes.) However, the data requires only seven lines of storage space in the library. Conventional storage methods require 15 lines, five lines for each of the three copies of the data.

You do not need to save backup copies of CMS library files in your account. Normal system backup procedures should be followed for a CMS library. CMS itself maintains a certain amount of backup information so that it can recover from an incomplete transaction after a system failure.

Elements are stored most efficiently when modifications leave the majority of the file lines unchanged. CMS stores only one copy of an element; this copy includes all lines from the first generation plus all modifications to successive generations. Thus, the number of differences (relative to the number of original lines) affects system efficiency.

For example, the modifications to successive generations of a FORTRAN source program might typically change 15 to 20 percent of the lines during the development of that program. Because the bulk of the program does not change, this kind of element is ideal for a CMS library. However, because the same program's listing file would change greatly with each modification due to the compiler's effect on line numbers, addresses, and so forth, each generation of the stored listing element would contain almost as many differences as original lines.

Appendix D

System Management Considerations

This appendix contains information about running CMS on a VMS system.

D.1 Library Backup

You should use normal system backup procedures to back up your CMS libraries. Although CMS is designed to recover from certain kinds of failures, there are some events that leave a library in a state that is impossible to recover or repair. If this is the case, you should have a recent backup copy of the library that you can use as a replacement.

If there is an error in any of the library data structures contained in the control file, you should use a backup copy of the entire library. In some cases, CMS may display an error message indicating that an element file has been corrupted. In this case, you can substitute a recent backup copy of the element file.

If you substitute a backup version of an element file for a corrupted element file, you should note the following effect: if the library control file indicates that additional generations were created after the backup element generation, the contents of those additional generations are the same as those of the backup element generation. Thus, if you have an element that has 10 generations, and you substitute a file that corresponds to generation 9, the element is still valid; however, generations 9 and 10 have the same contents.

When you use a backup copy of an element data file, follow these rules:

- Use full wildcards (file name, extension, and version number) when you copy the backup file so that you maintain all of the file header information.

- Remove the corrupted file from the library to avoid time conflicts between the earlier backup copy and the later corrupted file.
- After restoring the element data file, use VERIFY/REPAIR to fix the checksum.

See Chapter 9 for more information on library maintenance.

D.2 System Time Errors

Under certain circumstances, CMS may display the following messages:

```
%CMS-E-NOREF, error referencing 'directory'
-CMS-E-SYSTIMERR, system time has been set incorrectly
-CMS-E-SYSTIMDIF, last transaction executed at 'time'
```

CMS displays these messages if the current system time is before the time of the last transaction. One of three situations may have occurred:

- The system time was inadvertently set back.
- At some point the system time was set ahead and then later corrected (any transactions that were executed during this time period may appear to have occurred in the future).
- There is a difference between the system time of two nodes on a cluster.

When CMS indicates a system time error, you should first check the system times and correct any error. If your system is part of a cluster, check all of the nodes on the cluster. After you have corrected the system time error, compare the current system time with the time of the last transaction (reported in the SYSTIMDIF message). If the difference between the time of the last transaction and the current system time is small, wait until the current system time is later than the last transaction and execute the command again. If the time difference is large, use VERIFY/REPAIR.

D.3 Quotas

The following list shows quota requirements for CMS:

AST limit	14
Buffered I/O byte count quota	14,000
ENQUEUE	See the following paragraph

Open file quota	6
Timer queue quota	1

CMS requires an ENQUEUE quota of 1, plus 1 for each library involved in a transaction. Thus, for most transactions, CMS needs a quota of 2; however, COPY ELEMENT and CMS$FETCH_xxx transactions may use two or more libraries and thus require a higher ENQUEUE quota.

Index

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-DEC-DEMO (800-332-3366) using a 1200- or 2400-baud modem. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	————	Local Digital subsidiary or approved distributor
Internal[1]	————	USASSB Order Processing - WMO/E15 *or* U.S. Area Software Supply Business Digital Equipment Corporation Westminster, Massachusetts 01473

[1]For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

Reader's Comments

Guide to VAX DEC/Code Management System
AA–KL03D–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less ______________________________

____

What I like best about this manual is ______________________________

____

What I like least about this manual is ______________________________

____

I found the following errors in this manual:

Page	Description

Additional comments or suggestions to improve this manual:

____

____

____

I am using **Version** ______ of the software this manual describes.

Name/Title ______________________________ Dept. ____________

Company ______________________________ Date ____________

Mailing Address ______________________________

______________________________ Phone ____________

— Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK02-2/N53
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

— Do Not Tear - Fold Here

Reader's Comments

Guide to VAX DEC/Code Management System
AA-KL03D-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less ______________________________

____

What I like best about this manual is ______________________________

____

What I like least about this manual is ______________________________

____

I found the following errors in this manual:

Page	Description

Additional comments or suggestions to improve this manual:

____

____

____

I am using **Version** ______ of the software this manual describes.

Name/Title ______________________________ Dept. ______________

Company ______________________________ Date ______________

Mailing Address ______________________________

______________________________ Phone ______________

– Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK02-2/N53
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

– Do Not Tear - Fold Here