IBM

SC26-4504-1

MVS/ESA
Data Facility Product Version 3: Customization

Version 3 Release 1

MVS/ESA Data Facility Product Version 3: Customization

SC26-4504-1

Version 3 Release 1

Second Edition (June 1989)

This edition replaces and makes obsolete the previous edition, SC26-4504-0.

This edition applies to Version 3 Release 1 of MVS/DFP™, Program Number 5665-XA3, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

The changes for this edition are summarized under "Summary of Changes" following the table of contents. Specific changes are indicated by a vertical bar to the left of the change. A vertical bar to the left of a figure caption indicates that the figure has changed. Editorial changes that have no technical significance are not noted.

Changes are made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's program may be used. Any functionally equivalent program may be used instead.

Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. If you request publications from the address given below, your order will be delayed because publications are not stocked there.

A Reader's Comment Form is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Department J57, P. O. Box 49023, San Jose, California, U.S.A. 95161-9023. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Trademarks

The following names have been adopted by IBM for trademark use and are used throughout this publication:

ESA/370™

MVS/DFP™

MVS/ESA™

MVS/SP™

MVS/XA™

Summary of Changes

Second Edition, June 1989

New Programming Support for Release 1

A new installation exit has been provided to allow you to customize the messages for display on an IBM 3480 tape drive. You can also use this exit to request automatic cartridge load. Information on this installation exit is provided in Chapter 9, "Data Management Installation Exit Routines."

Service Changes

This book has been divided into three sections:

1. "Introduction to Customization"
2. "MVS/DFP User Exits"
3. "MVS/DFP Installation Exits"

The information on testing exits and user interfaces has been moved to "Introduction to Customization." The information on status following an input/output operation has been moved into Chapter 5, "Data Management User-Written Exit Routines." Descriptions of the ISMF command and line operator table formats have been moved into Chapter 8, "Interactive Storage Management Facility (ISMF)."

Information on the CREATE command has been added to Chapter 6, "User Exit Routines Specified with Utilities." Descriptions of the DADSM SCRATCH and RENAME parameter lists, IGGDASCR and IGGDAREN, have been added to Chapter 9, "Data Management Installation Exit Routines." Sample precalculation and postcalculation exits have been added to Chapter 9, "Data Management Installation Exit Routines."

Other minor technical and editorial changes have been made.

First Edition, December 1988

New Programming Support for Release 1

The DCB OPEN exit and the DCB OPEN installation exit can be used to force OPEN to obtain a system-determined block size for DASD data sets. Information has been added to Chapter 5, "Data Management User-Written Exit Routines" and Chapter 9, "Data Management Installation Exit Routines" that describes this support.

Four new replaceable modules are available as preprocessing and postprocessing exits for DADSM Scratch and Rename. Information on these modules has been added to Chapter 9, "Data Management Installation Exit Routines."

Automatic class selection (ACS) installation exits allow you to code exit routines that provide capabilities beyond the scope of the ACS routines. ACS installation exits exist for data class, storage class, and management class ACS routines. Information on these exits is in Chapter 11, "Automatic Class Selection (ACS) Installation Exits."

New indicators for the Storage Management Subsystem (SMS) have been added to the format-1 and format-4 data set control blocks (DSCBs) for use with DADSM installation exits. Information on these indicators is in Appendix B, "SMS Indicators for DADSM Installation Exits."

Service Changes

Information on the Interactive Storage Management Facility panel IDs, member names, panel displays, and formats has been updated to reflect any changes.

MVS/DFP publications have new order numbers. Publications listed in the preface reflect these new order numbers.

Other minor technical and editorial changes have been made.

Preface

About This Book

This book is intended to help you customize MVS/DFP and provide exit routines and modules that extend or replace IBM-supplied function at your installation.

This book is divided into three sections:

1. "Introduction to Customization"

   This section contains guidance information on customization in MVS/DFP. Unless specifically stated otherwise, the information in this section must not be used for programming purposes. However, this section also provides the following type of information, which is explicitly identified where it occurs:

   Product-Sensitive Programming Interface

   Installation exits and other product-sensitive interfaces are provided to allow your installation to perform tasks such as product tailoring, monitoring, modification, or diagnosis. They are dependent on the detailed design or implementation of the product. Such interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

   End of Product-Sensitive Programming Interface

2. "MVS/DFP User Exits"

   This section provides guidance and reference information on user exits in MVS/DFP. It contains general-use programming interfaces, which allow you to write programs that use the services of MVS/DFP.

3. "MVS/DFP Installation Exits"

   This section provides guidance and reference information on installation exits in MVS/DFP. It contains product-sensitive programming interfaces provided by MVS/DFP. Installation exits and other product-sensitive interfaces are provided to allow your installation to perform tasks such as product tailoring, monitoring, modification, or diagnosis. They are dependent on the detailed design or implementation of the product. Such interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

The guidance information provided in this manual can be used by administrators who wish to centralize customization at their installations. The areas where MVS/DFP can be customized include all user exits, exit locations

and replaceable modules, and the Interactive Storage Management Facility (ISMF) display panels.

Required Product Knowledge

To use this book effectively, you should be familiar with:

- Assembler language
- Job control language
- Standard program linkage conventions
- Data management access methods and macro instructions
- Access method services commands
- VSAM macro instructions
- Interactive System Productivity Facility (ISPF) dialog manager

Required Publications

You should be familiar with the information presented in the following publications:

Publication Title	Order Number
Interactive System Productivity Facility Dialog Management Services	SC34-2137
MVS/ESA Catalog Administration Guide	SC26-4502
MVS/ESA Data Administration Guide	SC26-4505
MVS/ESA Data Administration: Macro Instruction Reference	SC26-4506
MVS/ESA Data Administration: Utilities	SC26-4516
MVS/ESA Integrated Catalog Adminstration: Access Method Services Reference	SC26-4500
MVS/ESA Interactive Storage Management Facility User's Guide	SC26-4508
MVS/ESA JCL Reference	GC28-1829
MVS/ESA JCL User's Guide	GC28-1830
MVS/ESA Magnetic Tape Labels and File Structure Administration	SC26-4511
MVS/ESA Message Library: System Messages Volume 1	GC28-1812
MVS/ESA Message Library: System Messages Volume 2	GC28-1813
MVS/ESA Storage Administration Reference	SC26-4514
MVS/ESA System—Data Administration	SC26-4515
MVS/ESA VSAM Administration Guide	SC26-4518
MVS/ESA VSAM Administration: Macro Instruction Reference	SC26-4517

Publication Title	Order Number
MVS/ESA VSAM Catalog Administration: Access Method Services Reference	SC26-4501

Related Publications

Some publications from the MVS/SP Version 3 library are referenced in this book. The *MVS/ESA Library Guide for System Product Version 3*, GC28-1563, contains a complete listing of the MVS/SP Version 3 publications and their counterparts for the prior version.

The *MVS/ESA Data Facility Product Version 3: Master Index*, GC26-4512, contains both an index to the MVS/DFP library and a summary of the changes made to the library. You can use it to:

- Find information in other MVS/DFP publications
- Determine how new programming support changes information in the MVS/DFP library
- Determine which MVS/DFP publications have been changed.

In addition, the following publications may be helpful:

Publication Title	Order Number
MVS/ESA Data Facility Product Version 3: Planning Guide	SC26-4513
MVS Storage Management Library: Configuring Storage Subsystems	SC26-4409

Referenced Publications

Within the text, references are made to the publications listed below:

Short Title	Publication Title	Order Number
Access Method Services Reference	*MVS/ESA Integrated Catalog Adminstration: Access Method Services Reference*	SC26-4500
	MVS/ESA VSAM Catalog Administration: Access Method Services Reference	SC26-4501
Application Development Guide	*MVS/ESA Application Development Guide*	GC28-1821
Application Development Macro Reference	*MVS/ESA Application Development Macro Reference*	GC28-1822
Basics of Problem Determination	*MVS/ESA Basics of Problem Determination*	GC28-1839

Short Title	Publication Title	Order Number
Checkpoint/Restart User's Guide	*MVS/ESA Checkpoint/Restart User's Guide*	SC26-4503
Data Administration Guide	*MVS/ESA Data Administration Guide*	SC26-4505
Data Administration: Macro Instruction Reference	*MVS/ESA Data Administration: Macro Instruction Reference*	SC26-4506
Using Dumps and Traces	*MVS/ESA Diagnosis: Using Dumps and Traces*	LY28-1843
DFDSS/ISMF: Installation Planning Guide	*Data Facility Data Set Services/Interactive Storage Management Facility: Installation Planning Guide*	SC26-4129
DFHSM: Installation and Customization Guide	*Data Facility Hierarchical Storage Manager: Version 2 Release 4 Installation and Customization Guide*	SH35-0084
DFP: Diagnosis Reference	*MVS/ESA Data Facility Product Version 3: Diagnosis Reference*	LY27-9551
DFSORT Installation and Customization	*DFSORT Installation and Customization*	SC33-4034
ICKDSF System Control Programming Specifications	*Device Support Facilities System Control Programming Specifications*	GC26-3946
Initialization and Tuning	*MVS/ESA System Programming Library: Initialization and Tuning*	GC28-1828
ISMF User's Guide	*MVS/ESA Interactive Storage Management Facility User's Guide*	SC26-4508
ISPF Dialog Management Services	*Interactive System Productivity Facility Dialog Management Services*	SC34-2137
Magnetic Tape Labels and File Structure	*MVS/ESA Magnetic Tape Labels and File Structure Administration*	SC26-4511
System Generation	*MVS/ESA System Generation*	GC28-1825
Dump and Trace Services	*MVS/ESA Planning: Dump and Trace Services*	GC28-1838
SPL: RACF	*Resource Access Control Facility (RACF) System Programming Library*	SC28-1343
Service Aids	*MVS/ESA System Programming Library: Service Aids*	LY28-1844
MVSCP	*MVS/ESA Component Diagnosis: MVSCP*	LY28-1852

Short Title	Publication Title	Order Number
SPL: Application Development Macro Reference	*MVS/ESA System Programming Library: Application Development Macro Reference*	GC28-1857
Storage Administration Reference	*MVS/ESA Storage Administration Reference*	SC26-4514
System—Data Administration	*MVS/ESA System—Data Administration*	SC26-4515
System Messages Volume 1	*MVS/ESA Message Library: System Messages Volume 1*	GC28-1812
System Messages Volume 2	*MVS/ESA Message Library: System Messages Volume 2*	GC28-1813
System Modifications	*MVS/ESA System Programming Library: System Modifications*	GC28-1831
TSO/E V2 Command Reference	*TSO/E Version 2 Command Reference*	SC28-1881
MVS/XA TSO Extensions TSO Command Language Reference	*MVS/Extended Architecture TSO Extensions Command Language Reference*	SC28-1134
Utilities	*MVS/ESA Data Administration: Utilities*	SC26-4516
User Exits	*MVS/ESA System Programming Library: User Exits*	GC28-1836
VSAM Administration: Macro Instruction Reference	*MVS/ESA VSAM Administration: Macro Instruction Reference*	SC26-4517

Introduction to Customization

About This Section

This section is intended to help you understand customization in MVS/DFP. It discusses the types of customization, and some advice on testing and documenting exit routines. Unless specifically stated otherwise, the information in this section must not be used for programming purposes. However, this section also provides the following type of information, which is explicitly identified where it occurs:

Product-Sensitive Programming Interface

Installation exits and other product-sensitive interfaces are provided to allow your installation to perform tasks such as product tailoring, monitoring, modification, or diagnosis. They are dependent on the detailed design or implementation of the product. Such interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

End of Product-Sensitive Programming Interface

This section contains:

Chapter 1. Introduction

What is Customization?

Customization consists of actions to enhance or extend a program to a greater extent than is provided by standard system-supplied options. MVS/ESA is an operating system that consists of MVS/SP, MVS/DFP, and other products. Both MVS/SP and MVS/DFP provide exit facilities for user customization.

Types of Customization

There are several types of customization:

- Your installation takes advantage of customization functions by supplying a new module to be installed as part of the system. Such modules fall into one of the following categories:
  - The module replaces an IBM-supplied module that performs no useful function except to give a return code. Such IBM-supplied modules are sometimes called dummy modules. Examples are the DADSM exit routines. If they are not replaced, no extra function is performed.
  - The module replaces an IBM-supplied module that already performs a useful function. An example is the data management abend installation exit (IFG0199I). If such modules are not replaced, they will perform certain functions.
  - IBM does not supply a module that performs the function. Examples are the nonstandard tape label processing modules. If they are not supplied, the function cannot be used.

  The modules described above must be reentrant and refreshable. They are installed during system installation by using the system modification program (SMP/E) or by link-editing the module into the appropriate library.

  This type of customization uses product-sensitive programming interfaces. They are dependent on the detailed design and implementation of the product. Such interfaces should be used only for product-tailoring, monitoring, modification, or diagnosis. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.
- The application programmer or system programmer changes certain messages and default values within the Interactive Storage Management Facility (ISMF). This type of customization uses general-use programming interfaces.
- The application program requests certain functions and supplies the exit routines to perform these functions. This type of customization uses general-use programming interfaces. Examples are the access method functions described in Chapter 4, "VSAM User-Written Exit Routines" on page 19 and Chapter 5, "Data Management User-Written Exit Routines" on page 41. The installation may supply standard modules to implement these functions but the individual application program must request the appropriate module. These modules do not have to be reentrant.

Customization in the MVS/ESA System

When installing the MVS/ESA system, initialization parameters provide a means of tailoring or tuning the system for your particular installation requirements. How you tune MVS/ESA may affect your customizing of MVS/SP and MVS/DFP. For more information about initialization parameters, see *Initialization and Tuning*.

User exits provided by MVS/SP are documented in *User Exits*.

Customization at a system level is also described in *System Modifications*.

Customization in MVS/DFP

Customization in MVS/DFP can be separated into two levels: one that affects the entire installation's processing and another that is limited to individual application program processing.

Installation Level Customization

Replacing a System-Level Module

By definition, a replaceable module is a system-level module you are allowed to change. Replaceable modules are product-sensitive programming interfaces, dependent on the detailed design and implementation of MVS/DFP. Your modifications can alter processing for your entire installation. If you choose to install system-wide processing changes, you must consider how processing affects all users of the MVS/DFP component affected.

Customizing ISMF

You can modify the form and content of the ISMF displays. Customizing ISMF can be a system-wide application. Changes you make to ISMF libraries affect all users of ISMF.

Application Program Customization

User exit locations provide a means of customizing MVS/DFP within an application program. User exits are general-use programming interfaces. User-written routines can be specific to one application, or can be standardized to be used in many of your application programs. To standardize exits used frequently, you can maintain a library of proven exits that can be used in application programs.

You can also customize ISMF displays for your own use. Other ISMF users would not be affected. Customizing ISMF this way would be limited to your individual applications.

Considerations in Deciding to Customize MVS/DFP

Why Customize?

Your installation may decide to customize MVS/DFP to:

- Enforce your installation standards
- Intercept errors for analysis and additional processing
- Add specialized tape label processing

- Tailor I/O processing
- Extend security controls
- Change or bypass processing.

When your installation decides what areas need customization to meet the requirements of your installation, you must consider the impact of your proposed modification. Is it going to be something that will affect all users of a component or function, or is it something that should be handled in the individual application program? Replacing system-level modules affects your entire installation. In conjunction with customizing MVS/DFP, you should examine the customization features available at the system level as briefly described in "Customization in the MVS/ESA System" on page 4.

Programming Considerations

Most requirements for coding vary depending on the part of MVS/DFP you are customizing. In general, be aware of the following:

- 31-bit addressing: You should refer to the individual exit routine descriptions. Some exits do not support this function.
- Use only documented interfaces identified as general-use programming interfaces or product-sensitive programming interfaces in the MVS/DFP library.
- Upon entering your exit routine, save all registers and restore them before returning to your calling routine. Register 15 is an exception. In many cases you must supply a return code in register 15 upon returning to your main program or MVS/DFP processing.
- If you replace a module, make sure you thoroughly test it before making it available to your installation.
- Your routine should be reentrant so that it is able to handle concurrent requests.
- Keep an unmodified copy of any replaceable modules or ISMF libraries you choose to modify.

Programming Languages

This document assumes you understand assembler language, ISPF dialog management language, and JCL. The examples are coded in assembler language and your routines may be coded in assembler language. ISMF examples use the ISPF dialog management language.

Restrictions and Limitations

MVS/DFP is a licensed program and can be modified for your own use only. IBM provides support and maintenance only for unmodified IBM-supplied modules and unmodified ISMF libraries.

Where Can You Customize in MVS/DFP?

User Exit Locations

In MVS/DFP, user exit locations are provided as part of macros and commands where you can specify the name and/or address of your user-written exit routine. The DCB macro, VSAM macros, and some access methods services commands contain parameters in which you specify the address or name of your exit routine. Some data set utility programs also provide user exit locations for modifying data set processing.

User exits are available at various points in data set processing such as:

- End-of-data
- I/O errors
- Logical errors
- Non-VSAM abend conditions
- Waiting for I/O completion
- At open, close, and end-of-volume.

The chapters describing user exits are:

- Chapter 4, "VSAM User-Written Exit Routines" on page 19
- Chapter 5, "Data Management User-Written Exit Routines" on page 41
- Chapter 6, "User Exit Routines Specified with Utilities" on page 83
- Chapter 7, "EXCP Appendages" on page 95

Available user exits are summarized in the general guidance sections of each chapter.

Replaceable Modules

In this manual, replaceable modules refers to IBM supplied modules you can modify or replace with your own. This category also applies to EXCP appendages, dummy modules and tape label processing modules.

Replaceable modules are available at various stages of processing such as:

- Before and after direct access device storage management (DADSM) processing
- At open for VSAM datestamp processing
- At open of a DCB
- At open, close, and end-of-volume abend conditions
- Before and after DASD calculation services
- I/O operations (appendages)
- At open, close and end-of-volume for additional tape label processing
- During automatic class selection

Replaceable modules are described in the following chapters:

- Chapter 7, "EXCP Appendages" on page 95
- Chapter 9, "Data Management Installation Exit Routines" on page 135
- Chapter 10, "Tape Label Processing Installation Exit Routines" on page 173
- Chapter 11, "Automatic Class Selection (ACS) Installation Exits" on page 219

A list of modules available is included in the guidance section of each chapter.

Tailoring ISMF

Because ISMF was partially written using the procedures described in *ISPF Dialog Management Services*, it can be modified using similar techniques. You can tailor ISMF panels, messages, job skeletons, command tables, and the CLIST. Customizing ISMF is described in Chapter 8, "Interactive Storage Management Facility (ISMF)" on page 105.

Chapter 2. Exit Testing Techniques

Several techniques can be employed to make your exit testing safer and easier. They include methods for protecting the system from errors in the exit, facilities for invoking dumps in order to get debugging information or to find out what information is available in system data areas, and ways to issue messages from the exit.

Protecting the System From Exit Errors

Three problems need to be addressed:

1. How to avoid the need for frequent IPLs during testing, since the exits reside in the Link Pack Area (LPA)
2. How to prevent overwriting of vital storage, since exits run in protect key zero
3. How to limit the scope of the exit so that testing can proceed with minimal impact on other work in the system

You can at least partially resolve these problems by:

1. Writing a "front end" to the exit and placing the "front end" in a modified LPA library
2. Placing the "real" exit code in another library such as SYS1.LINKLIB

If these two things are done, the "front end" can be a fairly innocuous bit of code which limits the scope of the exit by testing for specific jobnames, for example, and then gives control to the "real" exit code outside the LPA. The "front end," once coded and tested, is unlikely to need changes very often. The "real" exit is now in another library, where it can be changed without the need for an IPL in order to effect the change. This technique removes the need for reentrant code in the "real" exit during testing, since it will be loaded for each invocation. You will have to run additional tests later with the "real" exit in the LPA in order to test that exits are truly reentrant. Running an exit from outside the LPA is unlikely to be desirable except in a testing environment, since there is overhead involved in loading the exit each time it is entered.

Another safety feature of this way of testing exits is that use of the "front end" can be eliminated by an IPL without the MLPA parameter, just in case something is wrong with the "front end."

Once the exit is in production mode, protection against unexpected problems can also be implemented by having the exit check the contents of the CVTUSER field (CVT + 204 decimal). If the contents are zero (the normal case if CVTUSER is not being used by your installation), the exit should proceed. If not, it should return to the caller without taking further action (except to set register 15 to zero). When the exit is being used and an unexpected error is encountered, the contents of CVTUSER should be set to a nonzero value with console alter/display. This will cause the exit code to be temporarily disabled. Remember that a re-IPL will cause CVTUSER to become zero again, reactivating the exit.

Exit Testing Using Dumps

While testing your exit you are likely to need to invoke a dump at some time in order to debug or to examine data areas to determine where to look for information your exit requires. For information on analyzing dumps, see the book *Basics of Problem Determination*. For information on requesting and reading dumps, see the book *Dump and Trace Services*. The items below are advice you can follow when invoking dumps.

Issuing the ABEND Macro in an Exit

If an ABEND is issued explicitly from a preprocessing exit entered for Allocate, you will get message IEF197I SYSTEM ERROR DURING ALLOCATION. The job attempting the allocation will be failed with a JCL error and a dump will not be invoked. So issuing ABEND alone is not a good way of getting the information you need.

Setting CVTSDUMP

Product-Sensitive Programming Interface

The CVTSDUMP flag in the CVT can be set on to cause dumps to SYS1.DUMP to be invoken when ABEND is issued from a DADSM function (this includes the exits). This flag is at offset 272 in the CVT, and can be set on via the console alter/display functions. If you are testing under Virtual Machine/370, use the CP DISPLAY and STORE commands.

End of Product-Sensitive Programming Interface

Issuing the SDUMP Macro

Dumps can be invoked from exits via the SDUMP macro. As an alternative approach to using the CVTSDUMP procedure described above, this method eliminates the need to modify storage to cause the dump to be invoked. For information on the syntax and coding of the SDUMP macro, see *SPL: Application Development Macro Reference*. For information on the SVC dump that is produced by issuing the the SDUMP macro, see *Dump and Trace Services*.

Using the Console DUMP Command

By issuing a WTOR from the exit and letting the exit wait for the reply, you can suspend the exit's processing at any point and invoke a console dump to SYS1.DUMP using the DUMP operator command.

Issuing Messages

To check that your exit is functioning correctly, especially during the early stages of testing, you may want to issue messages giving the status of processing at that point. For example, you can issue a message early in the exit giving the reason for entry (Allocate, Extend, Scratch, Partial Release, or Rename). If you use WTO with a routecode of 11 (sometimes called 'Write to Programmer'), the message will appear on the output of the job that issued the DADSM request. Other possible uses for the messages are to indicate that certain data areas have been found successfully and to display selected contents of data areas.

Once the exit testing reaches the stage where large numbers of jobs are being handled by the exit, you should remove the code which produces these messages. Large numbers of messages will consume system message buffers, and the text will add to the user's output unnecessarily. Of course there will still be cases where exception messages may be required—these are discussed later in the section "Exit Messages" on page 14.

Chapter 3. User Interfaces

The way you design and implement the interface between your DADSM exits and the users of your system will have a big influence on the degree of acceptance you get. Remember that users may view the results of your hard work as a nuisance! You have probably introduced new restrictions on the way they do things, so you will have to "sell" the idea to them by pointing out how, for example, controlling space usage will protect them from themselves and each other. Once you have sold the users on the idea, you should make things as easy as possible for them by providing clear messages and good documentation.

Messages

There are three sources of messages associated with DADSM exits:

1. The system, due to errors or return codes produced by the exits
2. Programs which use DADSM functions, which may now get new return codes from DADSM
3. The exits themselves, which can issue messages directly

System Messages

Here are some of the messages which the system may issue when DADSM exits have been implemented:

- IEF197I SYSTEM ERROR DURING ALLOCATION—This message may appear if the exit abends while entered for an Allocate request.
- IEC223I with module IGGPRE00 indicated—This message may appear if a program check occurs in the exit during an Allocate request.
- IEF274I jjj sss ddn SPACE REQUEST REJECTED BY INSTALLATION EXIT, REASON CODE nnnn—This message is produced when the exit has rejected an Allocate request without allowing retry on other non-SMS-managed volumes (register 15 = 8, as set by the exit). The reason code is the code placed in the installation reject reason code field of the exit parameter list (IEXREASN) by the exit before returning to DADSM.
- IEF275I jjj sss ddn SPACE REQUEST CANNOT BE SATISFIED, INSTALLATION EXIT REASON CODE nnnn—This message is produced when the exit has rejected an Allocate request and allowed retry on other non-SMS-managed volumes, but the request could not be satisfied (register 15 = 4, as set by the exit). The reason code is the code placed in the installation reject reason code field of the exit parameter list (IEXREASN) by the exit before returning to DADSM.

Messages from Other Programs

Utility programs may provide in their messages nonzero return codes received from DADSM. Here is a summary of the new return codes associated with the use of the exits:

- Allocate
  - X'B0' - Installation exit rejected the request; no further volumes attempted
  - X'B4' - Installation exit rejected the request; try another volume
- Extend
  - -20 - Installation exit rejected the request (yes, this is a 'minus 20')
- Scratch
  - 4 - Installation exit rejected the request (in addition to the previous meanings for this return code)
- Partial Release
  - 16 - Installation exit rejected the request (in addition to the previous meanings for this return code)
- Rename
  - 4 - Installation exit rejected the request (in addition to the previous meanings for this return code)

Exit Messages

If the main purpose of your exit is to selectively reject Allocate requests, you should set installation reject reason codes in the exit parameter list and allow the IEF274I and IEF275I messages or the corresponding dynamic allocation reason codes (X'47B0' and X'47B4') to appear, rather than producing your own messages from the exit. The reason is that it may be better to have a message or code that is documented in a standard publication and then document the reject reason codes locally than to have a totally new message. At least the user can look up the IEF messages in the Messages manual and get some idea of why the job failed. If you do produce your own message, try to make the contents self-explanatory so that separate documentation is not necessary.

There may be some cases where it is a good idea to provide additional information. For example, if you use a RACF scheme that includes running space totals, you may want to produce a message containing the current running space total value when you reject a job which asked for space which would have made its total too high. Then the user has some basis on which to make a decision—perhaps to resubmit the job and ask for less space on that DD statement. If your installation has someone who monitors I/O-related messages, you may want to produce a warning message when a running space total is getting close to being exceeded. There is probably not much point in putting this message on the user's listing unless that user is the only one whose space is being accumulated against that identifier. A warning message sent to the appropriate routing code would alert a space manager that a space shortage problem may be imminent.

Documenting Your Exit For Users

The need to provide documentation for your users gives you an opportunity to publicize your space control policies. The following list contains some ideas on what should be included in a space control plan, and these same planning elements can be carried forward as headings for sections of your documentation:

1. The need for space control
2. Space usage standards
3. How usage standards will be enforced
4. Space conservation hints

In addition, you may want to have a separate section which summarizes the new messages and any installation reject reason codes that you have implemented.

Again—clear, convincing documentation is important! Without it the users may see your exits as an obstacle to getting their work done, instead of an effort to provide an equitable space management scheme which will ultimately benefit them.

MVS/DFP User Exits

About This Section

This section is intended to help you customize MVS/DFP by describing how to create user-written exit routines and tailor ISMF. It contains general-use programming interfaces, which allow you to write programs that use the services of MVS/DFP.

This section contains:

Chapter 4. VSAM User-Written Exit Routines

General Guidance

VSAM user-written routines may be supplied to:

- Analyze logical errors
- Analyze physical errors
- Perform end-of-data processing
- Record transactions made against a data set
- Perform special user processing
- Perform user-security verification.

VSAM user-written exit routines are identified by macro parameters in access methods services commands and in the EXLST VSAM macro.

You use the EXLST VSAM macro to create an exit list. EXLST parameters EODAD, JRNAD, LERAD, SYNAD and UPAD are used to specify the addresses of your user-written routines. Only the exits marked active are executed. For more information on the EXLST macro see *VSAM Administration: Macro Instruction Reference.*

You can use access methods services commands to specify the addresses of user-written routines to perform exception processing and user-security verification processing. For more information on exits from access methods services commands, see *Access Method Services Reference.*

The exit locations available from VSAM are outlined in the following table.

Figure 1 (Page 1 of 2). VSAM User-Written Exit Routines

Exit Routine	When Available	Where Specified
End-of-data-set	When no more sequential records or blocks are available	EODAD parameter of EXLST macro
Exception exit	After an uncorrectable input/output error	EXCEPTIONEXIT parameter on access methods services commands
Journalize transactions against a data set	After an input/output completion or error, change to buffer contents, shared or nonshared request, program issues GET, PUT, ERASE, shift in data in a control interval	JRNAD parameter of EXLST macro
Analyze logical errors	After an uncorrectable logical error	LERAD parameter of EXLST macro
Error analysis	After an uncorrectable input/output error	SYNAD parameter of EXLST macro

Figure 1 (Page 2 of 2). VSAM User-Written Exit Routines		
Exit Routine	**When Available**	**Where Specified**
User processing	WAIT for I/O completion or for a serially reusable request	UPAD parameter of EXLST macro
User security verification	When opening a VSAM data set	AUTHORIZATION parameter on access methods services commands

Programming Considerations

Information

To code VSAM user exit routines you should be familiar with the contents and have available the following MVS/DFP manuals:

VSAM Administration Guide

VSAM Administration: Macro Instruction Reference

Access Method Services Reference

Coding Guidance

In general, you should observe these guidelines in coding your routine:

- Code your routine reentrant
- Save and restore registers (see individual routines for other requirements)
- Be aware of registers used by the VSAM request macros
- Be aware of the addressing mode (24 bit or 31 bit) your exit routine will receive control in
- Determine if VSAM or your program should load the exit routine.

If the exit routine is used by a program that is doing asynchronous processing with multiple request parameter lists or, if the exit routine is used by more than one data set, it must be coded so that it can handle an entry made before the previous entry's processing is completed. Saving and restoring registers in the exit routine or by other routines called by the exit routine is best accomplished by coding the exit routine reentrant; another way is to develop a technique for associating a unique save area with each request parameter list (RPL).

If the LERAD, EODAD, or SYNAD exit routine reuses the RPL passed to it, you should be aware that:

- Recursion occurs (that is, the exit routine is called again) if the request that issues the reused RPL results in the same exception condition that caused the exit routine to be entered originally.

- The original feedback code is replaced with the feedback code that indicates the status of the latest request issued against the RPL. If the exit routine returns to VSAM, VSAM (when it returns to the user's program) sets register 15 to also indicate the status of the latest request.

A user exit that is loaded by VSAM will be invoked in the addressing mode specified when the module was link-edited. A user exit that is not loaded by VSAM will receive control in the same addressing mode as the caller of VSAM.

Your exit routine can be loaded within your program or by using the JOBLIB or STEPLIB with the DD statement to point to the library location of your exit routine.

Returning to Your Main Program

Five exit routines can be entered when your main program issues a VSAM request macro (GET, PUT, POINT, and ERASE) and the macro has not completed: LERAD, SYNAD, EODAD, UPAD, or the EXCEPTIONEXIT routine. Entering the LERAD, SYNAD, EODAD, or EXCEPTIONEXIT indicates that the macro failed to complete successfully. When your exit routine completes its processing, it can return to your main program in one of two ways:

1. The exit routine can return to VSAM (via the return address in register 14); VSAM then returns to your program at the instruction following the VSAM request macro that failed to complete successfully. This is the easier way to return to your program.
2. The exit routine can determine the appropriate return point in your program, then branch directly to that point. Note that when VSAM enters your exit routine, none of the registers contains the address of the instruction following the failing macro.

   You are required to use this method to return to your program if, during the error recovery and correction process, your exit routine issued a GET, PUT, POINT, or ERASE macro that refers to the RPL referred to by the failing VSAM macro. (That is, the RPL parameter list has been reissued by the exit routine.) In this case, VSAM has lost track of its reentry point to your main program. If the exit routine returns to VSAM, VSAM issues an error return code.

If your error recovery and correction process needs to reissue the failing VSAM macro against the RPL in order to retry the failing request or to correct it:

- Your exit routine can correct the RPL (using MODCB), then set a switch to indicate to your main program that the RPL is now ready to retry. When your exit routine completes processing, it can return to VSAM (via register 14), which returns to your main program. Your main program can then test the switch and reissue the VSAM macro and RPL.
- Your exit routine can issue a GENCB macro to build an RPL, and then copy the RPL (for the failing VSAM macro) into the newly built RPL. At this point, your exit routine can issue VSAM macros against the newly built RPL. When your exit routine completes processing, it can return to VSAM (via register 14), which returns to your main program.

EODAD Exit Routine to Process End-of-Data

Description

VSAM exits to an EODAD routine when an attempt is made to sequentially retrieve or point to a record beyond the last record in the data set (one with the highest key for keyed access and the one with the highest RBA for addressed access). VSAM doesn't take the exit for direct requests that specify a record beyond the end. If the EODAD exit isn't used, the condition is considered a logical error (FDBK code X'04') and can be handled by the LERAD routine, if one is supplied (see "LERAD Exit Routine to Analyze Logical Errors" on page 32).

Register Contents

Figure 2 gives the contents of the registers when VSAM exits to the EODAD routine.

Register	Contents
0	Unpredictable.
1	Address of the RPL that defines the request that occasioned VSAM's reaching the end of the data set. The register must contain this address if you return to VSAM.
2-13	Unpredictable. Register 13, by convention, contains the address of your processing program's 72-byte save area, which must not be used as a save area by the EODAD routine if it returns control to VSAM.
14	Return address to VSAM.
15	Entry address to the EODAD routine.

Figure 2. Contents of Registers at Entry to EODAD Exit Routine

Programming Considerations

The typical actions of an EODAD routine are to:

- Examine RPL for information you need, for example, type of data set
- Issue completion messages
- Close the data set
- Terminate processing without returning to VSAM.

If the routine returns to VSAM and another GET request is issued for access to the data set, VSAM exits to the LERAD routine.

If a processing program retrieves records sequentially with a request defined by a chain of RPLs, the EODAD routine must determine whether the end of the data set was reached for the first RPL in the chain. If not, then one or more records have been retrieved but not yet processed by the processing program.

The type of data set whose end was reached can be determined by examining the RPL for the address of the access method control block that connects the program to the data set and testing its attribute characteristics.

If the exit routine issues GENCB, MODCB, SHOWCB, or TESTCB and returns to VSAM, it must provide a save area and restore registers 13 and 14, which are used by these macros.

When your EODAD routine completes processing, return to your main program as described in "Returning to Your Main Program" on page 21.

EXCEPTIONEXIT Exit Routine

Description

You can provide an exception exit routine to monitor I/O errors associated with a data set. You specify the name of your routine via the access method services DEFINE command using the EXCEPTIONEXIT parameter to specify the name of your user-written exit routine.

Register Contents

Figure 3 gives the contents of the registers when VSAM exits to the EXCEPTIONEXIT.

Register	Contents
0	Unpredictable.
1	Address of the RPL that contains a feedback return code and the address of a message area, if any.
2-13	Unpredictable. Register 13, by convention, contains the address of your processing program's 72-byte save area, which must not be used by the routine if it returns control to VSAM.
14	Return address to VSAM.
15	Entry address to the exception exit routine.

Figure 3. Contents of Registers at Entry to EXCEPTIONEXIT Routine

Programming Considerations

The exception exit is taken for the same errors as a SYNAD exit. If you have both an active SYNAD routine and an EXCEPTIONEXIT routine, the exception exit routine is processed first.

The exception exit is associated with the attributes of the data set (specified by the DEFINE) and is loaded on every call. Your exit must reside in the LINKLIB and the exit cannot be called when VSAM is in cross-memory mode.

When your exception exit routine completes processing, return to your main program as described in "Returning to Your Main Program" on page 21.

For information about how exception exits are established, changed, or nullified, see *Access Method Services Reference*.

JRNAD Exit Routine to Journalize Transactions

Description

A JRNAD exit routine can be provided to record transactions against a data set, to keep track of changes in the RBAs of records, and to monitor control interval splits. It is only available for VSAM shared resource buffering. For shared resources, you can use a JRNAD exit routine to deny a request for a control interval split. VSAM takes the JRNAD exit each time one of the following occurs:

- The processing program issues a GET, PUT, or ERASE
- Data is shifted right or left in a control interval or is moved to another control interval to accommodate a record's being deleted, inserted, shortened, or lengthened
- An I/O error occurs
- An I/O completion occurs
- A shared or nonshared request is received
- The buffer contents are to be changed.

Register Contents

Figure 4 gives the contents of the registers when VSAM exits to the JRNAD routine.

Register	Contents
0	Unpredictable.
1	Address of a parameter list built by VSAM.
2-13	Unpredictable.
14	Return address to VSAM.
15	Entry address to the JRNAD routine.

Figure 4. Contents of Registers at Entry to JRNAD Exit Routine

Programming Considerations

If the JRNAD is taken for I/O errors, a journal exit may zero out, or otherwise alter, the physical-error return code, so that a series of operations may continue to completion, even though one or more of the operations failed.

The contents of the parameter list built by VSAM, pointed to by register 1, can be examined by the JRNAD exit routine which is described in Figure 6 on page 28.

If the exit routine issues GENCB, MODCB, SHOWCB, or TESTCB, it must restore register 14, which is used by these macros, before it returns to VSAM.

If the exit routine uses register 1, it must restore it with the parameter list address before returning to VSAM. (The routine must return for completion of the request that caused VSAM to exit.)

The JRNAD exit must be indicated as active before the data set for which the exit is to be used is opened, and the exit must not be made inactive during processing. If you define more than one access method control block for a data set and want to have a JRNAD routine, the first ACB you open for the data set must specify the exit list that identifies the routine.

Journalizing Transactions

For journalizing transactions (when VSAM exits because of a GET, PUT, or ERASE), you can use the SHOWCB macro to display information in the request parameter list about the record that was retrieved, stored, or deleted (FIELDS=(AREA,KEYLEN,RBA,RECLEN), for example). You can also use the TESTCB macro to find out whether a GET or a PUT was for update (OPTCD=UPD).

If your JRNAD routine only journals transactions, it should ignore reason X'0C' and return to VSAM; conversely, it should ignore reasons X'00', X'04', and X'08' if it records only RBA changes.

Recording RBA Changes

For recording RBA changes, you must calculate how many records there are in the data being shifted or moved, so you can keep track of the new RBA for each. If all the records are the same length, you calculate the number by dividing the record length into the number of bytes of data being shifted. If record length varies, you can calculate the number by using a table that not only identifies the records (by associating a record's key with its RBA), but also gives their length.

You should provide a routine to keep track of RBA changes caused by control interval and control area splits. RBA changes that occur by way of keyed access to a key-sequenced data set must also be recorded if you intend to process the data set later by direct-addressed access.

Control Interval Splits

Some control interval splits involve data being moved to two new control intervals, and control area splits normally involve many control intervals' contents being moved. In these cases, VSAM exits to the JRNAD routine for each separate movement of data to a new control interval.

You may also want to use the JRNAD exit to maintain shared or exclusive control over certain data or index control intervals; and in some cases, in your exit routine you may reject the request for certain processing of the control intervals. For example, if you used this exit to maintain information about a data set in a shared environment, you might reject a request for a control

interval or control area split because the split might adversely affect other users of the data set.

Figure 5 is a skeleton program USERPROG with a user exit routine USEREXIT. It demonstrates the use of the JRNAD exit routine to cancel a request for a control interval or control area split.

```
USERPROG  CSECT
          SAVE(R14,R12)        Standard entry code
          .
          .
          .
          BLDVRP BUFFERS=(512(3)),  Build resource pool            X
                KEYLEN=4,                                          X
                STRNO=4,                                           X
                TYPE=LSR,                                          X
                SHRPOOL=1,                                         X
                RMODE31=ALL
          OPEN (DIRACB)        Logically connect KSDS1
          .
          .
          .
          PUT RPL=DIRRPL       This PUT causes the exit routine USEREXIT
                               to be taken with an exit code X'50' if
                               there is a CI or CA split
          LTR R15,R15          Check return code from PUT
          BZ NOCANCEL          Retcode = 0 if USEREXIT did not cancel
                                           CI/CA split
                                       = 8 if cancel was issued, assuming
                                           that we know a CI or CA split
                                           occurred
          .
          .                    Process the cancel situation
          .
NOCANCEL  .                    Process the noncancel situation
          .
          .
          CLOSE (DIRACB)       Disconnect KSDS1
          DLVRP TYPE=LSR,SHRPOOL=1  Delete the resource pool
          .
          .
          .
          RETURN               Return to caller.
          .
          .
          .
DIRACB    ACB AM=VSAM,                                             X
          DDNAME=KSDS1,                                            X
          BUFND=3,                                                 X
          BUFNI=2,                                                 X
          MACRF=(KEY,DDN,SEQ,DIR,OUT,LSR),                         X
          SHRPOOL=1,                                               X
          EXLST=EXITLST
```

Figure 5 (Part 1 of 2). Example of a JRNAD Exit

```
*
DIRRPL   RPL AM=VSAM,                                              X
         ACB=DIRACB,                                               X
         AREA=DATAREC,                                             X
         AREALEN=128,                                              X
         ARG=KEYNO,                                                X
         KEYLEN=4,                                                 X
         OPTCD=(KEY,DIR,FWD,SYN,NUP,WAITX),                        X
         RECLEN=128
*
DATAREC  DC  CL128'DATA RECORD TO BE PUT TO KSDS1'
KEYNO    DC  F'0'            Search key argument for RPL
EXITLST  EXLST AM=VSAM,JRNAD=(JRNADDR,A,L)
JRNADDR  DC  CL8'USEREXIT'   Name of user exit routine
         END                 End of USERPROG


USEREXIT CSECT               On entry to this exit routine, R1 points
                             to the JRNAD parameter list and R14 points
                             back to VSAM.
         .
         .                   Nonstandard entry code -- need not save
         .                   the registers at caller's save area and,
         .                   since user exit routines are reentrant for
         .                   most applications, save R1 and R14 at some
         .                   registers only if R1 and R14 are to be
         .                   destroyed
         .
         CLI 20(R1),X'50'    USEREXIT called because of CI/CA split?
         BNE EXIT            No.  Return to VSAM
         MVI 21(R1),X'8C'    Tell VSAM that user wants to cancel split
         .
         .
         .
EXIT     .                   Nonstandard exit code -- restore R1 and
         .                   R14 from save registers
         BR R14              Return to VSAM which returns to USERPROG
                             if cancel is specified
         END                 End of USEREXIT
```

Figure 5 (Part 2 of 2). Example of a JRNAD Exit

Parameter List

The parameter list built by VSAM contains reason codes to indicate why the exit was taken, and also locations where you can specify return codes for VSAM to take or not take an action upon returning from your routine. The information provided in the parameter list varies depending on the reason the exit was taken. Figure 6 shows the contents of the parameter list.

The parameter list will reside in the same area as the VSAM control blocks, either above or below the 16M line. For example, if the VSAM data set was opened and the ACB stated RMODE31=CB, the exit parameter list will reside above the 16M line. To access a parameter list that resides above the 16M line, you will need to use 31-bit addressing.

Figure 6 (Page 1 of 4). Contents of Parameter List Built by VSAM for the JRNAD Exit

Offset	Bytes	Description
0(X'0')	4	Address of the RPL that defines the request that caused VSAM to exit to the routine.
4(X'4')	4	Address of a 5-byte field that identifies the data set being processed. This field has the format:
		4 bytes — Address of the access method control block specified by the RPL that defines the request occasioned by the JRNAD exit.
		1 byte — Indication of whether the data set is the data (X'01') or the index (X'02') component.
8(X'8')	4	Variable, depends on the reason indicator at offset 20:
		Offset 20 — **Contents at offset 8**
		X'0C' — The RBA of the first byte of data that is being shifted or moved.
		X'20' — The RBA of the beginning of the control area about to be split.
		X'24' — The address of the I/O buffer into which data was going to be read.
		X'28' — The address of the I/O buffer from which data was going to be written.
		X'2C' — The address of the I/O buffer that contains the control interval contents that are about to be written.
		X'30' — Address of the buffer control block (BUFC) that points to the buffer into which data is about to be read under exclusive control.
		X'34' — Address of BUFC that points to the buffer into which data is about to be read under shared control.
		X'38' — Address of BUFC that points to the buffer which is to be acquired in exclusive control. The buffer is already in the buffer pool.
		X'3C' — Address of the BUFC that points to the buffer which is to be built in the buffer pool in exclusive control.
		X'40' — Address of BUFC which points to the buffer whose exclusive control has just been released.
		X'44' — Address of BUFC which points to the buffer whose contents have been made invalid.
		X'48' — Address of the BUFC which points to the buffer into which the READ operation has just been completed.
		X'4C' — Address of the BUFC which points to the buffer from which the WRITE operation has just been completed.

Figure 6 (Page 2 of 4). Contents of Parameter List Built by VSAM for the JRNAD Exit

Offset	Bytes	Description	
12(X'C')	4	Variable, depends on the reason indicator at offset 20:	
		Offset 20	**Contents at offset 12**
		X'0C'	The number of bytes of data that is being shifted or moved (this number doesn't include free space, if any, or control information—except for a control area split, when the whole contents of a control interval are moved to a new control interval.)
		X'20'	Unpredictable.
		X'24'	Unpredictable.
		X'28'	Bits 0 through 31 correspond with transaction IDs 0 through 31. Bits set to 1 indicate that the buffer that was being written when the error occurred was modified by the corresponding transactions. You can set additional bits to 1 to tell VSAM to keep the contents of the buffer until the corresponding transactions have modified the buffer.
		X'2C'	The size of the control interval whose contents are about to be written.
		X'30'	Size of the buffer into which data is about to be read under exclusive control.
		X'34'	Size of the buffer which is about to be read into shared status.
		X'38'	Size of the buffer which is to be acquired in exclusive control. The buffer is already in the buffer pool.
		X'3C'	Size of the buffer which is to be built in the buffer pool in exclusive control.
		X'48'	Size of the buffer into which the READ operation has just been completed.
		X'4C'	Size of the buffer from which the WRITE operation has just been completed.

Figure 6 (Page 3 of 4). Contents of Parameter List Built by VSAM for the JRNAD Exit

Offset	Bytes	Description	
16(X'10')	4	Variable, depends on the reason indicator at offset 20:	
		Offset 20	**Contents at offset 16**
		X'0C'	The RBA of the first byte to which data is being shifted or moved.
		X'20'	The RBA of the last byte in the control area about to be split.
		X'24'	The fourth byte contains the physical error code from the RPL FDBK field. You use this fullword to communicate with VSAM. Setting it to 0 indicates that VSAM is to ignore the error, bypass error processing, and let the processing program continue. Leaving it nonzero indicates that VSAM is to continue as usual: terminate the request that occasioned the error and proceed with error processing, including exiting to a physical error analysis routine.
		X'28'	Same as for X'24'.
		X'2C'	The RBA of the control interval whose contents are about to be written.
		X'48'	The RBA of the control interval into which the READ operation has just been completed.
		X'4C'	The RBA of the control interval from which the WRITE operation has just been completed.

Figure 6 (Page 4 of 4). Contents of Parameter List Built by VSAM for the JRNAD Exit

Offset	Bytes	Description	
20(X'14')	1	Indication of the reason VSAM exited to the JRNAD routine:	
		X'00'	GET request.
		X'04'	PUT request.
		X'08'	ERASE request.
		X'0C'	RBA change.
		X'10'	Read spanned record segment.
		X'14'	Write spanned record segment.
		X'18'	Reserved.
		X'1C'	Reserved.
		The following codes are for shared resources only:	
		X'20'	Control area split.
		X'24'	Input error.
		X'28'	Output error.
		X'2C'	Buffer write.
		X'30'	A data or index control interval is about to be read in exclusive control.
		X'34'	A data or index control interval is about to be read in shared status.
		X'38'	Acquire exclusive control of a control interval already in the buffer pool.
		X'3C'	Build a new control interval for the data set and hold it in exclusive control.
		X'40'	Exclusive control of the indicated control interval already has been released.
		X'44'	Contents of the indicated control interval have been made invalid.
		X'48'	Read completed.
		X'4C'	Write completed.
		X'50'	Control interval or control area split.
		X'54'—X'FF'	Reserved.
21(X'15')	1	JRNAD exit code set by the JRNAD exit routine. Indication of action to be taken by VSAM after resuming control from JRNAD (for shared resources only):	
		X'80'	Do not write control interval.
		X'84'	Treat I/O error as no error.
		X'88'	Do not read control interval.
		X'8C'	Cancel the request for control interval or control area split.

LERAD Exit Routine to Analyze Logical Errors

Description

A LERAD exit routine should examine the feedback field in the request parameter list to determine what logical error occurred. What the routine does after determining the error depends on your knowledge of the kinds of things in the processing program that may have caused the error.

Register Contents

Figure 7 gives the contents of the registers when VSAM exits to the LERAD exit routine.

Note: A LERAD exit is not taken for RPLFDBK 64(40) because a PLH is not available for register saving.

Register	Contents
0	Unpredictable.
1	Address of the RPL that contains the feedback field the routine should examine. The register must contain this address if you return to VSAM.
2-13	Unpredictable. Register 13, by convention, contains the address of your processing program's 72-byte save area, which must not be used as a save area by the LERAD routine if the routine returns control to VSAM.
14	Return address to VSAM.
15	Entry address to the LERAD routine. The register doesn't contain the logical-error indicator.

Figure 7. Contents of Registers at Entry to LERAD Exit Routine

Programming Considerations

The typical actions of a LERAD routine are:

1. Examine the feedback field in the RPL to determine what error occurred
2. Determine what action to take based on error
3. Close the data set
4. Issue completion messages
5. Terminate processing and exit VSAM or return to VSAM.

If the LERAD exit routine issues GENCB, MODCB, SHOWCB, or TESTCB and returns to VSAM, it must restore registers 1, 13, and 14, which are used by these macros. It must also provide two save areas; one, whose address should be loaded into register 13 before the GENCB, MODCB, SHOWCB, or TESTCB is issued, and the second, to separately store registers 1, 13, and 14.

If the error cannot be corrected, close the data set and either terminate processing or return to VSAM.

If a logical error occurs and no LERAD exit routine is provided (or the LERAD exit is inactive), VSAM returns codes in register 15 and in the feedback field of the RPL to identify the error.

When your LERAD exit routine completes processing, return to your main program as described in "Returning to Your Main Program" on page 21.

SYNAD Exit Routine to Analyze Physical Errors

Description

VSAM exits to a SYNAD routine if a physical error occurs when you request access to data. It also exits to a SYNAD routine when you close a data set if a physical error occurs while VSAM is writing the contents of a buffer out to direct-access storage.

Register Contents

Figure 8 gives the contents of the registers when VSAM exits to the SYNAD routine.

Register	Contents
0	Unpredictable.
1	Address of the RPL that contains a feedback return code and the address of a message area, if any. If you issued a request macro, the RPL is the one pointed to by the macro; if you issued an OPEN, CLOSE, or cause an end-of-volume to be done, the RPL was built by VSAM to process an internal request. Register 1 must contain this address if the SYNAD routine returns to VSAM.
2-13	Unpredictable. Register 13, by convention, contains the address of your processing program's 72-byte save area, which must not be used by the SYNAD routine if it returns control to VSAM.
14	Return address to VSAM.
15	Entry address to the SYNAD routine.

Figure 8. Contents of Registers at Entry to SYNAD Exit Routine

Programming Considerations

A SYNAD routine should typically:

- Examine the feedback field in the request parameter list to identify the type of physical error that occurred.
- Get the address of the message area, if any, from the request parameter list, to examine the message for detailed information about the error
- Recover data if possible
- Print error messages if uncorrectable error
- Close data set
- Terminate processing

The main problem with a physical error is the possible loss of data. You should try to recover your data before continuing to process. Input operations (ACB MACRF=IN) are generally less serious than output or update operations (MACRF=OUT), because your request was not attempting to alter the contents of the data set.

If the routine cannot correct an error, it might print the physical-error message, close the data set, and terminate the program. If the error occurred while VSAM was closing the data set, and if another error occurs after the exit routine issues a CLOSE macro, VSAM doesn't exit to the routine a second time.

If the SYNAD routine returns to VSAM, whether the error was corrected or not, VSAM drops the request and returns to your processing program at the instruction following the last executed instruction. Register 15 is reset to indicate that there was an error, and the feedback field in the RPL identifies it.

Physical errors affect positioning. If a GET was issued that would have positioned VSAM for a subsequent sequential GET and an error occurs, VSAM is positioned at the control interval next in key (RPL OPTCD=KEY) or in entry (OPTCD=ADR) sequence after the control interval involved in the error. The processing program can therefore ignore the error and proceed with sequential processing. With direct processing, the likelihood of reencountering the control interval involved in the error depends on your application.

If the exit routine issues GENCB, MODCB, SHOWCB, or TESTCB and returns to VSAM, it must provide a save area and restore registers 13 and 14, which are used by these macros.

See "Example of a SYNAD User-Written Exit Routine" for the format of a physical-error message that can be written by the SYNAD routine.

When your SYNAD exit routine completes processing, return to your main program as described in "Returning to Your Main Program" on page 21.

If a physical error occurs and no SYNAD routine is provided (or the SYNAD exit is inactive), VSAM returns codes in register 15 and in the feedback field of the RPL to identify the error. For a description of these return codes, see *VSAM Administration: Macro Instruction Reference*.

Example of a SYNAD User-Written Exit Routine

The example in Figure 9 on page 35 demonstrates a user-written exit routine. It is a SYNAD exit routine that examines the FDBK field of the RPL checking for the type of physical error that caused the exit. After the checking, special processing may be performed as necessary. The routine returns to VSAM after printing an appropriate error message on SYSPRINT.

```
ACB1     ACB    EXLST=EXITS

EXITS    EXLST  SYNAD=PHYERR

RPL1     RPL    ACB=ACB1,
                MSGAREA=PERRMSG,
                MSGLEN=128

PHYERR   USING  *,15            This routine is nonreentrant.

*                               Register 15 is entry address.

         .
         .                      Save caller's register
                                (1, 13, 14).
         .
         LA     13,SAVE         Point to routine's save area.
         .
         .                      If register 1=address of RPL1,
         .                      then error did not occur for a
                                CLOSE.
         SHOWCB RPL=RPL1,
                FIELDS=FDBK,
                AREA=ERRCODE,
                LENGTH=4

*                               Show type of physical error.

         .
         .                      Examine error, perform special
         .                      processing.

         PUT    PRTDCB,ERRMSG   Print physical error message.
         .
         .                      Restore caller's registers
         .                      (1, 13, 14).

         BR     14              Return to VSAM.
         .
         .
         .
ERRCODE  DC     F'0'            RPL reason code from SHOWCB.
```

Figure 9 (Part 1 of 2). Example of a SYNAD Exit Routine

```
PERRMSG  DS    0XL128          Physical error message.
         DS    XL12            Pad for unprintable part.
ERRMSG   DS    XL116           Printable format part of
                               message.
         .
         .
         .
PRTDCB   DCB   .....           QSAM DCB.
SAVE     DS    18F             SYNAD routine's save area.
SAVREG   DS    3F              Save registers 1, 13, 14.
```

Figure 9 (Part 2 of 2). Example of a SYNAD Exit Routine

UPAD Exit Routine for User Processing

Description

You can perform special processing during a VSAM request with the UPAD exit routine. For example, VSAM takes the UPAD exit immediately prior to issuing a WAIT for I/O completion or for a serially reusable resource. VSAM exits to the UPAD routine when the request's RPL specifies OPTCD=(SYN, WAITX) and the ACB specifies MACRF=LSR or MACRF=GSR, or MACRF=ICI.

If you are executing in cross-memory mode, you must have a UPAD routine. Cross-memory mode is described in *VSAM Administration Guide*.

Register Contents

Figure 10 shows the register contents passed by VSAM when the UPAD exit routine is entered.

Register	Contents
0	Unpredictable.
1	Address of a parameter list built by VSAM.
2-12	Unpredictable.
13	Reserved.
14	Return address to VSAM.
15	Entry address of the UPAD routine.

Figure 10. Contents of Registers at Entry to UPAD Exit Routine

Programming Considerations

The UPAD exit routine must be active before the data set is opened. The exit must not be made inactive during processing. If the UPAD exit is desired and many ACBs are used for processing the data set, the first ACB that is opened must specify the exit list that identifies the UPAD exit routine.

The contents of the parameter list built by VSAM, pointed to by register 1, can be examined by the UPAD exit routine (see Figure 11).

Figure 11. Parameter List Passed to UPAD Routine

Offset	Bytes	Description	
0(X'0')	4	Address of the RPL.	
4(X'4')	4	Address of a 5-byte data set identifier. The first four bytes of the identifier are the ACB address; the last byte identifies the component; data (X'01'), or index (X'02').	
8(X'8')	4	Address of the request's ECB.	
12(X'0C')	4	Post flag or cross-memory action flag (see cross-memory mode).	
16(X'10')	4	Reserved.	
20(X'14')	1	Reason code:	
		X'00'	VSAM is about to wait.
		X'04'	VSAM ready to resume request processing.
		X'08'—X'FC'	Reserved.

If the UPAD exit routine modifies register 14 (for example, by issuing a TESTCB), the routine must restore register 14 before returning to VSAM. If register 1 is used, the UPAD exit routine must restore it with the parameter list address before returning to VSAM.

The UPAD routine must return to VSAM under the same TCB from which it was called for completion of the request that caused VSAM to exit. The UPAD exit routine cannot use register 13 as a save area pointer without first obtaining its own save area.

The UPAD exit routine, when taken prior to a WAIT during LSR or GSR processing, might issue other VSAM requests to obtain better processing overlap (similar to asynchronous processing). However, the UPAD routine must not issue any synchronous VSAM requests that do not specify WAITX, because a started request might issue a WAIT for a resource owned by a starting request.

If the UPAD routine starts requests that specify WAITX, the UPAD routine must be reentrant. After multiple requests have been started, they should be synchronized by waiting for one ECB out of a group of ECBs to be posted complete rather than waiting for a specific ECB or for many ECBs to be posted complete. (Posting of some ECBs in the list might be dependent upon the resumption of some of the other requests that entered the UPAD routine.)

If you are not in cross-memory mode and the UPAD routine returns with a nonzero code, VSAM will cause a POST to be issued.

Cross-Memory Mode

If you are executing in cross-memory mode, you must have a UPAD routine. When posting of an event is required, the UPAD routine is given control (reason code 4).

When VSAM regains control from a UPAD exit that was taken for reason code 4, VSAM tests the return code at offset 12 in the parameter list. If it is nonzero and the request is in cross-memory mode, VSAM indicates a logical error rather than attempting to issue a POST. (POST would cause an abend if issued in cross-memory mode.)

Your UPAD routine must resume the request that caused the exit to be taken and set the appropriate return code in the parameter list before returning to VSAM.

User-Security-Verification Routine (USVR)

If you use VSAM password protection, you may also have your own routine to check a requester's authority. Your routine is invoked from OPEN, rather than via an exit list. VSAM transfers control to your routine, which must reside in SYS1.LINKLIB, when a requester gives a correct password other than the master password.

Note: You may use VSAM password protection, but it is not recommended. Protection provided by RACF or an equivalent product is recommended instead.

Through the access method services DEFINE command with the AUTHORIZATION parameter you may identify your user-security-verification routine (USVR) and associate as many as 256 bytes of your own security information with each data set to be protected. The user security-authorization record (USAR) is made available to the USVR when the routine gets control. You may restrict access to the data set as you choose; for example, you may require that the owner of a data set give ID when defining the data set and then allow only the owner to gain access to the data set.

If the USVR is being used by more than one task at a time, you must code the USVR reentrant or develop another method for handling simultaneous entries.

When your USVR completes processing, it must return (in register 15) to VSAM with a return code of 0 for authority granted or not 0 for authority withheld in register 15.

Figure 12 on page 39 gives the contents of the registers when VSAM gives control to the USVR.

Register	Contents	
0	Unpredictable.	
1	Address of a parameter list with the following format:	
	44 bytes	Name of the data set for which authority to process is to be verified (the name you specified when you defined it with access method services).
	8 bytes	Prompting code (or 0's).
	8 bytes	Owner identification (or 0's).
	8 bytes	The password that the requester gave (it has been verified by VSAM).
	2 bytes	Length of the user-security-authorization routine (in binary).
	—	The user-security-authorization.
2-13	Unpredictable.	
14	Return address to VSAM.	
15	Entry address to the USVR. When the routine returns to VSAM, it indicates by the following codes in register 15 whether the requester has been authorized to gain access to the data set:	
	0	Authority granted.
	not 0	Authority withheld.

Figure 12. Communication with User-Security-Verification Routine

Chapter 5. Data Management User-Written Exit Routines

General Guidance

Non-VSAM macros can be used to identify user-written exit routines. These user-written exit routines can perform a variety of functions for non-VSAM data sets, including error analysis, requesting user totaling, and creating your own data set labels. These functions are not for use with VSAM data sets.

The DCB macro can be used to identify the location of:

- A routine that performs end-of-data procedures
- A routine that supplements the operating system's error recovery routine
- A list that contains addresses of special exit routines.

The exit addresses can be specified in the DCB macro or you can complete the DCB fields before opening the data set. Figure 13 summarizes the exits that you can specify either explicitly in the DCB, or implicitly by specifying the address of an exit list in the DCB.

Figure 13 (Page 1 of 2). DCB Exit Routines

Exit Routine	When Available	Where Specified
End-of-data-set	When no more sequential records or blocks are available	EODAD parameter
Error analysis	After an uncorrectable input/output error	SYNAD parameter
Allocation retrieval list	When issuing an RDJFCB macro instruction	EXLST parameter and exit list
Block count	After unequal block count comparison by end-of-volume routine	EXLST parameter and exit list
DCB abend	When an abend condition occurs in OPEN, CLOSE, or end-of-volume routine	EXLST parameter and exit list
DCB open	When opening a data set	EXLST parameter and exit list
End-of-volume	When changing volumes	EXLST parameter and exit list
FCB image	When opening a data set or issuing a SETPRT macro	EXLST parameter and exit list
JFCB	When opening a data set with TYPE=J and reading the JFCB	EXLST parameter and exit list

Figure 13 (Page 2 of 2). DCB Exit Routines		
Exit Routine	**When Available**	**Where Specified**
Standard user label (physical sequential or direct organization)	When opening, closing, or reaching the end of a data set, and when changing volumes	EXLST parameter and exit list
JFCB extension (JFCBE)	When opening a data set for the IBM 3800	EXLST parameter and exit list
Open/EOV nonspecific tape volume mount	When a scratch tape is requested during OPEN or EOV routines	EXLST parameter and exit list
Open/EOV volume security/verification	When a scratch tape is requested during OPEN or EOV routines	EXLST parameter and exit list
QSAM parallel processing	Opening a data set	EXLST parameter and exit list
User totaling (for BSAM and QSAM)	When creating or processing a data set with user labels	EXLST parameter and exit list

Programming Considerations

Because OPEN/CLOSE/EOV enqueues on SYSZTIOT, functions that require SYSZTIOT cannot be executed in the OPEN/CLOSE/EOV exit routines. Some of these functions are LOCATE, OBTAIN, SCRATCH, CATALOG, and so forth.

Status Information Following an Input/Output Operation

Following an input/output operation with a DCB, the control program makes certain status information available to the problem program. This information is a 2-byte exception code, or a 16-byte field of standard status indicators, or both.

Exception codes are provided in the data control block (QISAM), or in the data event control block (BISAM and BDAM). The data event control block is described below, and the exception code lies within the block as shown in the illustration for the data event control block. If a DCBD macro instruction is coded, the exception code in a data control block can be addressed as two 1-byte fields, DCBEXCD1 and DCBEXCD2. The exception codes can be interpreted by referring to Figure 15, Figure 16, and Figure 17.

Status indicators are available only to the error analysis routine designated by the SYNAD entry in the data control block. A pointer to the status indicators is provided either in the data event control block (BSAM, BPAM, and BDAM), or in register 0 (QISAM and QSAM). The contents of registers on entry to the SYNAD exit routine are shown in Figure 18 on page 49, Figure 19 on page 50, and Figure 20 on page 51; the status indicators are shown in Figure 21 on page 52.

Data Event Control Block

A data event control block is constructed as part of the expansion of READ and WRITE macro instructions and is used to pass parameters to the control program, help control the read or write operation, and receive indications of the success or failure of the operation. The data event control block is named by the READ or WRITE macro instruction, begins on a fullword boundary, and contains the information shown in Figure 14.

Figure 14. Data Event Control Block

Offset from DECB Address (Bytes)	Field Contents BSAM and BPAM	BISAM	BDAM
0	ECB	ECB	ECB[1]
+4	Type	Type	Type
+6	Length	Length	Length
+8	DCB address	DCB address	DCB address
+12	Area address	Area address	Area address
+16	IOB address	Logical record address	IOB address
+20		Key address	Key address
+24		Exception code (2 bytes)	Block address
+28			Next address

[1] The control program returns exception codes in bytes +1 and +2 of the ECB.

Event Control Block

The event control block (ECB) is used by the control program to test for completion of the read or write operation. The ECB is located in the first word of the DECB.

The type, length, data control block address, area address, key address, block address, and next address information is taken from the operands of the macro instruction and placed in the DECB for use by the control program. For BISAM, exception codes are returned by the control program after the corresponding WAIT or CHECK macro instruction is issued, as indicated in Figure 15. For BDAM, BSAM, BPAM, and QSAM, the control program provides a pointer to the IOB containing the status indicators shown in Figure 21 on page 52.

Figure 15. Exception Code Bits—BISAM

Exception Code Bit in DECB	READ	WRITE	Condition if On
0	X	Type K	Record not found
1	X	X	Record length check
2		Type KN	Space not found
3	X	Type K	Invalid request
4	X	X	Uncorrectable I/O error
5	X	X	Unreachable block
6	X		Overflow record[1]
7		Type KN	Duplicate record
8-15			Reserved for control program use

[1] The SYNAD exit routine is entered only if the CHECK macro is issued after the READ macro, and bit 0, 4, 5, or 7 is also on.

Notes to Figure 15:

Record Not Found: This condition is reported if the logical record with the specified key is not found in the data set, if the specified key is higher than the highest key in the highest level index, or if the record is not in either the prime area or the overflow area of the data set.

Record Length Check: This condition is reported, for READ and update WRITE macro instructions, if an overriding length is specified and (1) the record format is blocked, (2) the record format is unblocked but the overriding length is greater than the length known to the control program, or (3) the record is fixed length and the overriding length does not agree with the length known to the control program. This condition is reported for the add WRITE macro instruction if an overriding length is specified.

When blocked records are being updated, the control program must find the high key in the block in order to write the block. (The high key is not necessarily the same as the key supplied by the problem program.) The high key is needed for writing because the control unit for direct access devices permits writing only if a search on equal is satisfied; this search can be satisfied only with the high key in the block. If the user were permitted to specify an overriding length shorter than the block length, the high key might not be read; then, a subsequent write request could not be satisfied. In addition, failure to write a high key during update would make a subsequent update impossible.

Space Not Found for Adding a Record: This condition is reported if no room exists in either the appropriate cylinder overflow area or the independent overflow area when a new record is to be added to the data set. The data set is not changed in any way in this situation.

Invalid Request: This condition is reported for either of two reasons. First, if byte 25 of the data event control block indicates that this request is an update WRITE macro instruction corresponding to a READ (for update) macro instruction, but the input/output block (IOB) for the READ is not found in the update queue. This condition could be caused by the problem program altering the contents of byte 25 of the data event control block. Second, if a READ or WRITE macro instruction specifies dynamic buffering (that is, 'S' in the area address operand) but the DCBMACRF field of the data control block does not specify dynamic buffering.

Uncorrectable Input/Output Error: This condition is reported if the control program's error recovery procedures encounter an uncorrectable error in transferring data.

Unreachable Block: This condition is reported if an uncorrectable input/output error occurs while searching the indexes or following an overflow chain. It is also posted if the data field of an index record contains an improper address (that is, points to the wrong cylinder or track or is an invalid address).

Overflow Record: This condition is reported if the record just read is an overflow record. (See the section on direct retrieval and update of an indexed sequential data set in *Data Administration Guide* for considerations during BISAM updating.)

Duplicate Record Presented for Inclusion in the Data Set: This condition is reported if the new record to be added has the same key as a record in the data set. However, if the delete option was specified and the record in the data set is marked for deletion, this condition is not reported. Instead, the new record replaces the existing record.

If the record format is blocked and the relative key position is zero, the new record cannot replace an existing record that is of equal key and is marked for deletion.

Figure 16. Exception Code Bits—QISAM

Exception Field	Code Bit	CLOSE	Code GET	Set PUT	by PUTX	SETL	Condition if On
DCBEXCD1	0					Type K	Record Not Found
	1					Type I	Invalid actual address for lower limit
	2	X					Space not found for adding a record
	3					X	Invalid request
	4		X				Uncorrectable input error
	5	X		X	X		Uncorrectable output error
	6		X			X	Block could not be reached (input)
	7	X	X				Block could not be reached (update)
DCBEXCD2	0			X			Sequence check
	1			X			Duplicate record
	2	X					Data control block closed when error routine entered
	3		X				Overflow record[1]
	4			X			Incorrect record length
	5-7						Reserved for future use

[1] The SYNAD exit routine is entered only if bit 4, 5, 6, or 7 of DCBEXCD1 is also on.

Notes to Figure 16:

Record Not Found: This condition is reported if the logical record with the specified key is not found in the data set, if the specified key is higher than the highest key in the highest level index, or if the record is not in either the prime area or the overflow area of the data set.

Invalid Actual Address for Lower Limit: This condition is reported if the specified lower limit address is outside the space allocated to the data set.

Space Not Found for Adding a Record: This condition is reported if the space allocated to the data set is already filled. In locate mode, a buffer segment address is not provided. In move mode, data is not moved.

Invalid Request: This condition is reported if (1) the data set is already being referred to sequentially by the problem program, (2) the buffer cannot contain the key and the data, or (3) the specified type is not also specified in the DCBMACRF field of the data control block.

Uncorrectable Input Error: This condition is reported if the control program's error recovery procedures encounter an uncorrectable error when transferring a block from secondary storage to an input buffer. The buffer address is placed in register 1, and the SYNAD exit routine is given control when a GET macro instruction is issued for the first logical record.

Uncorrectable Output Error: This condition is reported if the control program's error recovery procedures encounter an uncorrectable error when transferring a block from an output buffer to secondary storage. If the error is encountered during closing of the data control block, bit 2 of DCBEXCD2 is set to *1* and the SYNAD exit routine is given control immediately. Otherwise, control program action depends on whether load mode or scan mode is being used.

If a data set is being created (load mode), the SYNAD exit routine is given control when the next PUT or CLOSE macro instruction is issued. In the case of a failure to write a data block, register 1 contains the address of the output

buffer, and register 0 contains the address of a work area containing the first 16 bytes of the IOB; for other errors, the contents of register 1 are meaningless. After appropriate analysis, the SYNAD exit routine should close the data set or end the job step. If records are to be subsequently added to the data set using the queued indexed sequential access method (QISAM), the job step should be terminated by issuing an abend macro instruction. (Abend closes all open data sets. However, an ISAM data set is only partially closed, and it can be reopened in a later job to add additional records by using QISAM.) Subsequent execution of a PUT macro instruction would cause reentry to the SYNAD exit routine, because an attempt to continue loading the data set would produce unpredictable results.

If a data set is being processed (scan mode), the address of the output buffer in error is placed in register 1, the address of a work area containing the first 16 bytes of the IOB is placed in register 0, and the SYNAD exit routine is given control when the next GET macro instruction is issued. Buffer scheduling is suspended until the next GET macro instruction is reissued.

Block Could Not Be Reached (Input): This condition is reported if the control program's error recovery procedures encounter an uncorrectable error in searching an index or overflow chain. The SYNAD exit routine is given control when a GET macro instruction is issued for the first logical record of the unreachable block.

Block Could Not Be Reached (Update): This condition is reported if the control program's error recovery procedures encounter an uncorrectable error in searching an index or overflow chain.

If the error is encountered during closing of the data control block, bit 2 of DCBEXCD2 is set to 1 and the SYNAD exit routine is given control immediately. Otherwise, the SYNAD exit routine is given control when the next GET macro instruction is issued.

Sequence Check: This condition is reported if a PUT macro instruction refers to a record whose key has a smaller numeric value than the key of the record previously referred to by a PUT macro instruction. The SYNAD exit routine is given control immediately; the record is not transferred to secondary storage.

Duplicate Record: This condition is reported if a PUT macro instruction refers to a record whose key duplicates that of the record previously referred to by a PUT macro instruction. The SYNAD exit routine is given control immediately; the record is not transferred to secondary storage.

Data Control Block Closed When Error Routine Entered: This condition is reported if the control program's error recovery procedures encounter an uncorrectable output error during closing of the data control block. Bit 5 or 7 of DCBEXCD1 is set to 1, and the SYNAD exit routine is immediately given control. After appropriate analysis, the SYNAD routine must branch to the address in return register 14 so that the control program can finish closing the data control block.

Overflow Record: This condition is reported if the input record is an overflow record.

Incorrect Record Length: This condition is reported if the length of the record as specified in the record-descriptor word (RDW) is larger than the value in the DCBLRECL field of the data control block.

Figure 17. Exception Code Bits—BDAM

Exception Code Bit	READ	WRITE	Condition if On
0	X	X	Record not found
1	X	X	Record length check
2		X	Space not found
3	X	X	Invalid request—see bits 9-15
4	X	X	Uncorrectable I/O error
5	X	X	End of data
6	X	X	Uncorrectable error
7		X	Not read with exclusive control
8			Not used
9		X	WRITE to input data set
10	X	X	Extended search with DCBLIMCT=0
11	X	X	Block or track requested was outside data set
12		X	Tried to write capacity record
13	X	X	Specified key as search argument when KEYLEN=0 or no key address supplied
14	X	X	Request for options not in data control block
15		X	Attempt to add fixed-length record with key beginning with hexadecimal FF

Notes to Figure 17:

Record Not Found: This condition is reported if the search argument is not found in the data set.

Record Length Check: This condition occurs for READ and WRITE (update) and WRITE (add). For WRITE (update) variable-length records only, the length in the BDW does not match the length of the record to be updated. For all remaining READ and WRITE (update) conditions, the BLKSIZE, when **S** is specified in the READ or WRITE macro, or the length given with these macros does not agree with the actual length of the record. For WRITE (add), fixed-length records, the BLKSIZE, when **S** is specified in the WRITE macro, or the length give with this macro does not agree with the actual length of the record. For WRITE (add), all other conditions, no error can occur.

Space Not Found for Adding a Record: This condition occurs if either there is no dummy record when adding an F-format record, or there is no space available when adding a V- or U-format record.

Invalid Request: Occurs whenever one of the following bits is set to one:

Bit	Meaning
9	A WRITE was attempted for an input data set.
10	An extended search was requested, but LIMCT was zero.
11	The relative block or relative track requested was not in the data set.
12	Writing a capacity record (R0) was attempted.

13 A READ or WRITE with key was attempted, but either KEYLEN equaled zero or the key address was not supplied.

14 The READ or WRITE macro request options conflict with the OPTCD or MACRF parameters.

15 A WRITE (add) with fixed length was attempted with the key beginning with X'FF'.

Uncorrectable Input/Output Error: This condition is reported if the control program's error recovery procedures encounter an uncorrectable error in transferring data between real and secondary storage.

End of Data: This only occurs as a result of a READ (type DI, DIF, or DIX) when the record requested is an end-of-data record.

Uncorrectable error: Same conditions as for bit 4.

Not Read with Exclusive Control: A WRITE, type DIX or DKX, has occurred for which there is no previous corresponding READ with exclusive control.

Figure 18. Register Contents on Entry to SYNAD Routine—QISAM

Register	Bits	Meaning
0	0	Bit 0 = 1 indicates that bits 8-31 hold the address of the key in error (only set for a sequence error). If bit 0 = 1—address of key that is out of sequence. If bit 0 = 0—address of a work area.
	1-7	Not used.
	8-31	Address of a work area containing the first 16 bytes of the IOB (after an uncorrectable input/output error caused by a GET, PUT, or PUTX macro instruction; original contents destroyed in other cases). If the error condition was detected before I/O was started, register 0 contains all zeros.
1	0-7	Not used.
	8-31	Address of the buffer containing the error record (after an uncorrectable input/output error caused by a GET, PUT, or PUTX macro instruction while attempting to read or write a data record; in other cases, this register contains 0).
2-13	0-31	Contents that existed before the macro instruction was issued.
14	0-7	Not used.
	8-31	Return address. This address is either an address in the control program's close routine (bit 2 of DCBEXCD2 is on), or the address of the instruction following the expansion of the macro instruction that caused the SYNAD exit routine to be given control (bit 2 of DCBEXCD2 is off).
15	0-7	Not used.
	8-31	Address of the SYNAD exit routine.

Figure 19. Register Contents on Entry to SYNAD Routine—BISAM

Register	Bits	Meaning
0	0-7	Not used.
	8-31	Address of the first IOB sense byte. (Sense information is valid only when associated with a unit check condition.)
1	0-7	Not used.
	8-31	Address of the DECB.
2-13	0-31	Contents that existed before the macro instruction was issued.
14	0-7	Not used.
	8-31	Return address.
15	0-7	Not used.
	8-31	Address of the SYNAD exit routine.

Figure 20. Register Contents on Entry to SYNAD Routine—BDAM, BPAM, BSAM, and QSAM

Register	Bits	Meaning
0	0-7	Value to be added to the status indicator's address to provide the address of the first CCW (QSAM only).
	8-31	Address of the associated data event control block for BDAM, BPAM, and BSAM; address of the status indicators shown in Figure 21 on page 52 for QSAM.
1	0	Bit is on for error caused by input operation.
	1	Bit is on for error caused by output operation.
	2	Bit is on for error caused by BSP, CNTRL, or POINT macro instruction (BPAM AND BSAM only).
	3	Bit is on if error occurred during update of existing record or if error did not prevent reading of the record. Bit is off if error occurred during creation of a new record or if error prevented reading of the record.
	4	Bit is on if the request was invalid. The status indicators pointed to in the data event control block are not present (BDAM, BPAM, and BSAM only).
	5	Bit is on if an invalid character was found in paper tape conversion (BSAM and QSAM only).
	6	Bit is on for a hardware error (BDAM only).
	7	Bit is on if no space was found for the record (BDAM only).
	8-31	Address of the associated data control block.
2-13	0-31	Contents that existed before the macro instruction was issued.
14	0-7	Not used.
	8-31	Return address.
15	0-7	Not used.
	8-31	Address of the error analysis routine.

Offset From IOB Address			
Byte	**Bit**	**Meaning**	**Name**
+2	0	Command reject	Sense byte 1
	1	Intervention required	
	2	Bus-out check	
	3	Equipment check	
	4	Data check	
	5	Overrun	
	6,7	Device-dependent information; see the appropriate device manual	
+3	0-7	Device-dependent information; see the appropriate device manual	Sense byte 2
The following bytes make up the low-order seven bytes of the channel status word:			
+9	–	Command address	
+12	0	Attention	Status byte 1
	1	Status modifier	(Unit)
	2	Control unit end	
	3	Busy	
	4	Channel end	
	5	Device end	
	6	Unit check–must be on for sense bytes to be meaningful	
	7	Unit exception	
+13	0	Program-controlled interrupt	Status byte 2
	1	Incorrect length	(Channel)
	2	Program check	
	3	Protection check	
	4	Channel data check	
	5	Channel control check	
	6	Interface control check	
	7	Chaining check	
+14	–	Count field (2 bytes)	

Figure 21. Status Indicators for the SYNAD Routine—BDAM, BPAM, BSAM, and QSAM

Note: If the sense bytes are X'10FE', the control program has set them to this invalid combination because sense bytes could not be obtained from the device because of recurrence of unit checks.

The event control block is used for communication between the various components of the system and between problem programs and the system. An event control block is the subject of WAIT and POST macro instructions (see Figure 22).

Offset	Bytes and Alignment	Code	Bit	Hex. Dig.	Description
00	1	10xx	xxxx		W—Waiting for completion of an event.
1	3				Contains the address of the RB issuing the WAIT macro if the ECB has the WAIT bit on. Once the event has completed and the ECB is posted, the C bit is set with other bits in byte 0 and these 3 bytes (1-3) are zero, for all access methods except BDAM. Exception codes are returned in bytes 1 and 2 of the ECB for BDAM.
00		01xx	xxxx		C—The event has completed.
					One of the following completion codes will appear at the completion of a channel program:
					Access Methods other than BTAM
		0111	1111	7F	Channel program has terminated without error. (CSW contents useful.)
		0100	0001	41	Channel program has terminated with permanent error. (CSW contents useful.)
		0100	0010	42	Channel program has terminated because a direct access extent address has been violated. (CSW contents do not apply.)
		0100	0011	43	I/O abend condition occurred while loading the error recovery routine. (CSW contents do not apply.)
		0100	0100	44	Channel program has been intercepted because of permanent error associated with device end for previous request. You may reissue the intercepted request. (CSW contents do not apply.)
		0100	1000	48	Request element for channel program has been made available after it has been purged. (CSW contents do not apply.)

Figure 22 (Part 1 of 2). Status Indicators in the ECB

Offset	Bytes and Alignment	Code	Bit	Hex. Dig.	Description
		0100	1011	4B	One of the following errors occurred during tape error recovery processing: • The CSW command address in the IOB was zeros. • An unexpected load point was encountered. (CSW contents do not apply in either case.)
		0100	1111	4F	Error recovery routines have been entered because of direct access error but are unable to read home addresses or record 0. (CSW contents do not apply.)
		0101	0000	50	Channel program terminated with error. Input block was a DOS-embedded checkpoint record. (CSW contents do not apply.)

Figure 22 (Part 2 of 2). Status Indicators in the ECB

EODAD End-of-Data-Set Exit Routine

Described

The EODAD parameter of the DCB macro specifies the address of your end-of-data-set routine, which may perform any final processing on an input data set. This routine is entered when an FEOV macro is issued or when a CHECK or GET macro is issued and there are no more records or blocks to be retrieved. (This allows you to issue WRITE macros before an FEOV macro is issued.) On a READ request, this routine is entered when you issue a CHECK macro to check for completion of the read operation. For a BSAM data set that is opened for UPDAT, this routine is entered at the end of each volume.

Register Contents

When control is passed to the EODAD routine, the registers contain the following information:

Register	Contents
0-1	Reserved
2-13	Contents before execution of CHECK, GET, or FEOV macro instruction
14	Address of the instruction after the last issued GET, CHECK, or FEOV macro
15	Reserved

Programming Considerations

The EODAD routine is not a subroutine, but rather a continuation of the routine that issued the CHECK, GET, or FEOV macro. After it is in your EODAD routine, you can continue normal processing, such as repositioning and resuming processing of the data set, closing the data set, or processing another data set.

For BSAM, you must first reposition the data set that reached end-of-data if you want to issue a BSP, READ, or WRITE macro. You can reposition your data set by issuing a CLOSE TYPE=T macro instruction. If a READ macro is issued before the data set is repositioned, unpredictable results will occur.

For BPAM, you may reposition the data set by issuing a FIND or POINT macro. (CLOSE TYPE=T with BPAM results in no operation performed.)

For QISAM, you can continue processing the input data set that reached end-of-data by first issuing an ESETL macro to end the sequential retrieval, then issuing a SETL macro to set the lower limit of sequential retrieval. You can then issue GET macros to the data set.

Your task will be abnormally ended under either of the following conditions:

- No exit routine is provided.
- A GET macro is issued in the EODAD routine to the DCB that caused this routine to be entered (unless the access method is QISAM).

SYNAD Synchronous Error Routine Exit

Described

The SYNAD parameter of the DCB macro specifies the address of an error routine that is to be given control when an input/output error occurs. This routine can be used to analyze exceptional conditions or uncorrectable errors. The block being read or written can be accepted or skipped, or processing can be terminated.

If an input/output error occurs during data transmission, standard error recovery procedures that are provided by the operating system try to correct the error before returning control to your program. An uncorrectable error usually causes an abnormal termination of the task. However, if you specify in the DCB macro the address of an error analysis routine (called a SYNAD routine), that routine can try to correct the error and prevent an abnormal termination. The routine is given control when the application program issues the next access method macro after the system has detected an uncorrectable error.

Register Contents

For a description of the register contents on entry to your SYNAD routine, see "Status Information Following an Input/Output Operation" on page 42.

Programming Considerations

You can write a SYNAD routine to determine the cause and type of error that occurred by examining:

- The contents of the general registers
- The data event control block (see "Status Information Following an Input/Output Operation" on page 42)
- The exceptional condition code
- The standard status and sense indicators

You can use the SYNADAF macro to perform this analysis automatically. This macro produces an error message that can be printed by a later PUT or WRITE macro.

After completing the analysis, you can return control to the operating system or close the data set. If you close the data set, note that you may not use the temporary close (CLOSE TYPE=T) option in the SYNAD routine. To continue processing the same data set, you must first return control to the control program by a RETURN macro. The control program then transfers control to your processing program, subject to the conditions described below. Never attempt to reread or rewrite the record, because the system has already attempted to recover from the error.

When you are using GET and PUT to process a sequential data set, the operating system provides three automatic error options (EROPT) to be used if there is no SYNAD routine or if you want to return control to your program from the SYNAD routine:

- ACC—accept the erroneous block
- SKP—skip the erroneous block
- ABE—abnormally terminate the task

These options are applicable only to data errors, because control errors result in abnormal termination of the task. Data errors affect only the validity of a block of data. Control errors affect information or operations necessary for continued processing of the data set. These options are not applicable to output errors, except output errors on the printer. If the EROPT and SYNAD fields are not completed, ABE is assumed.

If a control error or a physical I/O error is encountered for a SYSIN or SYSOUT dataset, the EROPT options will be ignored and the task will be abnormally terminated.

You should not use the FEOV macro against the data set for which the SYNAD routine was entered, within the SYNAD routine.

Because EROPT applies to a physical block of data, and not to a logical record, use of SKP or ACC may result in incorrect assembly of spanned records.

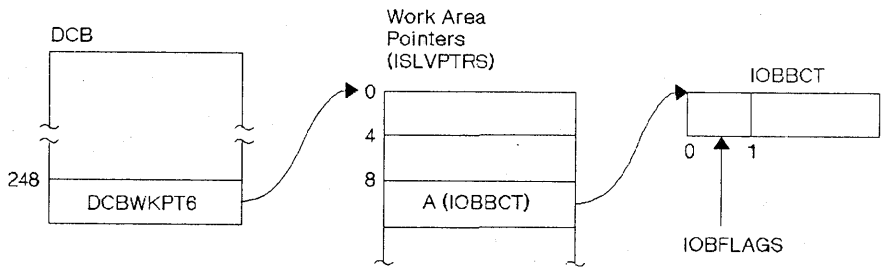
When you use READ and WRITE macros, errors are detected when you issue a CHECK macro. If you are processing a direct or sequential data set and you return to the control program from your SYNAD routine, the operating system assumes that you have accepted the bad record. If you are creating a direct data set and you return to the control program from your SYNAD routine, your task is abnormally terminated. In the case of processing a direct data set, the return should be made to the control program via register 14 to make a control block (the IOB) available for reuse in a later READ or WRITE macro.

Your SYNAD routine can end by branching to another routine in your program, such as a routine that closes the data set. It can also end by returning control to the control program, which then returns control to the next sequential instruction (after the macro) in your program. If your routine returns control, the conventions for saving and restoring register contents are as follows:

- The SYNAD routine must preserve the contents of registers 13 and 14. If required by the logic of your program, the routine must also preserve the contents of registers 2 through 12. On return to your program, the contents of registers 2 through 12 will be the same as on return to the control program from the SYNAD routine.
- The SYNAD routine must not use the save area whose address is in register 13, because this area is used by the control program. If the routine saves and restores register contents, it must provide its own save area.

- If the SYNAD routine calls another routine or issues supervisor or data management macros, it must provide its own save area or issue a SYNADAF macro. The SYNADAF macro provides a save area for its own use, and then makes this area available to the SYNAD routine. Such a save area must be removed from the save area chain by a SYNADRLS macro before control is returned to the control program.

If the error analysis routine receives control from the close routine when indexed sequential data sets are being created (the DCB is opened for QISAM load mode), bit 3 of the IOBFLAGS field in the load mode buffer control table (IOBBCT) is set to 1. The DCBWKPT6 field in the DCB contains an address of a list of work area pointers (ISLVPTRS). The pointer to the IOBBCT is at offset 8 in this list as shown in the following diagram:



If the error analysis routine receives control from the CLOSE routine when indexed sequential data sets are being processed using QISAM scan mode, bit 2 of the DCB field DCBEXCD2 is set to 1.

Figure 23 gives the contents of registers 0 and 1 when a SYNAD routine specified in a DCB gets control while indexed sequential data sets are being processed.

Figure 23. Register Contents for DCB-Specified ISAM SYNAD Routine

Register	BISAM	QISAM .
0	Address of the DECB	0, or, for a sequence check, the address of a field containing the higher key involved in the check
1	Address of the DECB	0

For information on QISAM error conditions and the meaning they have when the ISAM interface to VSAM is being used, see *VSAM Administration Guide.*

EXLST Exit List

The EXLST parameter of the DCB macro specifies the address of a list that may contain the addresses of special processing routines, a forms control buffer (FCB) image, a user totaling area, an area for a copy of the JFCB, and an allocation retrieval list. An exit list must be created if user label, data control block, end-of-volume, block count, JFCBE, or DCB abend exits are used, or if a PDAB macro or FCB image is defined in the processing program.

The exit list is built of 4-byte entries that must be aligned on fullword boundaries. Each exit list entry is identified by a code in the high-order byte, and the address of the routine, image, or area is specified in the 3 low-order bytes. Codes and addresses for the exit list entries are shown in Figure 24.

Figure 24 (Page 1 of 2). DCB Exit List Format and Contents

Entry Type	Hex Code	3-Byte Address—Purpose
Inactive entry	00	Ignore the entry; it is not active.
Input header label exit	01	Process a user input header label.
Output header label exit	02	Create a user output header label.
Input trailer label exit	03	Process a user input trailer label.
Output trailer label exit	04	Create a user output trailer label.
Data control block exit	05	Take a data control block exit.
End-of-volume exit	06	Take an end-of-volume exit.
JFCB exit	07	JFCB address for RDJFCB and OPEN TYPE=J SVCs.
	08	Reserved.
	09	Reserved.
User totaling area	0A	Address of beginning of user's totaling area.
Block count exit	0B	Take a block-count-unequal exit.
Defer input trailer label	0C	Defer processing of a user input trailer label from end-of-data until closing.
Defer nonstandard input trailer label	0D	Defer processing a nonstandard input trailer label on magnetic tape unit from end-of-data until closing (no exit routine address).
	0E-0F	Reserved.
FCB image	10	Define an FCB image.
DCB abend exit	11	Examine the abend condition and select one of several options.
QSAM parallel input	12	Address of the PDAB for which this DCB is a member.
Allocation retrieval list	13	Retrieve allocation information for one or more data sets with the RDJFCB macro.
	14	Reserved.
JFCBE exit	15	Take an exit during OPEN to allow user to examine JCL=specified setup requirements for a 3800 printer.

Figure 24 (Page 2 of 2). DCB Exit List Format and Contents		
Entry Type	**Hex Code**	**3-Byte Address—Purpose**
	16	Reserved.
OPEN/EOV nonspecific tape volume mount	17	Option to specify a tape volume serial number.
OPEN/EOV volume security/verification	18	Verify a tape volume and some security checks.
	19-7F	Reserved.
Last entry	80	Treat this entry as the last entry in the list. This code can be specified with any of the above but must always be specified with the last entry.

You can activate or deactivate any entry in the list by placing the required code in the high-order byte. Care must be taken, however, not to destroy the last entry indication. The operating system routines scan the list from top to bottom, and the first active entry found with the proper code is selected.

You can shorten the list during execution by setting the high-order bit to 1, and extend it by setting the high-order bit to 0.

Register Contents for Exits from EXLST

When control is passed to an exit routine, the registers contain the following information:

Register	Contents
0	Variable; see exit routine description.
1	The 3 low-order bytes contain the address of the DCB currently being processed, except when the user-label exits (X'01'-X'04'), user totaling exit (X'0A'), DCB abend exit (X'11'), nonspecific tape volume mount exit (X'17'), or the tape volume security/verification exit (X'18') is taken, when register 1 contains the address of a parameter list. The contents of the parameter list are described in the explanation of each exit routine.
2-13	Contents before execution of the macro.
14	Return address (must not be altered by the exit routine).
15	Address of exit routine entry point.

The conventions for saving and restoring register contents are as follows:

- The exit routine must preserve the contents of register 14. It need not preserve the contents of other registers. The control program restores the contents of registers 2 to 13 before returning control to your program.
- The exit routine must not use the save area whose address is in register 13, because this area is used by the control program. If the exit routine calls another routine or issues supervisor or data management macros, it must provide the address of a new save area in register 13.
- The exit routine must not issue an access method macro that refers to the DCB for which the exit routine was called, unless otherwise specified in the individual exit routine descriptions that follow.

Allocation Retrieval List

The RDJFCB macro uses the DCB exit list entry with code X'13' to retrieve allocation information (JFCBs and volume serial numbers). (**Note:** Although use of the RDJFCB macro is still supported, its use is not recommended.) When you issue RDJFCB, the JFCBs for the specified data sets, including concatenated data sets, and their volume serial numbers are placed in the area located at the address specified in the allocation retrieval list. The DCB exit list entry contains the address of the allocation retrieval list. The RDJFCB macro passes the following return codes in register 15:

Return Code	Meaning
0 (X'00')	RDJFCB has completed the allocation retrieval list successfully.
4 (X'04')	One or more DCBs had one of the following conditions and were skipped: • DCB currently being processed by Open/Close/End-Of-Volume or similar function. • No data set is allocated with the ddname that is in the DCB. • The DCB is not open and its ddname is blank. DCBs that were not skipped were handled successfully.
8 (X'08')	One or more DCBs had an allocation retrieval list which could not be handled. Each allocation retrieval list contains a reason code to describe its status. One or more DCBs may have an error described by return code 4, in which case their allocation retrieval lists will not have a reason code.

For more information on RDJFCB see *System—Data Administration*.

Programming conventions

The allocation retrieval list must be below the 16M line, but the allocation return area can be above the 16M line.

When you are finished obtaining information from the retrieval areas, free the storage with a FREEMAIN macro.

You can use the IHAARL macro to generate and map the allocation retrieval list. For more information on the IHAARL macro see *System—Data Administration*.

Restrictions

When OPEN TYPE=J is issued, the X'13' exit cannot be used. The JFCB exit at X'07' can be used instead (see "JFCB Exit" on page 71).

DCB Abend Exit

The DCB abend exit is provided to give you some options regarding the action you want the system to take when a condition arises that may result in abnormal termination of your task. This exit can be taken any time an abend condition arises during the process of opening, closing, or handling an end-of-volume condition for a DCB associated with your task.

When an abend condition arises, a write-to-programmer message about the abend is issued and your DCB abend exit is given control, provided there is an active DCB abend exit routine address in the DCB being processed. If STOW called the end-of-volume routines to get secondary space to write an end-of-file mark for a partitioned data set, or if the DCB being processed is for an indexed sequential data set, the DCB abend exit routine will not be given control if an abend condition occurs. The contents of the registers when your exit routine is entered are the same as for other DCB exit list routines, except that the 3 low-order bytes of register 1 contain the address of the parameter list described in Figure 25 on page 63. Your abend exit routine can choose one of four options:

- To immediately terminate your task
- To delay the abend until all the DCBs in the same OPEN or CLOSE macro are opened or closed
- To ignore the abend condition and continue processing without making reference to the DCB on which the abend condition was encountered, or
- To try to recover from the error.

Not all of these options are available for each abend condition. Your DCB abend exit routine must determine which option is available by examining the contents of the option mask byte (byte 3) of the parameter list. The address of the parameter list is passed in register 1. Figure 25 shows the contents of the parameter list and the possible settings of the option mask when your routine receives control. (All information in the parameter list is in binary.)

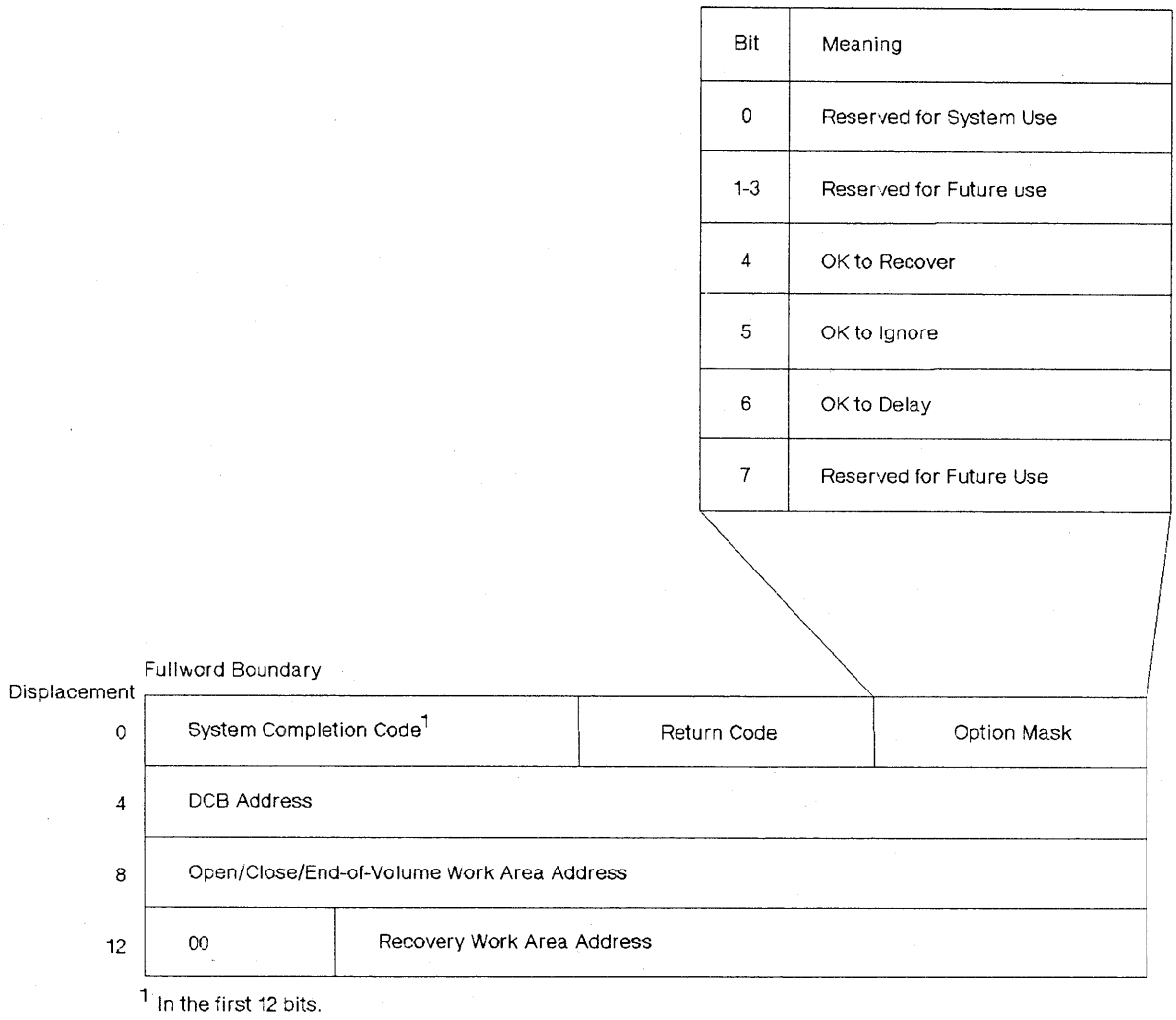


Figure 25. Parameter List Passed to DCB Abend Exit Routine

When your DCB abend exit routine returns control to the system control program (this can be done using the RETURN macro), the option mask byte must contain the setting that specifies the action you want to take. These actions and the corresponding settings of the option mask byte are:

Decimal Value	Action
0	Abnormally terminate the task immediately.
4	Ignore the abend condition.
8	Delay the abend until the other DCBs being processed concurrently are opened or closed.
12	Make an attempt to recover.

You must inspect bits 4, 5, and 6 of the option mask byte (byte 3 of the parameter list) to determine which options are available. If a bit is set to 1, the corresponding option is available. Indicate your choice by inserting the appropriate value in byte 3 of the parameter list, overlaying the bits you inspected. If you use a value that specifies an option that is not available, the abend is issued immediately.

If the contents of bits 4, 5, and 6 of the option mask are 0, you must not change the option mask. This unchanged option mask will result in a request for an immediate abend.

If bit 5 of the option mask is set to 1, you can ignore the abend by placing a value of 4 in byte 3 of the parameter list. Processing on the current DCB stops. If you subsequently attempt to use this DCB, the results are unpredictable. If you ignore an error in end-of-volume, control is returned to your program at the point that caused the end-of-volume condition (unless the end-of-volume routines were called by the close routines). If the end-of-volume routines were called by the close routines, an ABEND macro will be issued even though the ignore option was selected.

If bit 6 of the option mask is set to 1, you can delay the abend by placing a value of 8 in byte 3 of the parameter list. All other DCBs waiting for OPEN or CLOSE processing will be processed before the abend is issued. For end-of-volume, however, you can't delay the abend because the end-of-volume routine never has more than one DCB to process.

If bit 4 of the option mask is set to 1, you can attempt to recover. Place a value of 12 in byte 3 of the parameter list and provide information for the recovery attempt. Figure 26 lists the abend conditions for which recovery can be attempted. For abend conditions that can be ignored or delayed, see *System Messages Volume 1* and *System Messages Volume 2*.

Figure 26 (Page 1 of 2). Conditions for Which Recovery Can Be Attempted

System Completion Code	Return Code	Description of Error
117	X'38'	An I/O error occurred during execution of a read block ID command issued to establish tape position.
	X'3C'	DCB block count did not agree with the calculated block count.
137	X'24'	A specific volume serial number was specified for the second or subsequent volume of an output data set on magnetic tape. During EOV processing, it was discovered that the expiration date (from the HDR1 label of the first data set currently on the specified volume) had not passed. When requested to specify whether the volume could be used in spite of the expiration date, the operator did not reply **U**.
214	X'0C'	An I/O error occurred during execution of a read block ID command issued to establish tape position.

Figure 26 (Page 2 of 2). Conditions for Which Recovery Can Be Attempted

System Completion Code	Return Code	Description of Error
237	X'04'	Block count in DCB does not agree with block count in trailer label.
	X'0C'	DCB block count did not agree with the calculated block count.
413	X'18'	Data set was opened for input and no volume serial number was specified.
	X'24'	**LABEL=(n)** was specified, where **n** was greater than 1 and **vol=ser** was not specified for a tape data set.
613	X'08'	I/O error occurred during reading of tape label.
	X'0C'	Invalid tape label was read.
	X'10'	I/O error occurred during writing of tape label.
	X'14'	I/O error occurred during writing of tape label.
713	X'04'	A data set on magnetic tape was opened for INOUT, but the volume contained a data set whose expiration date had not been reached and the operator denied permission.
717	X'10'	I/O error occurred during reading of trailer label 1 to update block count in DCB.
737	X'28'	The EOV DA module was passed an error return code in register 15 after issuing the IEFSSREQ macro instruction. This indicates the subsystem (JES3) discovered a functional or logical error that it could not process.
813	X'04'	Data set name on header label does not match data set name on DD statement.

Recovery Requirements

For most types of recoverable errors, you should supply a recovery work area (see Figure 27 on page 66) with a new volume serial number for each volume associated with an error. If no new volumes are supplied for such errors, recovery will be attempted with the existing volumes, but the likelihood of successful recovery is greatly reduced.

If you request recovery for system completion code 117, return code 3C, or system completion code 214, return code 0C, or system completion code 237, return code 0C, you do not need to supply new volumes or a work area. The condition that caused the abend is disagreement between the DCB block count

and the calculated count from the hardware. To permit recovery, this disagreement is ignored and the value in the DCB will be used.

If you request recovery for system completion code 237, return code 04, you don't need to supply new volumes or a work area. The condition that caused the abend is the disagreement between the block count in the DCB and that in the trailer label. To permit recovery, this disagreement is ignored.

If you request recovery for system completion code 717, return code 10, you don't need to supply new volumes or a work area. The abend is caused by an I/O error during updating of the DCB block count. To permit recovery, the block count is not updated. Consequently, an abnormal termination with system completion code 237, return code 04, may result when you try to read from the tape after recovery. You may attempt recovery from the abend with system completion code 237, return code 04, as explained in the preceding paragraph.

System completion codes and their associated return codes are described in *System Codes*.

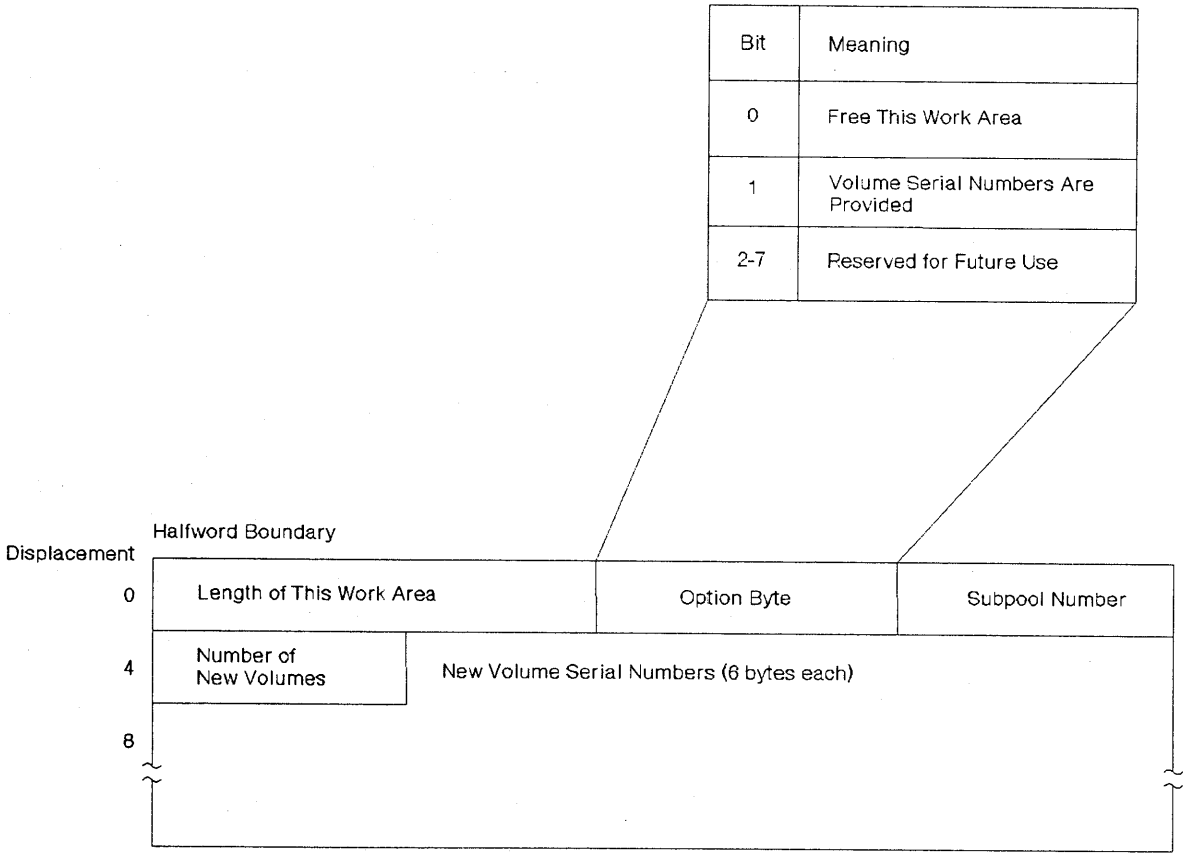


Figure 27. Recovery Work Area

The work area that you supply for the recovery attempt must begin on a halfword boundary and can contain the information described in Figure 27. Place a pointer to the work area in the last 3 bytes of the parameter list pointed to by register 1 and described in Figure 25 on page 63.

If you acquire the storage for the work area by using the GETMAIN macro, you can request that it be freed by a FREEMAIN macro after all information has been extracted from it. Set the high-order bit of the option byte in the work area to 1 and place the number of the subpool from which the work area was requested in byte 3 of the recovery work area.

Only one recovery attempt per data set is allowed during OPEN, CLOSE, or end-of-volume processing. If a recovery attempt is unsuccessful, you may not request another recovery. The second time through the exit routine you may request only one of the other options (if allowed): Issue the abend immediately, ignore the abend, or delay the abend. If at any time you select an option that is not allowed, the abend is issued immediately.

Note that, if recovery is successful, you still receive an abend message on your listing. This message refers to the abend that would have been issued if the recovery had not been successful.

Abend Installation Exit

The abend installation exit gives you an additional option for handling error situations that result in an abend. This exit is taken any time an abend condition occurs during the process of opening, closing, or handling an end-of-volume condition for a DCB. An IBM-supplied installation exit will give you the option to retry tape positioning when you receive a 613 system completion code, return code 08 or 0C. For additional information about the abend installation exit, see "Data Management Abend Installation Exit" on page 157.

DCB Open Exit

You can specify in an exit list the address of a routine that completes or modifies a DCB and does any additional processing required before the data set is completely open. The routine is entered during the opening process after the JFCB has been used to supply information for the DCB. The routine can determine data set characteristics by examining fields completed from the data set labels. When your DCB exit routine receives control, the 3 low-order bytes of register 1 will contain the address of the DCB currently being processed.

When opening a data set for output, you can force the system to determine the block size for a DASD data set by setting the block size in the DCB to zero before returning from this exit. If the zero value you supply is not changed by the DCB OPEN installation exit, OPEN will call DASD Calculation Services to obtain the system block size when OPEN takes control after return from the DCB OPEN installation exit.

As with label processing routines, the contents of register 14 must be preserved and restored if any macros are used in the routine. Control is returned to the operating system by a RETURN macro; no return code is required.

This exit is mutually exclusive with the JFCBE exit. If you need both the JFCBE and data control block OPEN exits, you must use the JFCBE exit to pass control to your routines.

The DCB OPEN exit is intended for modifying or updating the DCB. System functions should not be attempted in this exit prior to returning to OPEN processing; in particular, dynamic allocation, OPEN, CLOSE, EOV, and DADSM functions should not be invoked because of an existing OPEN enqueue on the SYSZTIOT resources.

EOV Defer Nonstandard Input Trailer Label Exit

In an exit list, you can specify a code that indicates that you want to defer nonstandard input trailer label processing from end-of-data until the data set is closed. The address portion of the entry is not used by the operating system.

An end-of-volume condition exists in several situations. Two examples are: (1) when the system reads a filemark or a tapemark at the end of a volume of a multivolume data set but that volume is not the last, and (2) when the system reads a filemark or a tapemark at the end of a data set. The first situation is referred to here as an end-of-volume condition, and the second as an end-of-data condition, although it, too, can occur at the end of a volume.

For an end-of-volume (EOV) condition, the EOV routine passes control to your nonstandard input trailer label routine, whether or not this exit code is specified. For an end-of-data condition when this exit code is specified, the EOV routine does not pass control to your nonstandard input trailer label routine. Instead, the close routine passes control to your end-of-data routine.

EOV Block Count Exit

You can specify in an exit list the address of a routine that will allow you to abnormally terminate the task or continue processing when the EOV routine finds an unequal block count condition. When you are using standard labeled input tapes, the block count in the trailer label is compared by the EOV routine with the block count in the DCB. The count in the trailer label reflects the number of blocks written when the data set was created. The number of blocks read when the tape is used as input is contained in the DCBBLKCT field of the DCB.

The routine is entered during EOV processing. The trailer label block count is passed in register 0. You may gain access to the count field in the DCB by using the address passed in register 1 plus the proper displacement, which is given in *Data Administration: Macro Instruction Reference*. If the block count in the DCB differs from that in the trailer label when no exit routine is provided, the task is abnormally terminated. The routine must terminate with a RETURN macro and a return code that indicates what action is to be taken by the operating system, as shown in Figure 28 on page 69. As with other exit routines, the contents of register 14 must be saved and restored if any macros are used.

Return Code	System Action
0 (X'00')	The task is to be abnormally terminated.
4 (X'04')	Normal processing is to be resumed.

Figure 28. System Response to Block Count Exit Return Code

EOV Exit for Physical Sequential Data Sets

You can specify in an exit list the address of a routine that is entered when EOV is reached in processing of a physical sequential data set.

When you concatenate data sets with unlike attributes, no EOV exits are taken.

When the EOV routine is entered, register 0 contains 0 unless user totaling was specified. If you specified user totaling in the DCB macro (by coding OPTCD=T) or in the DD statement for an output data set, register 0 contains the address of the user totaling image area. The routine is entered after a new volume has been mounted and all necessary label processing has been completed. If the volume is a reel of magnetic tape, the tape is positioned after the tapemark that precedes the beginning of the data.

You can use the EOV exit routine to take a checkpoint by issuing the CHKPT macro, which is discussed in *Checkpoint/Restart User's Guide*. If a checkpointed job step terminates abnormally, it can be restarted from the EOV checkpoint. When the job step is restarted, the volume is mounted and positioned as upon entry to the routine. Restart becomes impossible if changes are made to the link pack area (LPA) library between the time the checkpoint is taken and the time the job step is restarted. When the step is restarted, pointers to EOV modules must be the same as when the checkpoint was taken.

The EOV exit routine returns control in the same manner as the data control block exit routine. The contents of register 14 must be preserved and restored if any macros are used in the routine. Control is returned to the operating system by a RETURN macro; no return code is required.

FCB Image Exit

You can specify in an exit list the address of a forms control buffer (FCB) image. This FCB image can be loaded into the forms control buffer of the printer control unit. The FCB controls the movement of forms in printers that do not use a carriage control tape.

Multiple exit list entries in the exit list can define FCBs. The OPEN and SETPRT routines search the exit list for requested FCBs before searching SYS1.IMAGELIB.

The first 4 bytes of the FCB image contain the image identifier. To load the FCB, this image identifier is specified in the FCB parameter of the DD statement, by the SETPRT macro, or by the system operator in response to message IEC127D or IEC129D.

For an IBM 3203, 3211, 3262, 4245, or 4248 Printer, the image identifier is followed by the FCB image described in *System—Data Administration*. For a 3800 FCB image, see *IBM 3800 Printing Subsystem Programmer's Guide*. For a 3800 Model 3 FCB image, see *IBM 3800 Model 3 Printing Subsystem Programmer's Guide*.

You can use an exit list to define an FCB image only when writing to an online printer. Figure 29 illustrates one way the exit list can be used to define an FCB image.

```
                ...
                DCB     ..,EXLST=EXLIST
                ...
EXLIST          DS      0F
                DC      X'10'           Flag code for FCB image
                DC      AL3(FCBIMG)     Address of FCB image
                DC      X'80000000'     End of EXLST and a null entry
FCBIMG          DC      CL4'IMG1'       FCB identifier
                DC      X'00'           FCB is not a default
                DC      AL1(67)         Length of FCB
                DC      X'90'           Offset print line
* 16 line character positions to the right
                DC      X'00'           Spacing is 6 lines per inch
                DC      5X'00'          Lines 2-6, no channel codes
                DC      X'01'           Line 7, channel 1
                DC      6X'00'          Lines 8-13, no channel codes
                DC      X'02'           Line (or Lines) 14, channel 2
                DC      5X'00'          Line (or Lines) 15-19, no channel codes
                DC      X'03'           Line (or Lines) 20, channel 3
                DC      9X'00'          Line (or Lines) 21-29, no channel codes
                DC      X'04'           Line (or Lines) 30, channel 4
                DC      19X'00'         Line (or Lines) 31-49, no channel codes
                DC      X'05'           Line (or Lines) 50, channel 5
                DC      X'06'           Line (or Lines) 51, channel 6
                DC      X'07'           Line (or Lines) 52, channel 7
                DC      X'08'           Line (or Lines) 53, channel 8
                DC      X'09'           Line (or Lines) 54, channel 9
                DC      X'0A'           Line (or Lines) 55, channel 10
                DC      X'0B'           Line (or Lines) 56, channel 11
                DC      X'0C'           Line (or Lines) 57, channel 12
                DC      8X'00'          Line (or Lines) 58-65, no channel codes
                DC      X'10'           End of FCB image
                ...
                END
//ddname        DD      UNIT=3211,FCB=(IMG1,VERIFY)
/*
```

Figure 29. Defining an FCB Image for a 3211

JFCB Exit

The JFCB exit is used with the RDJFCB macro and OPEN TYPE=J. The RDJFCB macro uses the address specified in the DCB exit list entry at X'07' to place a copy of the JFCB for each DCB specified by the RDJFCB macro.

Note: Although use of the JFCB exit is still supported, its use is not recommended.

The area is 176 bytes and must begin on a fullword boundary. It must be located in the user's region. Users running in 31-bit addressing mode must ensure that this area is located below 16 megabytes virtual. The DCB may be either open or closed when the RDJFCB macro is executed.

If RDJFCB fails while processing a DCB associated with your RDJFCB request, your task is abnormally terminated. You cannot use the DCB abend exit to recover from a failure of the RDJFCB macro. For more information about the RDJFCB macro see *System—Data Administration*.

JFCBE Exit

JCL-specified setup requirements for the IBM 3800 Printing Subsystem cause a JFCB extension (JFCBE) to be created to reflect those specifications. A JFCBE exists if BURST, MODIFY, CHARS, FLASH, or any copy group is coded on the DD statement. The JFCBE exit can be used to examine or modify those specifications in the JFCBE. (**Note:** Although use of the JFCBE exit is still supported, its use is not recommended.) The address of the routine should be placed in an exit list. (The device allocated does not have to be a 3800.) This exit is taken during OPEN processing and is mutually exclusive with the data control block exit. If you need both the JFCBE and data control block exits, you must use the JFCBE exit to pass control to your routines.

With a 3800, when you issue the SETPRT macro to a SYSOUT data set, the JFCBE is further updated from the information in the SETPRT parameter list.

When control is passed to your exit routine, the contents of register 1 will be the address of the DCB being processed.

The area pointed to by register 0 will also contain the 4-byte FCB identification that is obtained from the JFCB. The FCB identification is placed in the 4 bytes following the 176-byte JFCBE. If the FCB operand was not coded on the DD statement, this FCB field will be binary zeros.

If your copy of the JFCBE is modified during an exit routine, you should indicate this fact by turning on bit JFCBEOPN (X'80' in JFCBFLAG) in the JFCBE copy. On return to OPEN, this bit indicates whether the system copy is to be updated. The 4-byte FCB identification in your area will be used to update the JFCB regardless of the bit setting. Checkpoint/restart also interrogates this bit to determine which version of the JFCBE will be used at restart time. If this bit is not on, the JFCBE generated by the restart JCL will be used.

Open/Close/EOV Standard User Label Exit

When you create a data set with physical sequential or direct organization, you can provide routines to create your own data set labels. You can also provide routines to verify these labels when you use the data set as input. Each label is 80 characters long, with the first 4 characters UHL1,UHL2,...,UHL8 for a header label or UTL1,UTL2,...,UTL8 for a trailer label. User labels are not allowed on indexed sequential data sets.

The physical location of the labels on the data set depends on the data set organization. For direct (BDAM) data sets, user labels are placed on a separate user label track in the first volume. User label exits are taken only during execution of the OPEN and CLOSE routines. Thus you may create or examine as many as eight user header labels only during execution of OPEN and as many as eight trailer labels only during execution of CLOSE. Because the trailer labels are on the same track as the header labels, the first volume of the data set must be mounted when the data set is closed.

For physical sequential (BSAM or QSAM) data sets, you may create or examine as many as eight header labels and eight trailer labels on each volume of the data set. For ASCII tape data sets, you may create an unlimited number of user header and trailer labels. The user label exits are taken during OPEN, close, and EOV processing.

To create or verify labels, you must specify the addresses of your label exit routines in an exit list as shown in Figure 24 on page 59. Thus you may have separate routines for creating or verifying header and trailer label groups. Care must be taken if a magnetic tape is read backward, because the trailer label group is processed as header labels and the header label group is processed as trailer labels.

When your routine receives control, the contents of register 0 are unpredictable. Register 1 contains the address of a parameter list. The contents of registers 2 to 13 are the same as when the macro instruction was issued. However, if your program does not issue the CLOSE macro, or abnormally ends before issuing CLOSE, the CLOSE macro will be issued by the control program, with control-program information in these registers.

The parameter list pointed to by register 1 is a 16-byte area aligned on a fullword boundary. Figure 30 shows the contents of the area.

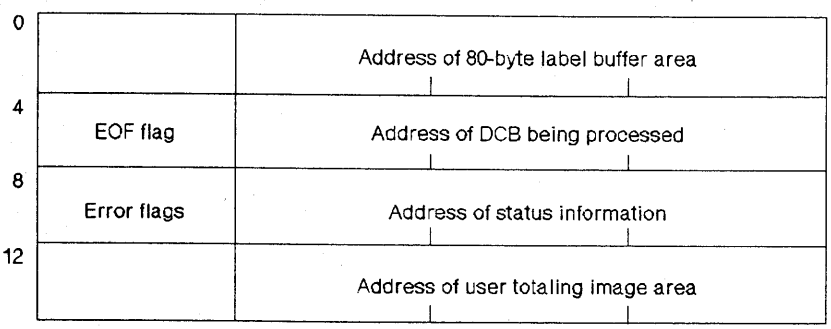


Figure 30. Parameter List Passed to User Label Exit Routine

The first address in the parameter list points to an 80-byte label buffer area. For input, the control program reads a user label into this area before passing control to the label routine. For output, the user label exit routine builds labels in this area and returns to the control program, which writes the label. When an input trailer label routine receives control, the EOF flag (high-order byte of the second entry in the parameter list) is set as follows:

```
Bit 0 = 0: Entered at EOV
Bit 0 = 1: Entered at end-of-file
Bits 1-7:  Reserved
```

When a user label exit routine receives control after an uncorrectable I/O error has occurred, the third entry of the parameter list contains the address of the standard status information. The error flag (high-order byte of the third entry in the parameter list) is set as follows:

```
Bit 0 = 1: Uncorrectable I/O error
Bit 1 = 1: Error occurred during writing of updated label
Bits 2-7:  Reserved
```

The fourth entry in the parameter list is the address of the user totaling image area. This image area is the entry in the user totaling save area that corresponds to the last record physically written on the volume. (The image area is discussed further under "User Totaling for BSAM and QSAM" on page 80.)

Each routine must create or verify one label of a header or trailer label group, place a return code in register 15, and return control to the operating system. The operating system responds to the return code as shown in Figure 31.

You can create user labels only for data sets on magnetic tape volumes with IBM standard labels or ISO/ANSI/FIPS labels and for data sets on direct access volumes. When you specify both user labels and IBM standard labels in a DD statement by specifying LABEL=(,SUL) and there is an active entry in the exit list, a label exit is always taken. Thus, a label exit is taken even when an input data set does not contain user labels, or when no user label track has been allocated for writing labels on a direct access volume. In either case, the appropriate exit routine is entered with the buffer area address parameter set to 0. On return from the exit routine, normal processing is resumed; no return code is necessary.

Figure 31 (Page 1 of 2). System Response to a User Label Exit Routine Return Code

Routine Type	Return Code	System Response
Input header or trailer label	0 (X'00')	Normal processing is resumed. If there are any remaining labels in the label group, they are ignored.
	4 (X'04')	The next user label is read into the label buffer area and control is returned to the exit routine. If there are no more labels in the label group, normal processing is resumed.

Figure 31 (Page 2 of 2). System Response to a User Label Exit Routine Return Code

Routine Type	Return Code	System Response
	8[1] (X'08')	The label is written from the label buffer area and normal processing is resumed.
	12[1] (X'0C')	The label is written from the label area, the next label is read into the label buffer area, and control is returned to the label processing routine. If there are no more labels, processing is resumed.
Output header or trailer label	0 (X'00')	Normal processing is resumed; no label is written from the label buffer area.
	4 (X'04')	User label is written from the label buffer area. Normal processing is resumed.
	8 (X'08')	User label is written from the label buffer area. If fewer than eight labels have been created, control is returned to the exit routine, which then creates the next label. If eight labels have been created, normal processing is resumed.

Note to Figure 31:

[1] Your input label routines can return these codes only when you are processing a physical sequential data set opened for UPDAT or a direct data set opened for OUTPUT or UPDAT. These return codes allow you to verify the existing labels, update them if necessary, then request that the system write the updated labels.

Label exits are not taken for system output (SYSOUT) data sets, or for data sets on volumes that do not have standard labels. For other data sets, exits are taken as follows:

- When an input data set is opened, the input header label exit 01 is taken. If the data set is on tape being opened for RDBACK, user trailer labels will be processed.
- When an output data set is opened, the output header label exit 02 is taken. However, if the data set already exists and DISP=MOD is coded in the DD statement, the input trailer label exit 03 is taken to process any existing trailer labels. If the input trailer label exit 03 does not exist, then the deferred input trailer label exit 0C is taken if it exists; otherwise, no label exit is taken. For tape, these trailer labels will be overwritten by the new

output data or by EOV or close processing when writing new standard trailer labels. For direct access devices, these trailer labels will still exist unless rewritten by EOV or close processing in an output trailer label exit.

- When an input data set reaches EOV, the input trailer label exit 03 is taken. If the data set is on tape opened for RDBACK, header labels will be processed. The input trailer label exit 03 is not taken if you issue an FEOV macro. If a defer input trailer label exit 0C is present, and an input trailer label exit 03 is not present, the 0C exit is taken. After switching volumes, the input header label exit 01 is taken. If the data set is on tape opened for RDBACK, trailer labels will be processed.
- When an output data set reaches EOV, the output trailer label exit 04 is taken. After switching volumes, output header label exit 02 is taken.
- When an input data set reaches end-of-data, the input trailer label exit 03 is taken before the EODAD exit, unless the DCB exit list contains a defer input trailer label exit 0C.
- When an input data set is closed, no exit is taken unless the data set was previously read to end-of-data and the defer input trailer label exit 0C is present. If so, the defer input trailer label exit 0C is taken to process trailer labels, or if the tape is opened for RDBACK, header labels.
- When an output data set is closed, the output trailer label exit 04 is taken.

To process records in reverse order, a data set on magnetic tape can be read backward. When you read backward, header label exits are taken to process trailer labels, and trailer label exits are taken to process header labels. The system presents labels from a label group in ascending order by label number, which is the order in which the labels were created. If necessary, an exit routine can determine label type (UHL or UTL) and number by examining the first four characters of each label. Tapes with IBM standard labels and direct access devices can have as many as eight user labels. Tapes with ISO/ANSI/FIPS labels can have an unlimited number of user labels.

If an uncorrectable error occurs during reading or writing of a user label, the system passes control to the appropriate exit routine, with the third word of the parameter list flagged and pointing to status information.

After an input error, the exit routine must return control with an appropriate return code (0 or 4). No return code is required after an output error. If an output error occurs while the system is opening a data set, the data set is not opened (DCB is flagged) and control is returned to your program. If an output error occurs at any other time, the system attempts to resume normal processing.

Open/EOV Nonspecific Tape Volume Mount Exit

This user exit gives you the option of identifying a specific tape volume to be requested in place of a nonspecific (scratch) tape volume. An X'17' in the DCB exit list (EXLST) activates this exit. (See "EXLST Exit List" on page 58 for more information about EXLST.) This exit, which supports only IBM standard labeled tapes, was designed to be used with the Open/EOV volume security and verification user exit. However, this exit can be used by itself.

Open or EOV calls this exit when either must issue mount message IEC501A or EIC501E to request a scratch tape volume. Open issues the mount message if you specify the DEFER parameter with the UNIT option, and you either didn't specify a volume serial number in the DD statement or you specified 'VOL=SER=SCRTCH'. EOV always calls this exit for a scratch tape volume request.

This user exit gets control in the key and state of the program that issued the OPEN or EOV, and no locks are held. After you are in control, you must provide a return code in register 15.

Return Code	Meaning
00 (X'00')	Continue with the scratch tape request as if this exit had not been called.
04 (X'04')	Replace the scratch tape request with a specific volume serial number. Do this by loading the address of a 6-byte volume serial number into register 0.

Note: A value other than 0 or 4 in register 15 is treated as a 0.

If OPEN or EOV finds that the volume pointed to by register 0 is being used either by this or by another job (an active ENQ on this volume), it takes this exit again and continues to do so until you either specify another volume serial number or request a scratch volume. If the volume you specify is available but is rejected by OPEN or EOV for some reason (I/O errors, expiration date, password check, and so forth), this exit is not taken again.

When this exit gets control, register 1 points to the parameter list described by the IECOENTE macro. Figure 32 shows this parameter list.

```
      .....
+ OENTID   DS    CL4       PLIST ID ('OENT')
+ OENTFLG  DS    X         FLAG BYTES
+ OENTOEOV EQU   X'80'     0=OPEN, 1=EOV
+ OENTNTRY EQU   X'01'     0=1ST ENTRY ,1=SUBSEQUENT ENTRY
+ OENTOPTN DS    X         OPEN OPTION (OUTPUT/INPUT/...)
+ OENTMASK EQU   X'0F'     TO MASK OFF UNNECESSARY BITS
+ OENTRSVD DS    XL2       RESERVED
+ OENTDCBA DS    A         ADDRESS OF USER DCB
+ OENTVSRA DS    A         ADDRESS OF VOLSER
+ OENTJFCB DS    A         ADDRESS OF O/C/E COPY OF JFCB
+ OENTLENG EQU   *-&L      PLIST LENGTH
+ OENTREGS DS    6F        REGISTER SAVE AREA
+ OENTAREA EQU   *-&L      MACRO LENGTH
      .....
```

Figure 32. IECOENTE Macro Parameter List

OENTOEOV
set to 0 if OPEN called this exit; set to 1 if EOV called this exit.

OENTNTRY
set to 1 if this is not the first time this exit was called because the requested tape volume is being used by this or any other job.

OENTOPTN
contains the OPEN options from the DCB parameter list (OUTPUT, INPUT, OUTIN, INOUT, and so forth). For EOV processing, the options byte in the DCB parameter list indicates how EOV is processing this volume. For example, if you open a tape volume for INOUT and EOV is called during an input operation on this tape volume, the DCB parameter list and OENTOPTN are set to indicate INPUT.

OENTVSRA
points to the last volume serial number you requested in this exit but was in use either by this or another job. OENTVSRA is set to 0 the first time this exit is called.

OENTJFCB
points to the OPEN or EOV copy of the JFCB. The high order bit is always on, indicating that this is the end of the parameter list.

OENTREGS
starts the register save area used by OPEN or EOV. Do not use this save area in this user exit.

Convention for Saving and Restoring General Registers

When this user exit is entered, the general registers contain:

Register	Contents
0	Variable
1	Address of the parameter list for this exit
2-13	Contents of the registers before the OPEN or EOV was issued
14	Return address (you must preserve the contents of this register in this user exit)
15	Entry point address to this user exit

You do not have to preserve the contents of any register other than register 14. The operating system restores the contents of registers 2 through 13 before it returns to OPEN or EOV and before it returns control to the original calling program.

Do not use the save area pointed to by register 13; the operating system uses it. If you call another routine, or issue a supervisor or data management macro in this user exit, you must provide the address of a new save area in register 13.

Open/EOV Volume Security and Verification Exit

This user exit lets you verify that the volume that is currently mounted is the one you want. You can also use it to bypass the OPEN or EOV expiration date, password, and data set name security checks. An X'18' in the DCB exit list (EXLST) activates this exit. (See "EXLST Exit List" on page 58 for more information about EXLST.) This exit, which supports IBM standard label tapes,

was designed to be used with the OPEN/EOV nonspecific tape volume mount user exit. (See "Open/EOV Nonspecific Tape Volume Mount Exit" on page 75 for more information about that user exit.) However, this exit can be used by itself.

Note: This exit is available only for APF-authorized programs.

This user exit gets control in the key and state of the program that issued the OPEN or EOV request, and no locks are held. After you are in control, you must provide a return code in register 15.

Return Code	Meaning
00 (X'00')	Use this tape volume. Return to OPEN or EOV as if this exit had not been called.
04 (X'04')	Reject this volume and: • Output — If the data set is the first data set on the volume, request a scratch tape. This causes OPEN or EOV to issue demount message IEC502E for the rejected tape volume, and mount message IEC501A for a scratch tape volume. If the nonspecific tape volume mount exit is active, it is called. — If the data set is other than the first one on the volume, process this return code as if it were return code 08. • Input — Treat this return code as if it were return code 08.
08 (X'08')	Abnormally terminate OPEN or EOV unconditionally; no scratch tape request is issued. Open abnormally terminates with a 913-34 abend code, and EOV terminates with a 937-29 abend code.
12 (X'0C')	Use this volume without checking the data set's expiration date, but check its password and name. If the expiration date of the current data set is in effect, the new data set can still write over it.
16 (X'10')	Use this volume. A conflict with the password, label expiration date, or data set name does not prevent the new data set from writing over the current data set if it is the first one on the volume. To write over other than the first data set, the new data set must have the same level of security protection as the current data set.

When this exit gets control, register 1 points to the parameter list described by the IECOEVSE macro. The parameter list is shown in Figure 33 on page 79.

```
  .....
+ OEVSID   DS    CL4      ID FIELD = OEVS
+ OEVSFLG  DS    X        FLAGS BYTE
+ OEVSEOV  EQU   X'80'    0=OPEN, 1=EOV
+ OEVSFILE EQU   X'01'    0=1ST FILE, 1=SUBSEQ FILE
  *                       BITS 1 THROUGH 6 RESERVED
+ OEVSOPTN DS    X        OPEN OPTION (OUTPUT/INPUT/...)
+ OEVSMASK EQU   X'0F'    MASK
+ OEVSRSVD DS    XL2      RESERVED
+ OEVSDCBA DS    A        ADDRESS OF USER DCB
+ OEVSVSRA DS    A        ADDRESS OF 6-BYTE VOLSER
+ OEVSHDR1 DS    A        ADDRESS OF HDR1/EOF1
+ OEVSJFCB DS    A        ADDRESS OF O/C/E COPY OF JFCB
+ OEVSLENG EQU   *-&L     PLIST LENGTH
+ OEVSREGS DS    6F       REGISTER SAVE AREA
+ OEVSAREA EQU   *-&L     MACRO LENGTH
  .....
```

Figure 33. IECOEVSE Macro Parameter List

OEVSFLG
a flag field containing two flags.

OEVSEOV is set to 0 if OPEN called this exit; set to 1 if EOV called this exit.

OEVSFILE is set to 0 if the first data set on the volume is to be written; set to 1 if this is not the first data set on the volume to be written. This bit is always 0 for INPUT processing.

OEVSOPTN
a 1-byte field containing the OPEN options from the DCB parameter list (OUTPUT, INPUT, INOUT, and so forth). For EOV processing, this byte indicates how EOV is processing this volume. For example, if you opened a tape volume for OUTIN and EOV is called during an output operation on the tape volume, the DCB parameter list and OEVSOPTN are set to indicate OUTPUT.

OEVSVSRA
a pointer to the current volume serial number that OPEN or EOV is processing.

OEVSHDR1
a pointer to a HDR1 label, if one exists; or an EOF1 label, if you are creating other than the first data set on this volume.

OEVSJFCB
a pointer to the OPEN, CLOSE, or EOV copy of the JFCB. The high-order bit is always on, indicating that this is the end of the parameter list.

OEVSREGS
a register save area used by OPEN or EOV. Do not use this save area in this user exit.

Convention for Saving and Restoring General Registers

When this user exit is entered, the general registers contain:

Register	Contents
0	Variable
1	Address of the parameter list for this exit.
2-13	Contents of the registers before the OPEN or EOV was issued
14	Return address (you must preserve the contents of this register in this user exit)
15	Entry point address to this user exit

You do not have to preserve the contents of any register other than register 14. The operating system restores the contents of registers 2 through 13 before it returns to OPEN or EOV and before it returns control to the original calling program.

Do not use the save area pointed to by register 13; the operating system uses it. If you call another routine or issue a supervisor or data management macro in this user exit, you must provide the address of a new save area in register 13.

QSAM Parallel Input Exit

QSAM parallel input processing may be used to process two or more input data sets concurrently, such as sorting or merging several data sets at the same time.

A request for parallel input processing is indicated by including the address of a parallel data access block (PDAB) in the DCB exit list. The address must be on a fullword boundary with the first byte of the entry containing X'12' or, if it is the last entry, X'92'. For more information on parallel input processing, see *Data Administration Guide*.

User Totaling for BSAM and QSAM

When creating or processing a data set with user labels, you may develop control totals for each volume of the data set and store this information in your user labels.

For example, a control total that was accumulated as the data set was created can be stored in your user label and later compared with a total accumulated during processing of the volume. User totaling helps you by synchronizing the control data you create with records physically written on a volume. For an output data set without user labels, you can also develop a control total that will be available to your EOV routine.

To request user totaling, you must specify OPTCD=T in the DCB macro instruction or in the DCB parameter of the DD statement. The area in which you collect the control data (the user totaling area) must be identified to the control program by an entry of X'0A' in the DCB exit list. OPTCD=T cannot be specified for SYSIN or SYSOUT data sets.

The user totaling area, an area in storage that you provide, must begin on a halfword boundary and be large enough to contain your accumulated data plus a 2-byte length field. The length field must be the first 2 bytes of the area and specify the length of the complete area. A data set for which you have specified user totaling (OPTCD=T) will not be opened if either the totaling area length or the address in the exit list is 0, or if there is no X'0A' entry in the exit list.

The control program establishes a user totaling save area, where the control program preserves an image of your totaling area, when an I/O operation is scheduled. When the output user label exits are taken, the address of the save area entry (user totaling image area) corresponding to the last record physically written on a volume is passed to you in the fourth entry of the user label parameter list. (This parameter list is described in "Open/Close/EOV Standard User Label Exit" on page 72.) When an EOV exit is taken for an output data set and user totaling has been specified, the address of the user totaling image area is in register 0.

When using user totaling for an output data set, that is, when creating the data set, you must update your control data in your totaling area before issuing a PUT or a WRITE macro. The control program places an image of your totaling area in the user totaling save area when an I/O operation is scheduled. A pointer to the save area entry (user totaling image area) corresponding to the last record physically written on the volume is passed to you in your label processing routine. Thus you can include the control total in your user labels. When subsequently using this data set for input, you can collect the same information as you read each record and compare this total with the one previously stored in the user trailer label. If you have stored the total from the preceding volume in the user header label of the current volume, you can process each volume of a multivolume data set independently and still maintain this system of control.

When variable-length records are specified with the totaling function for user labels, special considerations are necessary. Because the control program determines whether a variable-length record will fit in a buffer after a PUT or a WRITE has been issued, the total you have accumulated may include one more record than is really written on the volume. For variable-length spanned records, the accumulated total will include the control data from the volume-spanning record although only a segment of the record is on that volume. However, when you process such a data set, the volume-spanning record or the first record on the next volume will not be available to you until after the volume switch and user label processing are completed. Thus the totaling information in the user label may not agree with that developed during processing of the volume.

One way you can resolve this situation is to maintain, when you are creating a data set, control data pertaining to each of the last two records and include both totals in your user labels. Then the total related to the last complete record on the volume and the volume-spanning record or the first record on the next volume would be available to your user label routines. During subsequent processing of the data set, your user label routines can determine if there is agreement between the generated information and one of the two totals previously saved.

When the totaling function for user labels is selected with DASD devices and secondary space is specified, the total accumulated may be one less than the actual written.

Chapter 6. User Exit Routines Specified with Utilities

General Guidance

Exits can be specified with various utilities to:

- Modify physical records
- Handle I/O errors
- Process user input/output header and trailer labels.

For more information about utilities, see *Utilities*.

The exits are specified in a parameter of the EXITS statement in the various utilities. The exits available from utility programs are listed in Figure 34 on page 84.

Figure 34. User-Exit Routines Specified in Utilities

Exit Routine	When Available	Where Specified
Modify physical records before processing by IEBGENER	After the physical record is read and before any editing is performed	DATA parameter of IEBGENER
Input header or trailer label	When the data set is opened for input (header) or closed (trailer)	INHDR/INTLR parameters of IEBCOMPR, IEBPTPCH, IEBGENER
Output header or trailer label	When the data set is opened for output (header) or closed (trailer)	OUTHDR/OUTLR parameters of IEBCOMPR, IEBGENER
Totaling	Prior to IEBGENER writing of each physical record (sequential data sets only)	TOTAL parameter of IEBGENER
I/O error	When permanent error occurs in IEBGENER	IOERROR parameter of IEBGENER
Error detected by IEBCOMPR	After unequal comparison	ERROR parameter of IEBCOMPR
Build output record key	Prior to IEBGENER writing of a record	KEY of IEBGENER
Process logical records of input data sets before compared	Before input records are processed by IEBCOMPR	PRECOMP parameter of IEBCOMPR
Process IEBPTPCH input/output records	Before logical record is processed (INREC) or before logical record is written (OUTREC)	INREC/OUTREC parameter of IEBPTPCH
Analyze or modify IEBDG output record	After output record is constructed, but before it is placed in the output data set	CREATE parameter of IEBDG

Register Contents at Entry to Routines from Utility Programs

Register	Contents
1	Address of the parameter list.
13	Address of the register save area. The save area must not be used by user label processing routines.
14	Return address to utility.
15	Entry address to the exit routine.

Figure 35. Register Contents at Entry to Utility Exit Routines

Programming Considerations

The exit routine must reside in either the job library or link library.

Returning from an Exit Routine

An exit routine returns control to the utility program by means of the RETURN macro instruction in the exit routine. Registers 1 through 14 must be restored before control is returned to the utility program.

The format of the RETURN macro instruction is:

[*label*]	**RETURN**	[(r,r)] [,**RC**=*n*|**(15)**]

where:

(r,r)
: specifies the range of registers, from 0 to 15, to be reloaded by the utility program from the register save area. For example, (14,12) indicates that all registers except register 13 are to be restored. If this parameter is omitted, the registers are considered properly restored by the exit routine.

RC=
: specifies a decimal return code in register 15. If RC is omitted, register 15 is loaded as specified by (r,r).

  RC values can be coded:

  n
  : specifies a return code to be placed in the 12 low order bits of register 15.

  (15)
  : specifies that general register 15 already contains a valid return code.

The user's label processing routine must return a code in register 15 as shown in Figure 36 unless:

- The buffer address was set to zero before entry to the label processing routine. In this case, the system resumes normal processing regardless of the return code.
- The user's label processing routine was entered after an uncorrectable output error occurred. In this case the system attempts to resume normal processing.

Figure 36 shows the return codes that can be issued to utility programs by user exit routines. Slightly different return codes are used for the UPDATE=INPLACE option of the IEBUPDTE program. (See *Utilities* for more information).

Figure 36 (Page 1 of 2). Return Codes That Must Be Issued by User Exit Routines

Type of Exit	Return Code	Action
Input Header or Trailer Label	0	The system resumes normal processing. If there are more labels in the label group, they are ignored.
	4	The next user label is read into the label buffer area and control is returned to the user's routine. If there are no more labels, normal processing is resumed.
	16	The utility program is terminated on request of the user routine.
Output Header or Trailer Label	0	The system resumes normal processing. No label is written from the label buffer area.
	4	The user label is written from the label buffer area. The system then resumes normal processing.
	8	The user label is written from the label buffer area. If fewer than eight labels have been created, the user's routine again receives control so that it can create another user label. If eight labels have been created, the system resumes normal processing.
	16	The utility program is terminated on request of the user routine.
Totaling Exits	0	Processing continues, but no further exits are taken.
	4	Normal operation continues.
	8	Processing ceases, except for EOD processing on output data set (user label processing).
	16	Utility program is terminated.
All other exits (except IEBPTPCH's exit OUTREC)	0-11 (Set to next multiple of four)	Return code is compared to highest previous return code; the higher is saved and the other discarded. At the normal end of job, the highest return code is passed to the calling processor.
	12 or 16	Utility program is terminated and this return code is passed to the calling processor.

Figure 36 (Page 2 of 2). Return Codes That Must Be Issued by User Exit Routines

Type of Exit	Return Code	Action
ERROR	0	Record is not placed in the error data set. Processing continues with the next record.
	4	Record is placed in the error data set (SYSUT3).
	8	Record is not placed in error data set but is processed as a valid record (sent to OUTREC and SYSUT2 if specified).
	16	Utility program is terminated.
OUTREC (IEBPTPCH)	4	Record is not placed in normal output data set.
	12 or 16	Utility program is terminated.
	Any other number	Record is placed in normal output data set (SYSUT2).

Parameters Passed to Label Processing Routines

The parameters passed to a user's label processing routine are addresses of: the 80-byte label buffer, the DCB being processed, the status information if an uncorrectable input/output error occurs, and the totaling area.

The 80-byte label buffer contains an image of the user label when an input label is being processed. When an output label is being processed, the buffer contains no significant information at entry to the user's label processing routine. When the utility program has been requested to generate labels, the user's label processing routine must construct a label in the label buffer.

If standard user labels (SUL) are specified on the DD statement for a data set, but the data set has no user labels, the system still takes the specified exits to the appropriate user's routine. In such a case, the user's input label processing routine is entered with the buffer address parameter set to zero.

The format and content of the DCB are presented in *Data Administration: Macro Instruction Reference*.

Bit 0 of flag 1 in the DCB-address parameter is set to a value of 0 except when:

- Volume trailer or header labels are being processed at volume switch time.
- The trailer labels of a MOD data set are being processed (when the data set is opened).

If an uncorrectable input/output error occurs while reading or writing a user label, the appropriate label processing routine is entered with bit 0 of flag 2 in the status information address parameter set on. The three low order bytes of this parameter contain the address of standard status information as supplied for SYNAD routines. (The SYNAD routine is not entered.)

Parameters Passed to Nonlabel Processing Routines

Figure 37 shows the programs from which exits can be taken to nonlabel processing routines, the names of the exits, and the parameters available for each exit routine.

Figure 37. Parameter Lists for Nonlabel Processing Exit Routines

Program	Exit	Parameters
IEBGENER	KEY	Address at which key is to be placed (record follows key); address of DCB.
	DATA	Address of SYSUT1 record; address of DCB.
	IOERROR	Address of DECB; cause of the error and address of DCB. (Address in lower order three bytes and cause of error in high order byte.)
IEBCOMPR	ERROR	Address of DCB for SYSUT1; address of DCB for SYSUT2.[1]
	PRECOMP	Address of SYSUT1 record; length of SYSUT1 record, address of SYSUT2 record; length of SYSUT2 record.
IEBPTPCH	INREC	Address of input record; length of the input record.
	OUTREC	Address of output record; length of the output record.

Note to Figure 37:

[1] The IOBAD pointer in the DCB points to a location that contains the address of the corresponding data event control block (DECB) for these records. The format of the DECB is illustrated in "Status Information Following an Input/Output Operation" on page 42.

Processing User Labels

User labels can be processed by IEBCOMPR, IEBGENER, IEBPTPCH, IEBUPDTE, and IEHMOVE. In some cases, user-label processing is automatically performed; in other cases, you must indicate the processing to be performed. In general, user label support allows the utility program user to:

- Process user labels as data set descriptors.
- Process user labels as data.
- Total the processed records prior to each WRITE command (IEBGENER and IEBUPDTE only).

For either of the first two options, the user must specify SUL on the DD statement that defines each data set for which user-label processing is desired. For totaling routines, OPTCD=T must be specified on the DD statement.

The user cannot update labels by means of the IEBUPDTE program. This function must be performed by a user's label processing routines. IEBUPDTE will, however, allow you to create labels on the output data set from data

supplied in the input stream. For more information on the IEBUPDTE program, see *Utilities*.

IEHMOVE does not allow exits to user routines and does not recognize options concerning the processing of user labels as data. IEHMOVE always moves or copies user labels directly to a new data set. For more information about IEHMOVE, see *Utilities*.

Volume switch labels of a multivolume data set cannot be processed by IEHMOVE, IEBGENER, or IEBUPDTE. Volume switch labels are therefore lost when these utilities create output data sets. To ensure that volume switch labels are retained, process multivolume data sets one volume at a time.

Processing User Labels as Data Set Descriptors

When user labels are to be processed as data set descriptors, one of the user's label processing routines receives control for each user label of the specified type. The user's routine can include, exclude, or modify the user label. Processing of user labels as data set descriptors is indicated on an EXITS statement with keyword parameters that name the label processing routine to be used.

The EXIT keyword parameters indicate that a user routine should receive control each time the OPEN, EOV, or CLOSE routine encounters a user label of the type specified.

Figure 38 on page 90 illustrates the action of the system at OPEN, EOV, or CLOSE time. When OPEN, EOV, or CLOSE recognizes a user label and when SUL has been specified on the DD statement for the data set, control is passed to the utility program. Then, if an exit has been specified for this type of label, the utility program passes control to the user routine. The user's routine processes the label and returns control, along with a return code, to the utility program. The utility program then returns control to OPEN, EOV, or CLOSE.

This cycle is repeated up to eight times, depending upon the number of user labels in the group and the return codes supplied by the user's routine.

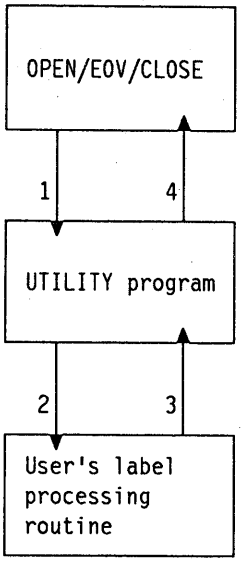


Figure 38. System Action at OPEN, EOV, or CLOSE Time

Exiting to a User's Totaling Routine

When an exit is taken to a user's totaling routine, an output record is passed to the user's routine just before the record is written. The first halfword of the totaling area pointed to by the parameter contains the length of the totaling area, and should not be used by the user's routine. If the user has specified user label exits, this totaling area (or an image of this area) is pointed to by the parameter list passed to the appropriate user label routine.

An output record is defined as a physical record (block), except when IEBGENER is used to process and reformat a data set that contains spanned records.

The code returned by the user's totaling routine determines system response as shown in Figure 39.

Codes	Meaning
00 (X'00')	Processing is to continue, but no further exits are to be taken.
04 (X'04')	Normal processing is to continue.
08 (X'08')	Processing is to terminate, except for EOD processing on the output data set (user label processing).
16 (X'10')	Processing is to be terminated.

Figure 39. User Totaling Routine Return Codes

Processing User Labels as Data

When user labels are processed as data, the group of user labels, as well as the data set, is subject to the normal processing done by the utility program. The user can have labels printed or punched by IEBPTPCH, compared by IEBCOMPR, or copied by IEBGENER.

To specify that user labels are to be processed as data, include a LABELS statement in the job step that is to process user labels as data.

There is no direct relationship between the LABELS statement and the EXITS statement. Either or both can appear in the control statement stream for an execution of a utility program. If there are user label-processing routines, however, their return codes may influence the processing of the labels as data. In addition, a user's output label-processing routine can override the action of a LABELS statement because it receives control before each output label is written. At this time, the label created by the utility as a result of the LABELS statement is in the label buffer, and the user's routine can modify it.

Using the CREATE Statement

IEBDG provides a user exit so you can provide your own routine to analyze or further modify a newly constructed record before it is placed in the output data set.

The CREATE statement defines the contents of a record (or records) to be made available to a user routine or to be written directly as an output record (or records).

The format of the CREATE statement is:

[*label*]	**CREATE**	[**QUANTITY**=*number*] [,**FILL**={'*character*'|**X**'*2-hex-digits*'}] [,**INPUT**=*ddname*|**SYSIN**[(*cccc*)]] [,**PICTURE**=*length*,*startloc*{,'*character-string*'| ,**P**'*decimal-number*'| [,**FILL**={'*character*'|**X**'*2-hex-digits*'}] [,**INPUT**=*ddname*|**SYSIN**[(*cccc*)]] ,**B**'*decimal-number*'}] [,**PICTURE**=*length*,*startloc*{,'*character-string*'| [,**NAME**=*name*|(*name1*,*namen...*)| ,**P**'*decimal-number*'| (*name*(**COPY**=*number*,*name1*,*namen...*),...)] [,**EXIT**=*routinename*]

After processing each potential output record, the user routine should provide a return code in register 15 to instruct IEBDG how to handle the output record. The user codes are listed below.

Codes	Meaning
00 (X'00')	The record is to be written.
04 (X'04')	The record is not to be written. The skipped record is not to be counted as a generated output record; processing is to continue as though a record were written. If skips are requested through user exits and input records are supplied, each skip causes an additional input record to be processed in the generation of output records. For example, if a CREATE statement specifies that 10 output records are to be generated and a user exit indicates that two records are to be skipped, 12 input records are processed.
12 (0C)	The processing of the remainder of this set of utility control statements is to be bypassed. Processing is to continue with the next DSD statement.
16 (10)	All processing is to halt.

Figure 40. IEBDG User Exit Return Codes

When an exit routine is loaded and you return control to IEBDG, register 1 contains the address of the first byte of the output record. Each keyword should appear no more than once on any CREATE statement.

Figure 41 shows the addition of field X to two different records. In record 1, field X is the first field referred to by the CREATE statement; therefore, field X begins in the first byte of the output record. In record 2, two fields, field A and field B, have already been referred to by a CREATE statement; field X, the next field referred to, begins immediately after field B. Field X does not have a special starting location in this example.

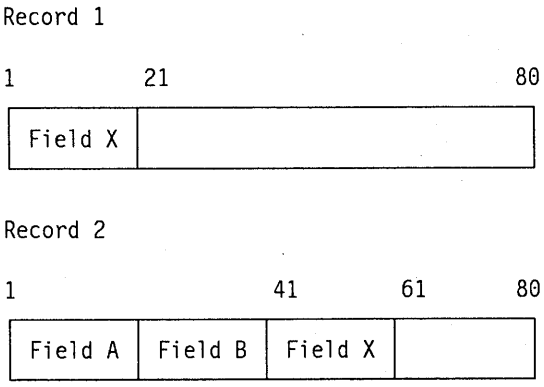


Figure 41. Default Placement of Fields within an Output Record Using IEBDG

You can also indicate that a numeric field is to be modified after it has been referred to *n* times by a CREATE statement or statements, that is, after *n* cycles, a modification is to be made. A modification will add a user-specified number to a field.

The CREATE statement constructs an output record by referring to previously defined fields by name and/or by providing a picture to be placed in the record. You can generate multiple records with a single CREATE statement.

When defining a picture in a CREATE statement, you must specify its length and starting location in the output record. The specified length must be equal to the number of specified EBCDIC or numeric characters. (When a specified decimal number is converted to packed decimal or binary, it is automatically right-aligned.)

Figure 42 on page 94 shows three ways in which output records can be created from utility control statements.

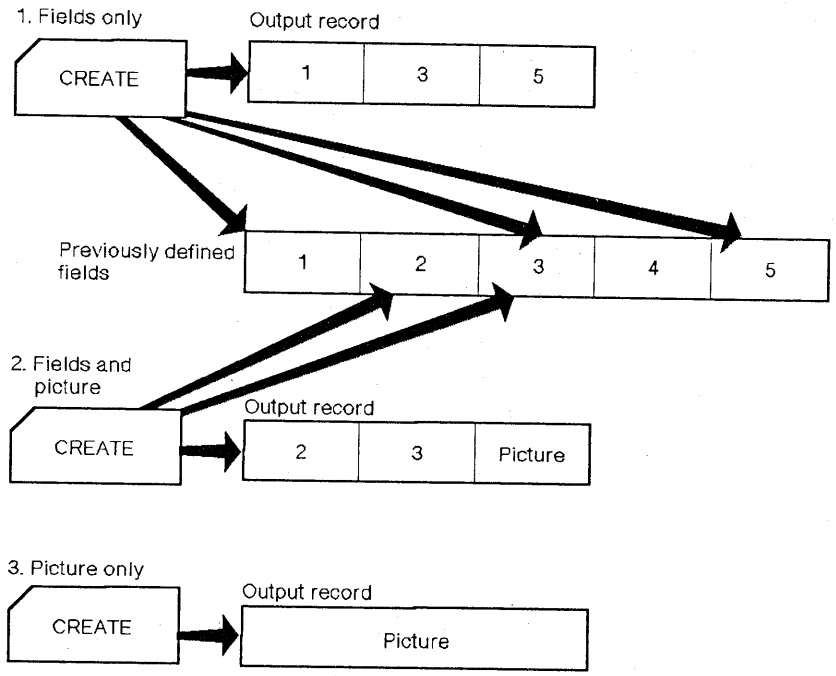


Figure 42. Creating Output Records with Utility Control Statements

As an alternative to creating output records from utility control statements alone, you can provide input records, which can be modified and written as output records. Input records can be provided directly in the input stream, or in a separate data set. Only one input data set can be read for each CREATE statement.

As previously mentioned, the CREATE statement is responsible for the construction of an output record. An output record is constructed in the following order:

1. A fill character, specified or default (binary zero), is initially loaded into each byte of the output record.
2. If the INPUT operand is specified on the CREATE statement, and not on an FD statement, the input records are left-aligned in the corresponding output record.
3. If the INPUT operand specifies a *ddname* in any FD statement, only the fields described by the FD statement(s) are placed in the output record.
4. FD fields, if any, are placed in the output record in the order of the appearance of their names in the CREATE statement.
5. A CREATE statement picture, if any, is placed in the output record.

A set of utility control statements contains one DSD statement, any number of FD, CREATE, and REPEAT statements, and one END statement when the INPUT parameter is omitted from the FD card.

When selecting fields from an input record (FD INPUT=ddname), the field must be defined by an FD statement within each set of utility control statements. In that case, defined fields for field selection are not usable across sets of utility control statements; such an FD card may be duplicated and used in more than one set of utility control statements within the job step.

Chapter 7. EXCP Appendages

General Guidance

An appendage is a programmer-written routine that provides additional control over I/O operations. By using appendages, you can examine the status of I/O operations and determine the actions to be taken for various conditions. An appendage may receive control when one of the following occurs:

- EXCP SVC
- Program-controlled interrupt
- End of extent
- Channel end
- Abnormal end.

EXCP appendages are shown in Figure 43.

Note: EXCP is not recommended for I/O operations with data storage devices.

Figure 43. EXCP Appendages

Appendage	Description	When Available
ABE	Abnormal-end	Abnormal conditions
CHE	Channel-end	Channel-end, unit exception, wrong-length record
EOE	End-of-extent	Address in I/O block outside allocated extent limits
PCI	Program-controlled interruption	When one or more PCI bits are on in a channel program
SIO	Start-I/O	EXCP processor right before translating channel program

Appendages get control in supervisor state, receiving the following pointers from the EXCP processor (see Figure 44).

Register	Contents
1	Points to the request queue element for the channel program
2	Points to the input/output block (IOB)
3	Points to the data extent block (DEB)
4	Points to the data control block (DCB)
6	Points to the seek address if control is given to an end-of-extent (EOE) appendage
7	Points to the unit control block (UCB)
13	Points to a 16-word area you can use to save input registers or data
14	Points to the location in the EXCP processor where control is to be returned after execution of an appendage. When returning control to the EXCP processor, you may use displacements from the return address in register 14. Allowable displacements are summarized in Figure 45, and described later for each appendage.
15	Points to the entry point of the appendage

Figure 44. Contents of Registers at Entry to EXCP Appendages

The processing done by appendages is subject to these requirements and restrictions:

- Register 9, if used, must be set to binary zeros before control is returned to the system. All other registers, except those indicated in the descriptions of each appendage, must be saved and restored if you use them. Figure 45 summarizes register conventions.
- No SVC instructions or instructions that change the status of the system (for example, WTO, LPSW, or any privileged instructions) can be issued.
- Loops that test for the completion of I/O operations must not be used.
- Storage used by the I/O supervisor or EXCP processor must not be altered.

The types of appendages are described in the following sections, with explanations of when they are created, how they return control to the system, and which registers they may use without saving and restoring their contents.

Note: All user-specified appendages are given control in 24-bit addressing mode and must return in the same mode.

Figure 45. Entry Points, Returns, and Available Work Registers for Appendages

Appendage	Entry Point	Returns		Available Work Registers
EOE	Reg 15	Reg 14 + 0 Reg 14 + 4 Reg 14 + 8	Return Skip Try Again	Reg. 10, 11, 12, and 13
SIO	Reg 15	Reg 14 + 0 Reg 14 + 4	Normal Skip	Reg. 10, 11, and 13
PCI	Reg 15	Reg 14 + 0	Normal	Reg. 10, 11, 12, and 13
CHE	Reg 15	Reg 14 + 0 Reg 14 + 4 Reg 14 + 8 Reg 14 + 12	Normal Skip Re-EXCP By-Pass	Reg. 10, 11, 12, and 13
ABE	Reg 15	Reg 14 + 0 Reg 14 + 4 Reg 14 + 8 Reg 14 + 12	Normal Skip Re-EXCP By-Pass	Reg. 10, 11, 12, and 13

Note to Figure 45: Certain register conventions for passing parameters from appendages to the EXCP processor must be followed. These conventions are described in the individual appendage descriptions.

Making Your Appendages Part of the System

Before your appendages can be executed, they must become members of either the SYS1.LPALIB or SYS1.SVCLIB data set. There are two ways to put appendages into SYS1.LPALIB or SYS1.SVCLIB: They can be included at system generation using the DATASET macro instruction (a full explanation appears in *System Generation*), or they can be link-edited into SYS1.LPALIB or SYS1.SVCLIB after the system has been generated. Each appendage must have an 8-character member name, the first six characters being IGG019 and the last two being anything in the range from WA to Z9. Note, however, if your program runs in a V=R address space and uses a PCI appendage, the PCI appendage and any appendage that the PCI appendage refers to must be placed in either SYS1.SVCLIB or the fixed link pack area (LPA). For information on providing a list of programs to be fixed in storage, see *Initialization and Tuning*.

The Authorized Appendage List (IEAAPP00)

If an "unauthorized" program opens a DCB to be used with an EXCP macro instruction, the names of any appendages associated with the DCB must be listed in the IEAAPP00 member of SYS1.PARMLIB. (An "unauthorized" program is one that runs in a protection key greater than 7 and has not been marked as authorized by the Authorized Program Facility.)

If your appendages were put in SYS1.LPALIB or SYS1.SVCLIB at system generation, their names are automatically put in IEAAPP00. If your appendages were added to SYS1.LPALIB or SYS1.SVCLIB after system generation, you can

add IEAAPP00 to SYS1.PARMLIB and put the names of the appendages in it in one job step with the IEBUPDTE utility.

Here is an example of JCL statements and IEBUPDTE input that will add IEAAPP00 to SYS1.PARMLIB and put the names of one EOE appendage, two SIO appendages, two CHE appendages, and one ABE appendage in IEAAPP00:

```
//              JOB     ...
//              EXEC    PGM=IEBUPDTE
//SYSPRINT      DD      SYSOUT=A
//SYSUT2        DD      DSN=SYS1.PARMLIB,DISP=OLD
//SYSIN         DD      *
./              ADD     NAME=IEAAPP00,LIST=ALL
EOEAPP WA,
SIOAPP X1,X2,
CHEAPP Z3,Z4,
ABEAPP Z2
/*
```

Note the following about the IEBUPDTE input:

- The type of appendage is identified by six characters that begin in column 1. EOEAPP identifies an EOE appendage, SIOAPP an SIO appendage, CHEAPP a CHE appendage, and ABEAPP an ABE appendage. (The PCI appendage identifier, PCIAPP, is not shown, because the example does not add a PCI appendage name to IEAAPP00.)
- Only the last two characters in an appendage's name are specified, beginning in column 8.
- Each statement that identifies one or more appendage names ends in a comma, except the last statement.

You can also use IEBUPDTE to add appendage names later or to delete appendage names. Here is an example of JCL statements and IEBUPDTE input that adds the names of a PCI and an ABE appendage to the IEAAPP00 appendage list that was created in the preceding example, and deletes the name of an SIO appendage from that list:

```
//              JOB. . .
//              EXEC    PGM=IEBUPDTE
//SYSPRINT      DD      SYSOUT=A
//SYSUT2        DD      DSN=SYS1.PARMLIB,DISP=OLD
//SYSIN         DD      *
./              REPL    NAME=IEAPP00,LIST=ALL
PCIAPP Y1,
EOEAPP WA,
SIOAPP X1,X2,
CHEAPP Z3,Z4,
ABEAPP Z2,Z4
/*
```

Note the following about the IEBUPDTE input:

- The command to IEBUPDTE in this case is REPL (replace).
- All the appendage names that are to remain in IEAAPP00 are repeated.
- IGG019Z4 is both a CHE and an ABE appendage.

Abnormal-End (ABE) Appendage

This appendage may be entered on abnormal conditions, such as unit check, unit exception, wrong-length indication, program check, protection check, channel data check, channel control check, interface control check, chaining check, out-of-extent error, and intercept condition (that is, device end error). It may also be entered when an EXCP is issued for a request queue element that has already been purged.

1. When this appendage is entered because of a unit exception or wrong-length record indication or both, IOBECBCC is set to X'41'. For further information on these conditions, see "Channel-End (CHE) Appendage" on page 100.
2. When the appendage is entered because of an out-of-extent error, the IOBECBCC is set to X'42'.
3. When this appendage is entered with IOBECBCC set to X'4B', it is because of:
   a. The tape error recovery procedure (ERP) encountering an unexpected load point, or
   b. The tape ERP finding zeros in the command address field of the CSW.
4. When the appendage is first entered because of an intercept condition, the IOBECBCC is set to X'7E'. If it is then determined that the error condition is permanent, the appendage will be entered a second time with the IOBECBCC set to X'44'. The intercept condition signals that an error was detected at device end after channel end on the previous request.
5. When the appendage is entered because of an EXCP being issued to an already purged request queue element, this request will enter the ABE appendage with the IOBECBCC set to X'48'. This applies only to related requests.
6. If the appendage is entered with IOBECBCC set to X'7F', it may be because of a unit check, program check, protection check, channel data check, channel control check, interface control check, or chaining check. If the IOBECBCC is X'7F', it is the first detection of an error in the associated channel program. If the IOBEX flag (bit 5 of the IOBFLAG1) is on, the IOBECBCC field will contain X'41', X'42', X'48', X'4B', or X'4F', indicating a permanent I/O error.

To determine if an error is permanent, you should check the IOBECBCC field of the IOB. To determine the type of error, check the channel status word field and the sense information in the IOB. However, when the IOBECBCC is X'42', X'48', or X'4F', these fields are not applicable. For X'44', the CSW is applicable, but the sense is valid only if the unit check bit is set.

If you use the return address in register 14 to return control to the system, the channel program is posted complete, and its request element is made available. You may use the following optional return addresses:

- Contents of register 14 plus 4: The channel program is not posted complete, but its request element is made available. You may post the channel program by using the calling sequence described under the SIO (start-I/O) appendage.
- Contents of register 14 plus 8: The channel program is not posted complete, and its request element is placed back on the request queue so that the request can be retried. Reinitialize the IOBFLAG1, IOBFLAG2, and IOBFLAG3 fields of the input/output block and set the IOBERRCT field to zero. As an added precaution, clear the IOBSENS0, IOBSENS1, and IOBCSW fields.
- Contents of register 14 plus 12: The channel program is not posted complete, and its request element is not made available. (This return must be used if, and only if, the appendage has passed the RQE to the exit effector for use in scheduling an asynchronous routine.)

You may use registers 10 through 13 in an ABE appendage without saving and restoring their contents.

Channel-End (CHE) Appendage

This appendage is entered when a channel end (CHE), unit exception (UE) with or without channel end or when channel end with wrong-length record (WLR) occurs without any other abnormal-end conditions.

If you use the return address in register 14 to return control to the EXCP processor, the channel program is posted complete, and its request element is made available. In the case of unit exception or wrong-length record, the ERP is performed before the channel program is posted complete, and the IOBEX flag (X'04') in IOBFLAG1 is set on. The CSW status may be obtained from the IOBCSW field.

If the appendage takes care of the wrong-length record or unit exception or both, it may turn off the IOBEX (X'04') flag in IOBFLAG1 and return normally. The event will then be posted complete (completion code X'7F' under normal conditions, taken from the high-order byte of the IOBECBCC field). If the appendage returns normally without resetting the IOBEX flag to zero, the request will be routed to the associated device ERP, and the ABE (abnormal-end) appendage will then be entered with the completion code in IOBECBCC set to X'41' if the ERP could not correct the error. (See Step 1 of "Abnormal-End (ABE) Appendage" on page 99.)

You may use the following optional return addresses:

- Contents of register 14 plus 4: The channel program is not posted complete, but its request element is made available. You may post the channel program by using the calling sequence described under the SIO (start-I/O) appendage. This is especially useful if you want to post an ECB other than the ECB in the input/output block.

- Contents of register 14 plus 8: The channel program is not posted complete, and its request element is placed back on the request queue so that the I/O operation can be retried. For correct reexecution of the channel program, you must reinitialize the IOBFLAG1, IOBFLAG2, and IOBFLAG3 fields of the input/output block and set the "Error Counts" field to zero. As an added precaution, the IOBSENS0, IOBSENS1, and IOBCSW fields should be cleared.
- Contents of register 14 plus 12: The channel program is not posted complete, and its request element is not made available. (This return must be used if, and only if, the appendage has passed the RQE to the exit effector for use in scheduling an asynchronous routine. For information on the exit effector, see *MVSCP*.)

You may use registers 10 through 13 in a CHE appendage without saving and restoring their contents.

End-of-Extent (EOE) Appendage

This appendage is entered when the seek address specified in the input/output block is outside the allocated extent limits indicated in the data extent block.

If you use the return address in register 14 to return control to the system, the ABE appendage is entered. An end-of-extent error code (X'42') is placed in the "ECB code" field of the input/output block for subsequent posting in the ECB.

You may use the following optional return addresses:

- Contents of register 14 plus 4: The channel program is posted complete; its request element is returned to the available queue.
- Contents of register 14 plus 8: The request is tried again.

You may use registers 10 through 13 in an EOE appendage without saving and restoring their contents.

Note: If an end-of-cylinder or file-protect condition occurs, the EXCP processor updates the seek address to the next higher cylinder or track address and reexecutes the request. If the new seek address is within the data set's extent, the request is executed; if the new seek address is not within the data set's extent, the EOE appendage is entered. If you want to try the request in the next extent, you must move the new seek address to the location pointed to by register 6.

If a file protect is caused by a full seek (command code=07) embedded within a channel program, the request is flagged as a permanent error, and the ABE appendage is entered.

Program-Controlled Interruption (PCI) Appendage

This appendage is entered at least once if the channel finds one or more PCI bits on in a channel program. It may be entered as many times as the channel finds PCI bits on. Before the appendage is entered, the contents of the subchannel status word are placed in the "channel status word" field of the input/output block.

A PCI appendage is reentered if an ERP is retrying a channel program in which a PCI bit is on. The IOB error flag is set when the ERP is in control (IOBFLAG1 = X'20'). (For special PCI conditions encountered with command retry, see "Channel Programming Considerations," *System—Data Administration*.)

To post the channel program from a PCI appendage, the procedure described for the SIO appendage is used if the step is running ADDRSPC=VIRT or an authorized program is running V=R. If the step is running ADDRSPC=REAL and an authorized program issued the EXCP request or if SVC 114(EXCPVR) was issued, the PCI appendage uses real storage addresses, and the following procedure is used to post the channel program from the PCI appendage.

1. Put the completion code in register 10 and place X'80' in the high-order byte to indicate the key is in register 0 (step 5).
2. Put X'80' in the high-order byte of register 11 and the address of the ECB in the low-order bytes.
3. Put X'80' in the high-order byte of register 12 and the address of a BR 14 instruction in the low-order bytes. This BR 14 must be in storage addressable from any address space (for example, CVTBRET) and must be in storage addressable by 24 bits. Note that only registers 9 and 14 are restored when you use this option.
4. Put the address of the ASCB in register 13.
5. Put the requester's key in register 0.
6. Put the address of the post routine (found at CVTOPT01 in the communications vector table) in register 15.
7. Go to the post routine using BALR 14,15. Upon return, only registers 9 and 14 are valid. For more information on the POST routine, see *Application Development Macro Reference*.

This procedure can be used even if the PCI appendage uses virtual storage addresses, but performance may be slightly slower.

To return control to the EXCP processor for normal interruption processing, use the return address in register 14.

Start-I/O (SIO) Appendage

Unless an ERP is in control, the EXCP processor passes control to the SIO appendage just before the EXCP processor translates your channel program.

Optional return vectors give the I/O requester the following choices:

Reg. 14 + 0
: Normal return. Normal channel program translation and initiation of I/O.

Reg. 14 + 4
: Skip the I/O operation. The channel program is not posted complete, but the request queue element is made available. You may post the channel program as follows:

1. Save necessary registers.
2. Put the address of the post routine (found at CVT0PT01 in the communications vector table) in register 15.
3. Place the ECB address from the IOB in register 11.
4. Set the completion code in register 10. These are the four bytes of an ECB.
5. Go to the post routine pointed to by the CVT, using BALR 14,15.

Chapter 8. Interactive Storage Management Facility (ISMF)

General Guidance

ISMF helps you manage data and storage interactively. It is designed to use the space management, backup/recovery, and sorting functions provided by Data Facility Hierarchical Storage Manager (DFHSM), Data Facility Data Set Services (DFDSS), Device Services Facility (ICKDSF), and Data Facility Sort (DFSORT) to perform a variety of tasks. As an ISPF application, ISMF has a structure that is modeled after ISPF. Because ISMF was written using the procedures described in *ISPF Dialog Management Services*, it can be modified using the same techniques. DFDSS/ISMF, ICKDSF/ISMF, DFSORT/ISMF, and DFHSM/ISMF can be modified with these techniques.

Restrictions to Customizing

Before we talk about what you can change about ISMF, there are three guidelines you should keep in mind:

1. Before you change anything you should make a backup copy of ISMF. Keep this unmodified version of the product for diagnostic purposes. IBM support and maintenance is provided for **only** the unmodified version of ISMF.
2. Do not delete or rename any of the parts of ISMF. Deleting or renaming a part could severely impact processing, or cause ISMF to fail.
3. ISMF is copyrighted. Under the IBM licensing agreement you may modify ISMF for your own use. You may not modify it for commercial resale.

Other restrictions apply to the individual parts. These are presented with the detailed descriptions of how to modify each part on pages 108 through 131.

The Parts of ISMF That You Can Customize

ISMF allows you to customize the following parts for all ISMF applications:

Panels
Messages
Job skeletons
Command tables
CLIST

They are shipped in individual libraries. The changes you can make to each library are discussed on page 106.

The Panel Library

ISMF allows you to make the following changes to the panel library:

Change the initial priming values that ISMF ships
Change the default values for data entry panels
Provide additional restrictions to values entered for certain fields on panels
Remove fields from functional panels
Change highlighting and color
Change the format of the panel
Modify existing functional panel text and help text
Add new fields to panels
Add new panels

The Message Library

In the message library you can modify existing messages and add new messages.

The Skeleton Library

In the skeleton library you can modify the job skeletons for ISMF commands and line operators.

The Table Library

In the table library you can modify the ISPF command tables.

The Load Library

In the load library you can modify the ISMF command and line operator tables. The tables are contained in nonexecutable CSECTs in the load library.

The CLIST Library

In the CLIST library you can modify the options on the CLIST CONTROL statement.

Finding the Libraries You Want to Customize

If you are currently running ISMF, you can use the procedures described in this section to find the ISMF libraries you want to customize. If you are not running ISMF, and you need information about linking to the correct libraries, these books will help you:

- *System Generation*
- *DFDSS/ISMF: Installation Planning Guide*
- *Data Facility Hierarchical Storage Manager: Version 2 Release 4 Installation and Customization Guide*
- *DFSORT Installation and Customization*
- *ICKDSF System Control Programming Specifications*

Once you are linked to ISMF, the method you use to find the ISMF libraries depends on the library you want to modify.

Panel, Message, Skeleton, and Table Libraries: To find the right libraries for panels, messages, skeletons, and tables, use the TSO LISTALC STATUS command to determine the data set name associated with the DDNAME for the library. The DDNAMEs ISMF uses are listed in Figure 46 on page 107.

Figure 46. DDNAMEs for the Panel, Message, Skeleton, and Table Libraries

Library	DDNAME
Panel	ISPPLIB
Message	ISPMLIB
Skeleton	ISPSLIB
Table	ISPTLIB[1]
	ISPTABL[2]

Load and CLIST Library: The placement of the load library and the CLIST library is determined by the way ISMF is installed. The CLIST library DDNAME is SYSPROC or ISPCLIB. The load library may be given a DDNAME ISPLLIB or STEPLIB, or it may be made a part of the link pack area or the system link library. Figure 47 lists the DDNAME for the CLIST library and location or DDNAME for the load library.

Figure 47. DDNAMEs or Locations for the Load and CLIST Libraries

Library	Location/DDNAME
Load	ISPLLIB *or* STEPLIB *or* System link library *or* Link pack area
CLIST	SYSPROC *or* ISPCLIB

Making Changes and Testing Them

The best way to make and test changes in any of the ISMF libraries is to copy the member you want to modify from the ISMF library into a personal library. Add your library to the beginning of the existing concatenation that you or your installation uses. This ensures that you can safely make changes without impacting the other libraries in the concatenation. Once you've tested the changes, you can then change the concatenation to make the modified part available to a larger group of people, your department for example. If you want the change to be used by the entire installation, you can copy the member from your personal library back into the ISMF library. For members of the panel, message, skeleton, table, and CLIST libraries you can note the changes in the comment section at the beginning of the modified member. Remember to keep an unmodified copy for service and maintenance.

[1] Input table library.

[2] Output table library

Note: The load library is an exception. The methods you can use to modify the static text and ISMF tables in the load library are discussed in "Customizing the ISMF Command and Line Operator Tables" on page 124.

Customizing Panels

This chapter describes how to customize panels. It explains the changes you can make in the panel library. There are several restrictions to keep in mind both as you plan the way you want to customize panels, and as you use the procedures described in this chapter. They are listed at the beginning of each section.

Modifying Panel Definition

Restrictions

1. If you decide to change the initial priming values or default values on data entry or data selection panels, the new values must be set to run through the same verification code as the values supplied by ISMF. Otherwise, you may pass a value that is invalid.
2. If you remove a field from a panel by removing it from the)BODY section of the panel, you still need to supply an acceptable value for it in the)PROC section.
3. You can add new fields to existing panels, or create new panels, but ISMF won't have reference to them.
4. You can't move input fields from one panel to another.
5. You can change the format of most ISMF panels. However, if you decide to modify the FILTER Entry Panel or the SORT Entry Panel, you should look carefully at the validity checking in the)PROC section. The checking on these panels is done from left to right; changing the order of the input fields on these panels might impact the processing of values entered on the panels.
6. ISPF can display screens with a maximum of 24 lines. So, even if you use terminals that can display larger panels, you should be careful not to increase the number of lines in the)BODY section beyond 24. If the)BODY section is larger than 24 lines, the panel display will fail.
7. ISMF entry panels for data set and volume applications are designed to display default values if the user blanks out any of the fields on the panel. ISMF entry panels for all other applications display blanks if user blanks out any of the fields on the panel. In either case, because of the cursor positioning, you should preserve the order of the statements in the)PROC section that control the default redisplay. The)PROC section of each entry panel contains comments that identify the statements that should be kept in order.

Finding the Panel You Want to Change

Most of the changes you can make to ISMF panels are done in the panel library. The member name for an individual panel in the library is the panel ID that appears in the upper left hand corner of the panel when you use the ISPF PANELID command (see Figure 48).

```
DGTDDDS1                DATA SET SELECTION ENTRY PANEL
COMMAND ===>
```

Figure 48. Displaying the Panel ID

Testing the Changes

There are three ways to test the customizing you do on panels:

1. Invoke ISPF in test mode

   This will cause ISPF to refetch the panel when you display it after you've made changes.

2. Test your changes using the ISPF Dialog Test option

   This will cause ISPF to refetch the panel when you display it after you've made changes.

3. Make your changes and then exit and reinvoke ISPF

   When you invoke ISMF the modified panel will be displayed.

Changing Initial Priming Values on Data Entry Panels

The initial priming values for data entry panels are controlled by the)INIT section of each panel, with the exception of the profile entry panels. When you invoke a panel, ISPF executes the)INIT section before displaying the panel. The statements in the)INIT section look at the value stored in the application profile pool (APP) for each field on the panel. If the value in the APP is blank, ISPF substitutes the value from the)INIT section of the panel. Because the initial priming values for the profile entry panels are already stored in the APP, they cannot be changed.

To change the priming values for a particular panel, you change the statements in the)INIT section of that panel. For example, Figure 49 on page 110 is the Delete Entry Panel as it is initially displayed.

```
DGTDDDL2
                                DELETE ENTRY PANEL
COMMAND ===>

OPTIONALLY SPECIFY ONE OR MORE TO UNCATALOG
 DATA SET:  USER2.TEMP.TEMP

  SCRATCH DATA SET              ===> Y        (Y or N)

  CLEAR DATA SET WITH ZEROES    ===> Y        (Y or N)

  DELETE EVEN IF UNEXPIRED      ===> N        (Y or N)

  DATA SET PASSWORD             ===>          (if password protected)









USE ENTER TO PERFORM DELETE;
USE HELP COMMAND FOR HELP; USE END COMMAND TO EXIT.
```

Figure 49. Entry Panel for Delete

Figure 50 shows the priming values from the)INIT section of the panel. For example,

```
IF (&FDDLSCDS = '')        &FDDLSCDS = 'Y'
```

states that if the value for SCRATCH DATA SET is blank in the APP, ISMF will substitute a Y when the)INIT section is executed before the panel is displayed. If you want that field to be primed with an N, you can change the statement to read

```
IF (&FDDLSCDS = '')        &FDDLSCDS = 'N'
```

```
)INIT
  &ZHINDEX = DGTHIX00
  .HELP = DGTHDL02
  &DGTMHELP = DGTHDL02

  .ZVARS = '(FDDLSCDS FDDLCDWZ FDDLDEIU)'

  IF (&FDDLSCDS = '' )      &FDDLSCDS = 'Y'
  IF (&FDDLCDWZ = '' )      &FDDLCDWZ = 'Y'
  IF (&FDDLDEIU = '' )      &FDDLDEIU = 'N'
  &FDDLDSPW = ''

  .CURSOR = &FDDLFLDP
  .CSRPOS = &FDDLCPOS
```

Figure 50. Values in the INIT Section of the Delete Entry Panel

Changing Default Values for Data Entry Panels

When you blank out fields on a data set or volume application data entry panel, ISMF will supply the defaults. The defaults come from the statements in the)PROC section of each entry panel. Figure 51 shows the default values in the)PROC section of the Delete Entry Panel.

```
)REINIT
 REFRESH(*)
)PROC

/*************************************************************************/
/*                                                                       */
/*  Default values for variables left blank                              */
/*                                                                       */
/*************************************************************************/

  &DDDL2RD   = 'N'
  IF (&FDDLCDWZ = '' )      &FDDLCDWZ = 'Y'
     &DDDL2RD = 'Y'
  IF (&FDDLDEIU = '' )      &FDDLDEIU = 'N'
     &DDDL2RD = 'Y'
/*  The following two statements MUST remain together to ensure      */
/*  correct cursor positioning on the re-display of the panel.       */
  IF (&FDDLSCDS = '' )      &FDDLSCDS = 'Y'
     &DDDL2RD = 'Y'
  IF (&DDDL2RD = 'Y')
      .MSG = DGTUV091
```

Figure 51. ISMF Default Values for the Delete Entry Panel

If you want to change the value ISMF displays when you blank out a specific field, you can change the statement that corresponds to that field in the)PROC section of the panel. To ensure that the cursor is positioned in the correct place when the panel is redisplayed, be sure to preserve the order of the statements that are identified by the comments in the)PROC section.

Restricting Values for Specific Input Fields

The)PROC section also checks each value entered on a panel to make sure that it is valid. Figure 52 is the first page of the Data Set Selection Entry Panel. Figure 53 on page 113 shows the validity checking that ISMF does for the values entered on this panel.

```
DGTDDDS1
                          DATA SET SELECTION ENTRY PANEL                Page 1 of 3
COMMAND ===>

TO GENERATE A DATA SET LIST, SPECIFY:

  DATA SET NAME   ===> **
                                                  (fully or partially qualified)

  SELECT SOURCE OF GENERATED LIST  ===> 2         (1 or 2)

    1  GENERATE LIST FROM VTOC
         VOLUME SERIAL NUMBER  ===>               (fully or partially specified)

    2  GENERATE LIST FROM CATALOG
         CATALOG NAME  ===>
           CATALOG PASSWORD    ===>               (if password protected)
         VOLUME SERIAL NUMBER  ===>               (fully or partially specified)
         ACQUIRE DATA FROM VOLUME        ===> Y   (Y or N)
         ACQUIRE DATA IF DFHSM MIGRATED  ===> N   (Y or N)


USE ENTER TO PERFORM SELECTION; USE DOWN COMMAND TO VIEW NEXT SELECTION PANEL;
USE HELP COMMAND FOR HELP; USE END COMMAND TO EXIT.
```

Figure 52. Page 1 of the Data Set Selection Entry Panel

```
/*********************************************************************/
/*                                                                   */
/*  Check input variables for illegal values.                        */
/*                                                                   */
/*  If SELECT SOURCE OF GENERATED LIST is 1 then VOLUME SERIAL NUM-  */
/*    BER must be specified. Note that VOLUME SERIAL cannot be *.    */
/*                                                                   */
/*  If SELECT SOURCE OF GENERATED LIST is 2 then the following things*/
/*    must be checked:                                               */
/*                                                                   */
/*    1. If DATA SET NAME IS '*' or '**' then the CATALOG NAME must  */
/*       be specified.                                               */
/*    2. ACQUIRE DATA FROM VOLUME must be specified. (Y or N)        */
/*    3. ACQUIRE DATA IF DFHSM MIGRATED must be specified. (Y or N)  */
/*    4. Note that CATALOG NAME must be a valid dsn but it cannot    */
/*       be a member of a pds.                                       */
/*                                                                   */
/*********************************************************************/

  VER (&FDDSDSNM NB)
  VER (&FDDSSSGL NB LIST 1 2)

  IF (&FDDSSSGL = '1')
    VER (FDDDSVSN1 NB)
    IF (&FDDSVSN1 = '*')
      VER (&FDDSVSN1 LIST '' MSG=DGTUV019)

  IF (&FDDSSSGL = '2')

    &DSNCK1 = TRUNC(&FDDSDSNM,'.'
    IF (&DSNCK1 = '''*',''**',''*'',''**'')
      VER (&FDDSCTLN NB)

    IF (&ZPREFIX = '')
      IF (&DSNCK1 = '*','**')
        VER (&FDDSCTLN NB)

    VER (&FDDSADFV NB LIST Y N MSG=DGTUV005)
    VER (&FDDSADHM NB LIST Y N MSG=DGTUV005)
)END
```

Figure 53. Validity Checking on the Data Set Selection Entry Panel

If you want to further restrict valid values for any of the fields on the panel, you can add your own statements to the part of the)PROC section that does validity checking. For example, to prevent users from accessing the system residence volume, you could add a statement that makes '******' an invalid entry for the VOLUME SERIAL NUMBER field. The format of the statement would be

```
IF (&FDDSVSN1 = '******')
  VER (&FDDSVSN1 LIST '' MSG=XXXXXXXX)
```

The message ID, XXXXXXXX, is a message you have added explaining the restriction. In this case the user will not be able to generate a data set list until the value in the VOLUME SERIAL NUMBER field is valid. For more information on creating messages, see "Customizing Messages" on page 121, and *ISPF Dialog Management Services.*

Removing Fields

You can remove a field from a panel by deleting it from the)BODY section of the coding for the panel. However, you should keep in mind that there may be more work involved than simply deleting the field. When you plan to remove a field you should look carefully at the)INIT and)PROC sections of the panel to see how that field is referenced. To accommodate changes you make to the body of the panel, you may need to modify the statements for defaulting in the)INIT and)PROC sections, and the verification code in the)PROC section. For example, to remove the CATALOG NAME field from the Data Set Selection Entry Panel, you would look at the code from the panel that applies to CATALOG NAME:

1. The initial default value supplied by the)INIT section
2. The default supplied by the)PROC section if the user enters a blank
3. The verification code that corresponds to the field

Since ISMF does not ship a default for CATALOG NAME in the APP, and both of the default statements supply a blank,

```
IF (&FDDSCTLN = '')      &FDDSCTLN = ''
```

you do not need to modify either of the default statements to remove the field.

However, you do need to change the verification code. The code that applies to the CATALOG NAME field is

```
IF (&FDDSSSGL = '2')

  &DSNCK1 = TRUNC(&FDDSDSNM,'.'
  IF (&DSNCK1 = '''*',''**',''*''',''**''')
    VER (&FDDSCTLN NB)

    IF (&ZPREFIX = '')
      IF (&DSNCK1 = '*','**')
        VER (&FDDSCTLN NB)
```

If option 2 is specified for SELECT SOURCE OF GENERATED LIST (the variable &FDDSSSGL) and the data set name (the variable &DSNCK1) is either quoted with an asterisk as the high level qualifier ('*.LOAD'), or a quoted double asterisk ('**'), the code checks to ensure that a catalog name has been supplied. Thus to remove the CATALOG NAME field from the panel you need to change the verification code. The new code should refer to a message that explains that for a list that is generated from the catalog, '*' and '**' are not valid ways of specifying the data set name:

```
IF (&DSNCK1 = '''*',''**',''*''',''**''')
  .MSG = XXXXXXXX
```

Highlighting and Color

The highlighting and color on ISMF panels are controlled by the statements in the)ATTR section of the panel.

For highlighting, the attribute characters are set explicitly by ISMF. For example,

```
¬ TYPE(INPUT) INTENS(NON)
$ TYPE(INPUT) INTENS(HIGH) JUST(RIGHT)
+ TYPE(TEXT) INTENS(LOW) SKIP(ON)
% TYPE(TEXT) INTENS(HIGH) SKIP(ON)
```

Color is based on ISPF defaults for the protection and intensity attributes specified with the TYPE and INTENS keywords. Color is also dependent on the hardware capabilities of the terminals you use. Figure 54 shows the ISPF defaults.

Figure 54. Default Colors

Color	Field Type	Intensity
Green	input	low
Blue	text/output	low
Red	input	high
White	text/output	high

If you want to change the color you can add the COLOR keyword to the statements in the)ATTR section and remove the INTENS keyword. For example, the statement

```
$ TYPE(INPUT) COLOR(PINK)
```

sets pink as the color for the characters entered in fields with the $ attribute. If you code both the INTENS keyword and the COLOR keyword, the COLOR keyword is ignored. For more information on specifying color and highlighting, and how the two are related, see *ISPF Dialog Management Services*.

Changing the Format

You can change the format of a panel by changing the position of the fields. If you do there are several things to keep in mind:

Field length

Each field has its own length. When you move a field you need to make sure that you don't change the length. This will ensure that none of the fields on the panel is truncated.

Attribute characters

Each field starts with an attribute character and ends with another attribute character, or the end of the line. When you move a field you need to identify the attribute characters and decide whether to modify them to accommodate the change.

Autoskip

The panels are coded to use autoskip to move from one input field to the next. If you move a field, you may need to adjust the attribute characters that control autoskip.

Input fields

Many of the input fields are grouped together because they supply related information, or because they are dependent on each other. If you move a field, you may need to move some of the fields around it to preserve the structure and logic of the panel.

Validity Checking

The logic of the validity checking in the)PROC section generally matches the order of the fields on the panel; the checking is done from top to bottom. When you move a field, you should make sure the validity checking parallels the new order.

Double lines for input fields

Whenever you move input fields around on a panel, you need to move all the lines associated with that field. For example, for data set application, both the FILTER Entry Panel and the Data Set Selection Entry Panel have fields that allow input on two lines (DATA SET ORGANIZATION, DEVICE TYPE, and RECORD FORMAT). If you move these fields around, you need to move both lines.

Number of lines in the)BODY section

ISPF can display screens with a maximum of 24 lines. So, even if you use terminals that can display larger panels, you should be careful not to increase the number of lines in the)BODY section beyond 24. If the)BODY section is larger than 24 lines, panel display will fail.

Modifying Text

You can modify text on any of the functional panels or help panels by editing the)BODY section. Remember that the attribute character to the left and right of the text you are working with controls the field length, spacing, indentation, and centering.

Adding Fields

When you add a field, you need to look at the)ATTR section of the panel and pick an attribute character to make the new field consistent with the rest of the panel. For example, you could use the ISPF ZTIME and ZDATE system variables to display the current time and date on the Data Set List panel. Figure 55 on page 117 shows the)ATTR section and the original coding for the top of list panel. Figure 56 on page 117 shows the coding for the added fields. The next time we invoke the list panel, it will display the current date and time. Figure 57 on page 117 is the customized list panel as it is displayed.

```
  * AREA(DYNAMIC) EXTEND(OFF) SCROLL(OFF)
  ¬ TYPE(INPUT) INTENS(NON)
  $ TYPE(INPUT) INTENS(HIGH) JUST(RIGHT)
  ¢ TYPE(OUTPUT) INTENS(LOW) SKIP(ON) JUST(ASIS)
  - TYPE(OUTPUT) INTENS(HIGH) SKIP(ON) JUST(ASIS) CAPS(OFF)
  + TYPE(TEXT) INTENS(HIGH) SKIP(ON)
  # TYPE(TEXT) INTENS(LOW) SKIP(ON)
)BODY
+                                DATA SET LIST
+COMMAND ===>_ZCMD                                          +SCROLL ===>_ZAMT+
+                                                  #&FDDSENTR
#ENTER LINE OPERATORS BELOW:                       &FDDSDCOL
```

Figure 55. Original Version of the List Panel

```
  * AREA(DYNAMIC) EXTEND(OFF) SCROLL(OFF)
  ¬ TYPE(INPUT) INTENS(NON)
  $ TYPE(INPUT) INTENS(HIGH) JUST(RIGHT)
  ¢ TYPE(OUTPUT) INTENS(LOW) SKIP(ON) JUST(ASIS)
  - TYPE(OUTPUT) INTENS(HIGH) SKIP(ON) JUST(ASIS) CAPS(OFF)
  + TYPE(TEXT) INTENS(HIGH) SKIP(ON)
  # TYPE(TEXT) INTENS(LOW) SKIP(ON)
)BODY
+                                DATA SET LIST
+COMMAND ===>_ZCMD                                          +SCROLL ===>_ZAMT+
#DATE:¢ZDATE                                      #&FDDSENTR
#TIME:¢ZTIME                                      #&FDDSDCOL
+
```

Figure 56. Adding Date and Time to the List Panel

```
                              DATA SET LIST
COMMAND ===>                                              SCROLL ===> HALF
Date: 85/11/02                                      Entries 1-9 of 9
Time: 12:08                                         Data Columns 3-7 of 21
```

Figure 57. List Panel Customized to Show Date and Time

Creating Panels

You can use the panel definition procedures described in *ISPF Dialog Manager Services* to add your own panels to those provided by ISMF. When you add panels you should consider:

Variables

Make sure that the variable names you assign do not conflict with existing names, unless the function that uses the new panel runs from a different ISPF function pool.

Consistency

For ease-of-use and to prevent errors, make your new panels consistent with ISMF style. Use the same format and operational characteristics. For example, input fields on ISMF panels are denoted by a white or intensified arrow to the left of the field. To avoid confusion, input fields on panels you add should look the same. Or, for example, if you add a functional panel, the ENTER key should execute the function.

Modifying Fields on the List Panel

You may now modify the following fields on ISMF's List Panel:

- Column Headings.
- "Entries" line in the fixed area located in the upper right corner of the ISMF List Panel. See Figure 58.
- "Data Columns" line in the fixed area.
- "BOTTOM OF DATA" line located at the end of the list.

```
DGTLDDS1                        DATA SET LIST
COMMAND ===>                                                SCROLL ===> PAGE
                                                          Entries 1-3 of 3
ENTER LINE OPERATORS BELOW:                               Data Columns 8-11 of 26

    LINE                                        SEC     DS    REC   RECORD
    OPERATOR          DATA SET NAME             ALLOC   ORG   FMT   LENGTH
   ---(1)----  ------------(2)------------  --(8)--  (9)  (10)-  -(11)-
               USER2.CLIST.CLIST                 94   PO   VB        255
               USER2.ISMF.ISPPLIB                94   PO   FB         80
               USER2.ISPF.PROFILE                94   PO   FB         80
   ----------  ------  -----------  BOTTOM  OF  DATA  -----------  ------  ----
```

```
USE HELP COMMAND FOR HELP; USE END COMMAND TO EXIT.
```

Figure 58. ISMF Data Set List Panel

Where Do You Make the Changes?

You can modify the following members of the message library to change the column headings:

Application	Member Names
Data Set	DGTDS05, DGTDS06, DGTDS07, DGTDS08, DGTDS09, DGTDS10, DGTDS11, DGTDS12, DGTDS13
Volume	DGTVA05, DGTVA06, DGTVA07, DGTVA08, DGTVA09, DGTVA10, DGTVA11, DGTVA12, DGTVA13, DGTVA14, DGTVA15, DGTVA16, DGTVA17
Data Class	DGTBL00, DGTBL01, DGTBL02, DGTBL03, DGTBL04, DGTBL05, DGTBL06, DGTBL07
Storage Class	DGTSC01, DGTSC02, DGTSC03, DGTSC04
Management Class	DGTML00, DGTML01, DGTML02, DGTML03, DGTML04, DGTML05, DGTML06
Storage Group	DGTGL00, DGTGL01, DGTGL02, DGTGL03, DGTGL04, DGTGL05, DGTGL06, DGTGL07

The statements in the fixed area can be modified from the member, DGTFO00, of the message library.

You can modify one or more of the following members of the panel library to change the "BOTTOM OF DATA" line:

Application	Member Name
Data Set	DGTLDDS1
Volume	DGTLVVA1
Data Class	DGTLCDC1
Storage Class	DGTLCSC1
Management Class	DGTLCMC1
Storage Group	DGTLCSG1

Note: When ISMF is installed, the message library name is SYS1.DGTMLIB, and the panel library is called SYS1.DGTPLIB.

Special Considerations

- You should make a copy of the library you modify because the next time a link-edit or maintenance is performed on the member you have changed, your modification will be lost.
- When editing ISMF libraries, do not change the NUM field in the profile and do not issue the RENUM editing command.
- You can change the wording, but you can't change the order of the columns or the characters to the left or right of the headings. Also, you can't add or delete columns.
- The widths of the first two columns are fixed, so any textual changes you make will not alter the size of the fields.

- You can modify the lengths as well as the text of the third through the last column headings. Be sure to update the lengths associated with the text you lengthen.

```
/*********************************************************************
/*  Column 10: REC FMT
/*  Length   : 5
/*********************************************************************
DGTDS080   '5'
' REC '

DGTDS081
' FMT '

DGTDS082
'(10)-'
```

Figure 59. Column 10 of Member DGTDS08.

If, for example, you would like to change column 10 to say RECORD FORMAT instead of REC FMT, simply:

1. Access member DGTDS08 of the MESSAGE library (see Figure 59).
2. Enter PROFILE on the command line and verify that NUMBER is set OFF.
3. Replace ' REC ' with 'RECORD' and replace ' FMT ' with 'FORMAT'.
4. Replace the decimal length '5' with '6' (the new length).
5. Pad the tag with dashes. In other words, '(10)-' would become '-(10)-'.
6. Compare your results to Figure 60.

Note: Although you should not shorten the column headings, you can expand the headings for the third through the last columns up to 35 characters.

- If you change the headings on the List Entry Panel, you should also change the corresponding fields and text on the Selection Entry Panel, and the SORT Entry Panel. For data set and volume applications, you should also change the corresponding fields and text on the FILTER Entry Panel. You should also change the help panels and messages that support these entry panels and the list panel.

```
/*********************************************************************
/*  Column 10: RECORD FORMAT
/*  Length   : 6
/*********************************************************************
DGTDS080   '6'
'RECORD'

DGTDS081
'FORMAT'

DGTDS082
'-(10)-'
```

Figure 60. Column 10 of Member DGTDS08 After Customization.

Customizing Messages

This chapter explains how to modify ISMF messages and how to add your own messages. It is divided into two sections, "Modifying ISMF Messages," and "Creating New Messages" on page 122.

Modifying ISMF Messages

Restrictions

1. Do not change the names of any of the variables contained in ISMF messages.
2. Do not change the message number.
3. Short messages cannot exceed 24 characters.
4. Long messages cannot exceed 78 characters.
5. Message text can be entered in upper and lower case, but the other fields in the message—the message number, variables, keywords, and the help panel ID—must be in uppercase.
6. When you change the text of a message you should change the corresponding message help panel.

Finding the Message You Want to Change

To find the message you want to change you need to know the message number. The message number is listed at the top of each message help panel (see Figure 61).

```
HELP-----------------------------ISMF MESSAGE-----------------------------HELP
COMMAND ===>

   MESSAGE NUMBER:  DGTMD006

   SHORT MESSAGE:   DFHSM LEVEL UNKNOWN

   LONG MESSAGE:    DFHSM LINE OPERATORS MAY FAIL - DFHSM V2 R2.1 OR LATER
                    NEEDED
```

Figure 61. Identifying the Message Number

Related ISMF short and long messages are stored together in members of the message library. To determine where the message you want to change is stored, truncate the last digit of the message number. This will give you the member name. Thus, the message DGTMD006 is stored in DGTMD00 with other messages that begin with DGTMD00.

Making the Change

Once you have identified the member the message is stored in, you are ready to make the change. Modify the message and save your changes. Then modify the message help panel that is pointed to by .HELP. For example, to change message DGTMD006, you would edit the message itself in member DGTMD00 and the related text in the message help panel DGTMMD06. Figure 62 on

page 122 shows the entry in the message library for DGTMD006. The .HELP field is highlighted.

```
DGTMD006   'DFHSM LEVEL UNKNOWN'      .HELP= DGTMMD06   .ALARM= YES
'DFHSM LINE OPERATORS MAY FAIL - DFHSM V2 R2.1 OR LATER NEEDED

DGTMD007   'DFDSS LEVEL UNKNOWN'      .HELP= DGTMMD07   .ALARM= YES
'DFDSS LIST COMMANDS AND LINE OPERATORS MAY FAIL - DFDSS V2 R2 OR LATER NEEDED

DGTMD008   'ISMF FAILED'              .HELP= DGTMMD08   .ALARM= YES
'UNABLE TO INITIALIZE ISMF CONTROL BLOCKS

DGTMD009   'ISMF FAILED'              .HELP= DGTMMD09   .ALARM= YES
'UNABLE TO DISPLAY ISMF PRIMARY OPTION MENU
```

Figure 62. Changing the Short and Long Messages

Creating New Messages

You can use the procedures for message definition described in *ISPF Dialog Management Services* to add your messages to those provided by ISMF. When you add messages you should consider:

Message numbers

Make sure that the message numbers you assign do not duplicate existing ones.

Consistency

ISMF uses short and long messages, and message help panels to identify errors. If you add short messages, you should add the supporting long messages and message help panels. The style of the message help panels should be consistent with ISMF panels.

Customizing Job Skeletons

This chapter explains how to tailor the job skeletons that ISMF uses to generate the job streams used by DFDSS, ICKDSF, IEBPTPCH, and IDCAMS.

Restrictions

1. You can remove variables from the skeletons, but you should make sure that a variable you remove from one part of a skeleton isn't needed by some other part.
2. Do not change any of the variable names in the skeletons. ISMF code is dependent on these names.
3. If you add variables, make sure that the names you use do not duplicate existing ones.

Finding the Skeleton You Want to Change

The ISMF skeletons for MVS/DFP, DFSORT, and DFDSS line operators and list commands are kept in their respective skeleton libraries. MVS/DFP and DFDSS members begin with DGTK. The remaining characters in the name identify the line operator or command. Thus the member DGTKCY01 contains the job skeleton for the COPY line operator. DFSORT members begin with ICEK. The member ICEKSRRC contains the job skeleton for the SORTREC line operator.

Note: When ISMF is installed, the skeleton library name is SYS1.DGTSLIB.

Making the Changes

There are several ways to customize the ISMF skeletons for MVS/DFP and DFDSS jobs:

- You can add statements to imbed skeletons of your own.
- You can modify the variables in the skeletons to override the input that the skeletons get from the values entered on the data entry and job submission panels.
- You can add pre- and postprocessing steps to the job stream.

For example, if you want to imbed your own skeleton in the ISMF skeleton, you begin by creating the skeleton you want to imbed. The new skeleton might contain statements that add new steps. Then you imbed the name of this skeleton in the original skeleton. The job stream that is generated from the tailored skeleton contains the new steps.

Customizing Tables

This section describes how to customize command tables. It is divided into two sections, "Customizing the ISPF Command Tables" and "Customizing the ISMF Command and Line Operator Tables" on page 124. The first explains the additions you can make to the ISPF command tables in the table library. The second explains the changes and additions you can make to the ISMF command and line operator tables in the load library. Restrictions to customizing the tables are listed at the beginning of each section.

Customizing the ISPF Command Tables

Restrictions

1. Do not delete any of the entries in the command tables
2. Do not delete any of the tables

Finding the Table You Want to Change

The ISPF command tables are kept in the table library. The tables you can change have a name that ends in CMDS.

Making the Changes

You can make changes to the table library using the ISPF command table utility (option 3.9). Figure 63 is an example of a table displayed using option 3.9.

```
     VERB      T  ACTION
                    DESCRIPTION
'''' CLEAR     0  PASSTHRU
'''' COMPRESS  0  PASSTHRU
'''' COPY      0  PASSTHRU
'''' DOWN      0  PASSTHRU
'''' DUMP      0  PASSTHRU
'''' FILTER    0  PASSTHRU
'''' FIND      0  PASSTHRU
'''' LEFT      0  PASSTHRU
'''' PROFILE   0  PASSTHRU
'''' RELEASE   0  PASSTHRU
'''' RESHOW    0  PASSTHRU
```

Figure 63. Using Command Table Utility to Update ISPF Tables

The command table utility reads the table from ISPTLIB and writes it out to ISPTABL. If you use the utility to update a command table, you should make sure that both libraries use the same data set for the table you want to change. When you add a command to the ISPF command tables, you should also add it to the ISMF tables. The method you use to do this is described in "Customizing the ISMF Command and Line Operator Tables."

Controlling Truncation

Truncation is determined by the ZCTTRUNC and the ZTACT fields in the command table. All ISMF commands in the ISPF table are set with a truncation of 0 and an action of PASSTHRU. This passes the entire command to the ISMF dialog for resolution. When you add a command, you should coordinate the truncation value you specify with the values specified for the existing commands in the ISPF tables, the system tables, and the tables for ISMF commands. For more information on the structure of ISPF command tables, and how to alter them, see *ISPF Dialog Management Services*.

Customizing the ISMF Command and Line Operator Tables

Restrictions

1. Do not change the name of the command or line operator. You can, however, change the name of the routine that gets control.
2. You can replace one of the empty command or line operator tables that ISMF ships with a table of your own, but your table should use the same format as the ISMF tables. The control block, DGTMCTAP, in "ISMF

Command Table Format" on page 125, contains the format for the command tables. The control block, DGTMLPAP, in "ISMF Line Operator Table Format" on page 126, contains the format for the line operator tables. If new commands are added to the tables, ISMF will recognize them.

ISMF Command Table Format

DGTMCTAP

```
CTAP
Offsets      Type      Length  Name          Description
================================================================================
      COMMAND TABLE - APPLICATION TABLE (CTAP)
================================================================================
    0    (0) CHARACTER     *  CTAP
    0    (0) CHARACTER     8  CTAPMAIN
    0    (0) CHARACTER     4  CTAPVID        VISUAL ID: 'CTAP'
    4    (4) FIXED         2  CTAPLEN        LENGTH OF CTAP
    6    (6) FIXED         2  CTAPCNT        # OF COMMAND ENTRIES
    8    (8) CHARACTER    28  CTAPENT(*)
    8    (8) CHARACTER     8  CTAPNAME       COMMAND NAME
   16   (10) FIXED         1  CTAPTRUN       MIN. # OF CHARACTERS USED IN
                                              TRUNCATION
   17   (11) BITSTRING     1  CTAPFLAG       FLAG FIELD
         1... ....           CTAPST         COMMAND STATUS
         .1.. ....           CTAPIMED       IMMEDIATE COMMAND
         ..1. ....           CTAPLIST       LIST COMMAND
         ...1 ....           CTAPACMD       ALTERNATE COMMAND
         .... 1111           CTAPRSVD       RESERVED
   18   (12) CHARACTER     8  CTAPRTNM       COMMAND ROUTINE NAME
   26   (1A) CHARACTER     8  CTAPTENM       CMD TERMINATION ROUTINE
   34   (22) BITSTRING     2  *              FILL UP END OF WORD
Constants
Length   Type      Value      Name          Description
================================================================================
      END OF COMMAND TABLE - APPLICATION TABLE (CTAP) DEFINE
      COMMAND STATUS BITS
================================================================================
        BIT       1          CMDENABL      COMMAND STATUS IS ENABLE
        BIT       1          CMDDSABL      COMMAND STATUS IS DISABL
```

ISMF Line Operator Table Format

DGTMLPAP

```
LPAP
Offsets      Type      Length  Name          Description
=============================================================================
     LINE OPERATOR TABLE - APPLICATION TABLE (LPAP)
=============================================================================
   0    (0) CHARACTER     *  LPAP
   0    (0) CHARACTER     8  LPAPMAIN
   0    (0) CHARACTER     4  LPAPVID        VISUAL ID: 'LPAP'
   4    (4) FIXED         2  LPAPLEN        LENGTH OF LPAP
   6    (6) FIXED         2  LPAPCNT        # OF LINE OPERATORS
   8    (8) CHARACTER    28  LPAPENT(*)
   8    (8) CHARACTER     8  LPAPLONM       LINE OPERATOR NAME
  16   (10) FIXED         1  LPAPTRUN       MIN. # OF CHARACTERS USED IN
                                             TRUNCATION
  17   (11) CHARACTER     3  *              RESERVED, UNUSED
  20   (14) CHARACTER     8  LPAPRTNM       LINE OP ROUTINE NAME
  28   (1C) CHARACTER     8  LPAPTENM       TERMINATION ROUTINE
```

Finding the Tables

The ISMF tables for line operators and commands are kept in the load library. They are grouped by function. Figure 64 lists the application member names for line operators. Figure 65 on page 128 lists the application member names for commands.

Figure 64 (Page 1 of 2). Member Names for Line Operator Tables

Function:	MVS/DFP ISMF	DFHSM ISMF	DFDSS ISMF	DFSORT ISMF	blank
Data Set Member	DGTTLPD1	DGTTLPD2	DGTTLPD3	DGTTLPD4	DGTTLPD5 DGTTLPD6 DGTTLPD7 DGTTLPD8
Volume Member	DGTTLPV1	none	DGTTLPV2	none	DGTTLPV3 DGTTLPV4 DGTTLPV5 DGTTLPV6 DGTTLPV7 DGTTLPV8
Data Class Member	DGTTLPB1 DGTTLPB2 DGTTLPB3 DGTTLPB4 DGTTLPB5 DGTTLPB6 DGTTLPB7 DGTTLPB8	none	none	none	none

Figure 64 (Page 2 of 2). Member Names for Line Operator Tables

Function:	MVS/DFP ISMF	DFHSM ISMF	DFDSS ISMF	DFSORT ISMF	blank
Storage Class Member	DGTTLPS1 DGTTLPS2 DGTTLPS3 DGTTLPS4 DGTTLPS5 DGTTLPS6 DGTTLPS7 DGTTLPS8	none	none	none	none
Management Class Member	DGTTLPM1 DGTTLPM2 DGTTLPM3 DGTTLPM4 DGTTLPM5 DGTTLPM6 DGTTLPM7 DGTTLPM8	none	none	none	none
Storage Group Member	DGTTLPG1 DGTTLPG2 DGTTLPG3 DGTTLPG4 DGTTLPG5 DGTTLPG6 DGTTLPG7 DGTTLPG8	none	none	none	none

Figure 65. Member Names for Command Tables

Function:	MVS/DFP ISMF	DFDSS ISMF	blank
Data Set Member	DGTTCTD1	DGTTCTD2	DCTTCTD3 DGTTCTD4 DGTTCTD5 DGTTCTD6 DGTTCTD7 DGTTCTD8
Volume Member	DGTTCTV1	none	DGTTCTV2 DCTTCTV3 DGTTCTV4 DGTTCTV5 DGTTCTV6 DGTTCTV7 DGTTCTV8
Data Class Member	DGTTCTB1 DGTTCTB2 DGTTCTB3 DGTTCTB4 DGTTCTB5 DGTTCTB6 DGTTCTB7 DGTTCTB8	none	none
Storage Class Member	DGTTCTS1 DGTTCTS2 DGTTCTS3 DGTTCTS4 DGTTCTS5 DGTTCTS6 DGTTCTS7 DGTTCTS8	none	none
Management Class Member	DGTTCTM1 DGTTCTM2 DGTTCTM3 DGTTCTM4 DGTTCTM5 DGTTCTM6 DGTTCTM7 DGTTCTM8	none	none
Storage Group Member	DGTTCTG1 DGTTCTG2 DGTTCTG3 DGTTCTG4 DGTTCTG5 DGTTCTG6 DGTTCTG7 DGTTCTG8	none	none
ACS Member	DGTTCTA1 DGTTCTA2 DGTTCTA3 DGTTCTA4 DGTTCTA5 DGTTCTA6 DGTTCTA7 DGTTCTA8	none	none
CDS Member	DGTTCTC1 DGTTCTC2 DGTTCTC3 DGTTCTC4 DGTTCTC5 DGTTCTC6 DGTTCTC7 DGTTCTC8	none	none

Figure 66 lists the member names for the profile application command tables. The tables are used for all applications.

Figure 66. Member Names for Profile Application Command Tables

Function	Member Name
MVS/DFP ISMF commands	DGTTCTP1
DFDSS ISMF commands	DGTTCTP2
blank	DCTTCTP3
blank	DGTTCTP4
blank	DGTTCTP5
blank	DGTTCTP6
blank	DGTTCTP7
blank	DGTTCTP8

Making the Changes

There are two ways to change the ISMF tables for line operators and commands. You can add new entries to the existing tables or to one of the blank tables ISMF ships. If you add entries to the ISMF tables, you should also update the ISPF command table.

Whenever a new command is added to an application in ISMF, it must be added to the command table for all applications in ISMF, not only for the application affected.

Modifying the Existing Tables

Because the tables are stored in the load library, you cannot edit them directly.

If you want to make extensive changes:

1. Create your own table following the format that ISMF uses. See "ISMF Line Operator Table Format" on page 126 for the format of the line operator tables.
2. Enter the line operators along with the ISMF entries in the new table.
3. Link-edit the new table under the original member name. This will overlay the original table with your new table.

If you want to make minor changes: you can SUPERZAP the member that contains the table you want to change. However, the next time a link-edit or maintenance is performed on the member, the change will be lost. For information on how to use SUPERZAP, see the publication, *Service Aids*.

Using One of the Blank Tables

ISMF ships 29 blank tables in the load library: 19 overlay command tables and 10 overlay line operator tables. You can use the line operator tables to add your own entries. Figure 64 on page 126 and Figure 66 list the member names for the blank tables. To make entries in one of the blank tables:

1. Create a table following the format that ISMF uses.
2. Enter the new line operators in the table. For new commands, set the CTAPACMD bit to 1. Also, be sure to update the count value in CTAPCNT

to reflect the number of entries in the table. See "ISMF Command Table Format" on page 125.

3. Link-edit the table using the member name for the blank table that you want to overlay.

Customizing the ISMF CLIST

This chapter explains how to change the CONTROL statement on the ISMF CLISTs.

Restrictions

Do not alter the CLISTs themselves. Changes to the logic may create problems with job submission. For example, jobs may be submitted incorrectly, or not submitted at all. Logging of submission may fail, or it may be incorrect. Changing the CLISTs could also cause incorrect feedback for job submission. If you wish to modify the job streams, you can do so by tailoring the job skeletons. The method you use to do this is described in "Customizing Job Skeletons" on page 122. It is easier than changing the CLISTs, and less error prone.

Finding the CLIST

CLISTs are stored in the CLIST library. The member names include DGTQSU01 for the DFDSS CLIST, DGTQSF01 for the MVS/DFP CLIST, ICESRCFG for the DFSORT CLIST, and DGTQCB01 for SETCACHE.

Note: When ISMF is installed, the CLIST library name is SYS1.DGTCLIB.

Making the Changes

You can change the CONTROL statement that ISMF ships with the CLIST using any of the operands for CONTROL listed in *TSO/E V2 Command Reference*. Figure 67 shows the CONTROL statement in the DFDSS ISMF CLIST. It is located at the beginning of the data set, immediately after the comment section.

```
/*                                                                    *
/*                                                                    *
/*   PROCESSOR:  ISPF                                                 *
/*                                                                    *
/*   CHANGE ACTIVITY:  LEVEL 0                                        *
/*   $L0=ISMFREL,JAE2211,,PRGRMA:                                     *
/*                                                                    *
/**********************************************************************
CONTROL NOFLUSH PROMPT
/**********************************************************************
/*   BEGIN CLIST MAINLINE                                             *
/**********************************************************************
```

Figure 67. Control Statement in the ISMF CLIST

To change the CONTROL statement you need to edit the CLIST member in the CLIST library. For example, to change the CONTROL statement for DFDSS, you need to edit the DGTQSU01 member in the CLIST library. You could add the LIST operand as shown in Figure 68.

```
/*                                                                    *
/*                                                                    *
/*   PROCESSOR:  ISPF                                                 *
/*                                                                    *
/*   CHANGE ACTIVITY:  LEVEL 0                                        *
/*   $L0=ISMFREL,JAE2211,,PRGRMA:                                     *
/*                                                                    *
/**********************************************************************
CONTROL NOFLUSH PROMPT LIST
/**********************************************************************
/*   BEGIN CLIST MAINLINE                                             *
/**********************************************************************
```

Figure 68. Changing the Control Statement

MVS/DFP Installation Exits

About This Section

This section is intended to help you to customize MVS/DFP by describing how to write installation exit routines and replace modules. It contains product-sensitive programming interfaces provided by MVS/DFP. Installation exits and other product-sensitive interfaces are provided to allow your installation to perform tasks such as product tailoring, monitoring, modification, or diagnosis. They are dependent on the detailed design or implementation of the product. Such interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

This section contains:

Chapter 9. Data Management Installation Exit Routines

General Guidance

This chapter discusses how installation-written exit modules can:

- Take control before and after direct access device storage management (DADSM) processing.
- Take control during Open for a DCB.
- Determine whether a missing data set control block (such as for a data set that has been moved to another volume) can be restored to a volume.
- Recover from errors that may occur during the opening, closing, or handling of an end-of-volume condition for a data set associated with the user's task.
- Bypass, limit, or override system-calculated values that assist you in selecting optimum DASD data set block size/CI size.
- Bypass or change datestamp processing for VSAM.

The data management replaceable modules are listed in Figure 69.

Figure 69 (Page 1 of 2). Data Management Replaceable Modules

Module Name	Description	When Available
IDATMSTP	Datestamp processing in VSAM	During VSAM OPEN
IFG0EX0A	Open/EOV installation exit	Format-1 DSCB not found during OPEN or end-of-volume
IFG0EX0B	DCB open installation exit	At open
IFG0199I	Data management abend installation exit	open, close, end of volume abnormal conditions
IGBDCSX1 IGBDCSX2	precalculation and postcalculation exit	DASD calculation services
IGGDARU2	DADSM RENAME preprocessing exit module	After initialization for RENAME
IGGDARU3	DADSM RENAME postprocessing exit module	Before exit from RENAME
IGGDASU2	DADSM SCRATCH preprocessing module	After initialization for SCRATCH
IGGDASU3	DADSM SCRATCH postprocessing exit module	Before exit from SCRATCH

Figure 69 (Page 2 of 2). Data Management Replaceable Modules		
Module Name	**Description**	**When Available**
IGGPRE00 IGGPOST0	DADSM preprocessing and postprocessing exit	DADSM functions allocate, extend, scratch, partial release and rename.
IGG026DU	Catalog preinitialization exit module	Before or after CATALOG
IGG029DM	DADSM SCRATCH failure exit module	SCRATCH, after error return code of 4 or 8
IGG029DU	DADSM SCRATCH preinitialization exit module	Before or after SCRATCH
IGG030DU	DADSM RENAME preinitialization exit module	Before or after RENAME
IGXMSGEX	Message display exit module	Before end of message display processing

Programming Considerations

The data management replaceable modules you decide to replace must be named the same as the IBM-supplied modules.

In general, the data management replaceable module you replace must

- Handle multiple requests (reentrant)
- Reside in SYS1.LPALIB (or link-edit into LINKLIB)
- Save and restore registers

Limitations and Restrictions

Be aware of the impact other products have on the modifications you install. For example, RACF takes control at the same time as some of the installation exit modules. There may be contention for resources.

DADSM Preprocessing and Postprocessing Exit Routines

DADSM allows an installation-written preprocessing module (exit routine) to take control before DADSM processing, and an installation-written postprocessing module after DADSM processing. DADSM uses an exit parameter list to communicate with these exit routines. This parameter list is obtained from storage below the 16M line. The format of the parameter list is shown in Figure 70 on page 138.

The Exit Modules

All DADSM functions (allocate, extend, scratch, partial release, and rename) have a common preprocessing exit routine and a common postprocessing exit routine that the installation exit routine can replace. These exit routines enable you to gain control before and after DADSM processing. The preprocessing exit routine module is IGGPRE00; the postprocessing exit routine module is IGGPOST0. Each is used by all the DADSM functions just mentioned. The modules reside in SYS1.LPALIB. You can use System Modification Program

(SMP) to replace the IBM-supplied exit routine modules with an installation exit routine you write.

The Exit Environment

The exit routines are given control in supervisor state and protect key zero with no locks held. The exit routines may execute in either 24-bit or 31-bit addressing mode. If they execute in 24-bit mode, it is important to consider the following:

> The DADSM caller, usually VSAM, O/C/EOV, or the scheduler, passes a JFCB pointer. If this caller's work area resides above the 16M line, the IEXPTR1 field, which contains the JFCB address for allocate, extend, and partial release, is a 31-bit address. When your exit routine is called for allocate, extend, or partial release and the JFCB resides above the 16M line, it must be in 31-bit addressing mode before using the IEXPTR1 field in the exit parameter list.

The exit routines must be reentrant. DADSM or the program that invokes DADSM (by issuing enqueue, reserve, and so forth) has acquired the system resources needed to serialize system functions. These enqueues may prevent other system services from completing successfully. In particular, exit routines must not issue dynamic allocation, OPEN, CLOSE, EOV, LOCATE, and other DADSM functions because they issue an enqueue on the SYSZTIOT resource. If the exit routines require access to an installation data set, the control blocks required to access that data set (DCB, DEB) should be built during system initialization (IPL/NIP).

The type and number of resources held by DADSM depend upon the DADSM function and the exit taken. For example, on entry to the installation preprocessing exit (IGGPRE00), DADSM holds an enqueue on the VTOC and a reserve on the device for the subject volume of a SCRATCH, RENAME, or partial release function. DADSM releases these resources before the installation postprocessing exit (IGGPOST0) takes control.

You must anticipate system resource contention when services are requested from an exit routine. For example, RACF services issue an enqueue on the RACF data set or a reserve on that data set's volume. This contention can cause system performance problems or an interlock condition.

When IGGPRE00 Gets Control

The preprocessing exit routine, IGGPRE00, is given control before the first VTOC update and after the initial validity check is successful. Input to IGGPRE00 is a parameter list, mapped by macro IECIEXPL, that contains addresses of input data and a function code that identifies the DADSM function. IGGPRE00 is given control once for each volume in the volume list supplied to scratch and rename. A field in the parameter list, IEXRSVWD, may be used to pass data from the preprocessing exit routine to the postprocessing exit routine.

A zero return code from IGGPRE00 indicates the DADSM function may proceed.

Rejecting a DADSM Request

A preprocessing exit routine may reject a DADSM request by providing either return code 4 or 8 to the calling DADSM function. If this occurs for partial release, then partial release provides an I/O error return code. If this occurs for either scratch or rename, they provide an I/O error status code. If a preprocessing exit rejects a call from DADSM create (allocate), then create (allocate) provides either X'B4' or X'B0', respectively, to its caller. Scheduler allocation treats X'B4' as a conditional rejection of the allocate request only for the volume being processed. If the allocate request is not for a specific volume, another volume may be chosen and the allocate function retried. Scheduler allocation treats X'B0' return code from allocate as an unconditional rejection of the allocate request. If the allocate request is rejected, the preprocessing exit routine can put a reason code in the parameter list field, IEXREASN, and the code is returned by create (allocate) to its caller, together with the X'B0' or X'B4' return code. The reason code appears as the last 2 bytes of the diagnostic information displayed as part of the IGD17040I JCL message. A nonzero return code from IGGPRE00 to extend causes extend to return an error return code of X'FFFF FFEC' to its caller. If the caller is end-of-volume, an E37-0C abend is issued.

Rejecting a DADSM Scratch Request

In the integrated catalog facility environment, VSAM deletes the VVR entry first and then calls DADSM to continue with the scratch of the format-1 DSCB. If a preprocessing exit routine rejects the DADSM request, the format-1 DSCB remains while the VVR entry no longer exists. This results in a damaged catalog. **It is your responsibility to ensure your preprocessing exits do not reject a DADSM scratch request for a VSAM data set.**

Data that DADSM Passes to the Exits

The format of the parameter list (IEPL) is shown in Figure 70.

Figure 70 (Page 1 of 3). Format of DADSM Preprocessing and Postprocessing Exit Parameter List

Name	Offset	Bytes	Description
IEXID	00(X'00')	4	EBCDIC "IEPL"
IEXLENG	04(X'04')	1	Length of parameter list
IEXFUNC	05(X'05')	1	DADSM function code
IEXALL			X'01'-Allocate
IEXEXT			X'02'-Extend
IEXSCR			X'03'-Scratch
IEXPR			X'04'-Partial release
IEXREN			X'05'-Rename
IEXPREL			X'06'-PARTREL Partial release
IEXVEXT			X'07'-Extend (VSAM caller without DEB parameter)[3]

[3] If IEXVEXT is on, you must ensure that your installation exit modules do not attempt to use the IEXPTR2 field (DEB address is undefined for this extend function).

Figure 70 (Page 2 of 3). Format of DADSM Preprocessing and Postprocessing Exit Parameter List

Name	Offset	Bytes	Description
IEXEXTCD	06(X'06')	1	Extend code X'01' Extend data set on current volume X'02' Extend an OS catalog on current volume X'04' Extend data set on new volume X'81' Extend VSAM data space on current volume
IEXFLAG	07(X'07')	1	Flag byte
IEXENQ		1...	VTOC is enqueued upon entry.
IEXVIO		.1..	VIO data set
IEXMF1		..1.	IEXFMT1 points to DS1FMTID of a partial format-1 DSCB (partial DSCB passed as input to allocate, and JFCB is not available).
IEXFDSCB		...1	Full format-1 DSCB (ALLOC=ABS)
*		 xxxx	Reserved
IEXREASN	08(X'08')	2	Installation reject reason code
*		2	Reserved
IEXUCB	12(X'0C')	4	Address of UCB. The UCB address is not available to the pre-exit for VIO allocation.
IEXPTR1[4]	16(X'10')	4	Address of one of the following: • JFCB (allocate, extend, partial release) • Data set name (PARTREL partial release) • Scratch/rename input parameter list followed by the address of parameter lists IGGDASCR or IGGDAREN.[5]
IEXPTR2	20(14)	4	Address of one of the following: • DSAB list (ISAM allocate) • DEB (extend on old volume) • DCB (partial release) • Partial DCB (PARTREL partial release) DCBFDAD and DCBDEBA are defined, the associated DEB has been constructed; DEBDSCBA, DEBNMEXT, and the DEBDASD segment(s) are defined. DEBDVMOD is not defined. • Current volume list entry (scratch/rename)
IEXDSN	24(X'18')	4	Address of the data set name

Figure 70 (Page 3 of 3). Format of DADSM Preprocessing and Postprocessing Exit Parameter List

Name	Offset	Bytes	Description
IEXFMT1	28(X'1C')	4	Address of the 96-byte data portion of format-1 DSCB (preexit for scratch; pre- and postexit for partial release and rename; postexit for allocate). May be supplied by preexit of allocate, and extend on new volume, to serve as a model if IEXMF1 and IEXVIO are zero.
IEXFMT2	32(X'20')	4	Address of format-2 DSCB. (ISAM allocate post exit.)
IEXFMT3	36(X'24')	4	Address of format-3 DSCB (ALLOC=ABS)
IEXEXTBL	40(X'28')	4	Address of DADSM table (pre- and postexit for scratch and partial release; postexit for allocate and extend). For VIO allocate postexit, this is the address of DS1EXT1 in the virtual FMT1 DSCB.
IEXDCC	44(X'2C')	4	DADSM return code (postexit). See "Setting Return Codes in IGGDASU2 and IGGDARU2" on page 147 for a list of the applicable DADSM return codes.
IEXRSVWD	48(X'30')	4	Reserved word for use by installation exit.

Passing a Model Format-1 DSCB

The preprocessing exit for allocate and extend on a new volume may return, in the parameter list field IEXFMT1, the address of the data portion of a model format-1 DSCB, starting with field DS1FMTID. The DSCB is moved to the allocate or extend work area before building the format-1 DSCB. The only fields that may be nonzero in the area are the DS1REFD (the data-last-referenced field) and fields currently unused. Failure to zero out all fields, except for DS1REFD and all currently unused fields in the model format-1 DSCB, can result in the abnormal termination of the task or lead to unpredictable results. All other fields are initialized by allocate or extend.

IEXFMT1 may not be supplied by IGGPRE00 for a VIO allocate request (indicated by flag, IEXVIO, set to one), or, if a partial DSCB instead of a JFCB has been supplied to allocate (indicated by flag, IEXMF1, set to one). In the latter case, IEXFMT1 is passed to IGGPRE00 initialized to the address of the DS1FMTID field of the partial format-1 DSCB (supplied to allocate by its caller) in the allocate work area, and DS1REFD may be initialized by IGGPRE00. If extend was successful, IEXFMT1 is zeroed out prior to taking the postexit, IGGPOST0.

---

[4] When the scheduler work area (SWA) resides above the 16M line, you may have to modify installation exit module references to the IEXPTR1 field. See "The Exit Environment" on page 137 for details.

[5] These parameter lists are described in Figure 72 on page 146 and Figure 71 on page 145.

When IGGPOST0 Gets Control

The postprocessing exit module, IGGPOST0, is given control after a DADSM function has been completed or attempted. IGGPOST0 is given control if IGGPRE00 was given control, whether the DADSM function was successful or not. IGGPOST0 is not given control if IGGPRE00 was not given control, or if the DADSM function terminated abnormally. IGGPRE00 may establish a recovery routine, if required, to clean up system resources. The DADSM recovery routine does not give IGGPOST0 control. Input to IGGPOST0 is the same parameter list passed to IGGPRE00. No return codes from IGGPOST0 are defined.

Registers at Entry to DADSM Exits

At entry to your exit routine, register contents are as follows:

Register	Contents
1	Address of the exit parameter list
13	Address of an 18-word save area
14	Return address to DADSM
15	Address of your exit routine

Registers at Return from DADSM exits

When you return to DADSM, register contents must be as follows:

Register	Contents
0-14	Same as on entry to your exit routine
15	A return code from IGGPRE00

Return Codes from DADSM Exits

No return codes are defined for IGGPOST0. The IGGPRE00 return codes and their meanings are as follows:

Code	Meaning
00(X'00')	Indicates that you want the DADSM request to be processed
04(X'04')	Indicates that no DADSM request for the current volume is to be processed
08(X'08')	Indicates that you do not want the DADSM request to be processed

Replacing DADSM Installation Exit Modules

You may replace these modules in SYS1.AOSD0 *prior* to system generation, or you may replace them in SYS1.LPALIB *after* system generation. The stage I system generation macro SGIEC4DM in SYS1.AGENLIB and the appropriate link edit step of the STAGE I system generation output are other sources of information about replacing the modules with your own versions.

Your replacement modules must follow all the characteristics and programming conventions for SVC routines. For information on these characteristics and conventions, see *Application Development Guide*.

You may apply PTFs to SCRATCH or RENAME with SMP without modifying your own versions of IGG029DM, IGG029DU, IGG030DU, IGGDASU2, IGGDASU3, IGGDARU2, and IGGDARU3.

System Control Blocks

The DADSM installation exit parameter lists contain the addresses of system control blocks. The mapping macros of those control blocks are listed below.

Macro	Control Block
DCBD	DCB
ICVARXNT	Extent Table
IECIEXPL	DADSM installation exit parameter list
IECPDSCB	Partial DSCB
IECSDSL1	DSCB
IEFJFCBN	JFCB
IEFTIOT1	TIOT
IEFUCBOB	UCB
IEZDEB	DEB
IHADSAB	DSAB

There is no mapping macro for the SCRATCH/RENAME parameter list or the associated volume list that is used when DADSM receives control from an SVC.

For extend and partial release, the address of the JFCB passed to the user exit points to a copy of the real JFCB. Updating the copied JFCB does not result in a corresponding change to the real JFCB.

During catalog processing, a dummy JFCB is built with minimal information and passed to DADSM for space allocation. Certain bits that are turned on in the real JFCB may not be turned on in this dummy JFCB. If a preprocessing exit uses this JFCB, you may need to modify the JFCB. For information on how to modify the JFCB, see "Modifying the JFCB" on page 163.

For PARTREL partial release, the DCB and DEB (see Figure 70 on page 138) have been constructed for internal DADSM processing only.

During the X'02' extend of a VSAM data set, the exit is passed the address of a dummy DEB. This DEB does not contain any extent information.

SCRATCH and RENAME Installation Exit Modules

Modules IGG029DU, IGG029DM, and IGG030DU

The load modules for DADSM SCRATCH and DADSM RENAME contain modules IGG029DU and IGG030DU, respectively. The IBM-provided IGG029DU module receives control and immediately passes control to module IGC0002I without performing any processing. The IBM-provided IGG030DU module receives control and immediately passes control to module IGC00030 without performing any processing.

The load module for DADSM SCRATCH also contains the module IGG029DM. The IBM-provided IGG029DM module receives control from IGG0290D through IGG029DN when an error return code of either 4 or 8 is indicated, and immediately passes control to the location pointed to by register 14 without performing any processing.

If you require special processing either before or after SCRATCH or RENAME, replace the appropriate IBM-provided module(s) with your own module(s). IGG029DU, IGG030DU, and IGG029DM may request control and pass control in either 24-bit or 31-bit addressing mode. The modules may reside either above or below 16Mb virtual. If you have replaced them, you may wish to change them to benefit fully from 31-bit addressing support. For example, if your parameter list resides above the 16M line, but your replacement modules are not defined as AMODE 31, DADSM copies the parameter list, incurring additional overhead. If you are not invoking DADSM SCRATCH or RENAME through an SVC, IGG029DU and IGG030DU are not given control.

Modules IGGDASU2, IGGDASU3, IGGDARU2, and IGGDARU3

The load module IGGDADSM contains modules IGGDASU2, IGGDASU3, IGGDARU2, and IGGDARU3. These four modules perform no processing and can be replaced. When DADSM SCRATCH is invoked, IGGDASU2 receives control before any SCRATCH processing and IGGDASU3 receives control immediately after SCRATCH processing. When DADSM RENAME is invoked, IGGDARU2 receives control before any RENAME processing and IGGDARU3 receives control immediately after RENAME processing.

When DADSM SCRATCH or RENAME is invoked, and you require special processing either before or after DADSM SCRATCH or RENAME, replace the appropriate IBM-provided module(s) with your own module(s). Your modules must be reentrant, execute in 31-bit addressing mode, and observe standard register linkage conventions. The modules receive control in PSW key 5 and obtain storage for the parameter lists in key 5. Your modules that replace IGGDASU2 and IGGDASU3 receive parameter list IGGDASCR as input, and your modules that replace IGGDARU2 and IGGDARU3 receive parameter list IGGDAREN as input. For descriptions of IGGDASCR and IGGDAREN, see Figure 71 on page 145 and Figure 72 on page 146. The DADSM volume list is also passed as input to the installation replaceable modules IGGDARU2, IGGDARU3, IGGDASU2, and IGGDASU3. For a description of the volume list, see Figure 73 on page 146.

When DADSM SCRATCH or RENAME is invoked through an SVC, the module IGG029DU or IGG030DU receives control first. Either IGGDASU2 and IGGDASU3 (SCRATCH) or IGGDARU2 and IGGDARU3 (RENAME) will receive control after

any processing done by IGG029DU or IGG030DU has completed. A flag bit in the parameter list (IGGDASCR for SCRATCH or IGGDAREN for RENAME) is set whenever DADSM SCRATCH or RENAME is invoked with an SVC. Your modules that replace IGGDASU2, IGGDASU3, IGGDARU2, and IGGDARU3 can check this bit to determine whether IGG029DU or IGG030DU has already received control and completed processing. The flag bit for SCRATCH (in the IGGDASCR parameter list) is DASSVCEP. The flag bit for RENAME (in the IGGDAREN parameter list) is DARSVCEP.

Figure 71. Format of the DADSM SCRATCH Parameter List (IGGDASCR)

Offset	Bytes	Name	Description
0 (X'00')	44	IGGDASCR	DADSM DELETE parameter list
0 (X'0')	8	DASPLID	ID = 'IGGDASCR'
8 (X'8')	2	DASPVER	Version of parameter list
10 (X'A')	2	DASPLEN	Length of parameter list
12 (X'C')	1	DASPKEY	Key of parameter list
	xxxx	DASPSKEY	Storage key of parameter list
	 xxxx		Reserved
13 (X'D')	1		Reserved
14 (X'E')	2	DASHRTCD	DADSM SCRATCH return code
16 (X'10')	4	DASDIAGI	Diagnostic information
16 (X'10')	1	DASERRCD	DADSM SCRATCH error code
17 (X'11')	1	DASSFNID	DADSM SCRATCH subfunction ID
18 (X'12')	1	DASSFRET	Subfunction return code
19 (X'13')	1	DASSFREA	Subfunction reason code
20 (X'14')	1		Reserved
21 (X'15')	1	DASFLAG2	SMS indicator flag
	X'80'	DASSMSMG	SMS-managed data set
	X'40'	DASUNCAT	Uncataloged SMS-managed data set
	X'20'	DASCATCL	Catalog is the caller
	X'10'	DASGDGRO	GDG rolloff in progress
	X'08'	DASGDGRN	GDG rolloff noscratch
	xxx		Reserved
22 (X'16')	1	DASFLAG3	Functionally authorized request flags (Part 1)
	X'80'	DASFAUTH	Caller guarantees functional authorization of request
	X'40'	DASSAUTH	Caller guarantees security authorization of request
	X'20'	DASPROFM	RACF profile managed by caller
	X'10'	DASBTIOT	Bypass SYSZTIOT ENQ
	X'08'	DASBDSN	Bypass SYSDSN ENQ
	X'04'	DASBDEB	Bypass DEB check (DSN not open)
	X'02'	DASNVALL	Do not dynamically allocate volumes
	X'01'	DASNERAS	Do not erase any extents
23 (X'17')	1	DASFLAG4	Functionally authorized request flags (Part 2)
	X'80'	DASVOLMT	Caller guarantees that the volume is already mounted
	.xxx xxxx		Reserved
24 (X'18')	1	DASFLAG5	Auth not required request flags
	X'80'	DASOVRPD	Override purge date
	X'40'	DASVRFRD	Verify last-referenced-date unchanged (DFHSM)
	X'20'	DASERASE	Erase all extents
	...x xxxx		Reserved
25 (X'19')	1	DASFLAG6	Parameter flag byte
	x...	DASSVCEP	1 = SCRATCH entered at SVC entry point, 0 = SCRATCH entered via VDSS branch entry interface
	.xxx xxxx		Reserved
26 (X'1A')	2		Reserved
28 (X'1C')	4	DASUCB	Address of primary mountable UCB
28 (X'1C')	4	DASVDSCB	Address of virtual DSCB, VIO data set
32 (X'20')	4	DASAVOLL	Address of volume list (IGGDAVLL); see Figure 73 on page 146
36 (X'24')	4	DASADSN	Address of data set name
40 (X'28')	3	DASREFDT	Reference date to check (DFHSM)
43 (X'2B')	1		Reserved

Figure 72. Format of the DADSM RENAME Parameter List (IGGDAREN)

Offset	Bytes	Name	Description
0 (X'00')	44	IGGDAREN	DADSM RENAME parameter list
0 (X'0')	8	DARPLID	ID = 'IGGDAREN'
8 (X'8')	2	DARPVER	Version of parameter list
10 (X'A')	2	DARPLEN	Length of parameter list
12 (X'C')	1	DARPKEY	Key of parameter list
	xxxx	DARPSKEY	Storage key of parameter list
	 xxxx		Reserved
13 (X'D')	1		Reserved
14 (X'E')	2	DARHRTCD	DADSM RENAME return code
16 (X'10')	4	DARDIAGI	Diagnostic information
16 (X'10')	1	DARERRCD	DADSM RENAME error code
17 (X'11')	1	DARSFNID	DADSM RENAME subfunction ID
18 (X'12')	1	DARSFRET	Subfunction return code
19 (X'13')	1	DARSFREA	Subfunction reason code
20 (X'14')	1		Reserved
21 (X'15')	1	DARFLAG2	SMS indicator flag
	X'80'	DARSMSMG	SMS managed data set
	X'40'	DARUNCAT	Rename uncataloged data sets only
	..xx xxxx		Reserved
22 (X'16')	1	DARFLAG3	Functionally authorized request flags (Part 1)
	X'80'	DARFAUTH	Caller guarantees functional authorization of request
	X'40'	DARSAUTH	Caller guarantees security authorization of request
	X'20'	DARPROFM	RACF profile managed by caller
	...x xxxx		Reserved
23 (X'17')	1	DARFLAG4	Functionally authorized request flags (Part 2)
	x...	DARSVCEP	1 = RENAME entered at SVC entry point, 0 = RENAME entered via VDSS branch entry interface
	.x..	DARBPDSC	When this bit is on, do not update "data set changed" bit in the format-1 DSCB
	..xx xxxx		Reserved
24 (X'18')	4		Reserved
28 (X'1C')	4	DARUCB	Address of primary mountable UCB
32 (X'20')	4	DARAVOLL	Address of volume list (IGGDAVLL); see Figure 73.
36 (X'24')	4	DARADSN	Address of old data set name
40 (X'28')	4	DARANDSN	Address of new data set name

Figure 73. Format of the DADSM Volume List (IGGDAVLL)

Offset	Bytes	Name	Description
0 (X'00')	16	IGGDAVLL	DADSM volume list
0 (X'00')	16	DAVLLHDR	Volume list header
0 (X'00')	8	DAVLLID	ID = 'IGGDAVLL'
8 (X'08')	2	DAVLVER	Version of volume list
10 (X'0A')	2	DAVLLEN	Length of volume list header
12 (X'0C')	1	DAVLKEY	Key of volume list
	xxxx	DAVLSKEY	Storage key of parameter list
	 xxxx		Reserved
13 (X'0D')	1		Reserved
14 (X'0E')	2	DAVCOUNT	Number of volumes
16 (X'10')	12	DAVLVOLE	Volume entries; number = DAVCOUNT
16 (X'10')	4	DAVLUCBT	Device type
20 (X'14')	6	DAVLVOLS	Volume serial number
26 (X'1A')	1	DAVLSSTC	Secondary status byte
27 (X'1B')	1	DAVLSTAT	SCRATCH/RENAME status byte

Setting Return Codes in IGGDASU2 and IGGDARU2

The preprocessing modules, IGGDASU2 and IGGDARU2, return to DADSM with a return code in register 15. If the return code is nonzero, the DADSM function terminates. For a return code of 4, the DADSM function passes back a return code of zero to its caller. If the return code is greater than 4, the DADSM function passes back a return code of 4 to its caller.

DADSM SVC Return Code	Meaning
00(X'00')	Indicates successful data set rename.
04(X'04')	Indicates that no volume containing any part of the data set was mounted, nor was a unit available for mounting.
08(X'08')	Indicates that an unusual condition was encountered on one or more volumes.
12(X'0C')	Indicates that either the DADSM RENAME parameter list or the volume list is invalid.

Figure 74. DADSM RENAME Return Codes

DADSM SVC Return Code	Meaning
00(X'00')	Indicates successful data set deletion.
04(X'04')	Indicates that no volume containing any part of the data set was mounted, nor was a unit available for mounting.
08(X'08')	Indicates that an unusual condition was encountered on one or more volumes.
12(X'0C')	Indicates that either the DADSM SCRATCH parameter list or the volume list is invalid.

Figure 75. DADSM SCRATCH Return Codes

The postprocessing modules, IGGDASU3 and IGGDARU3, do not set return codes.

CATALOG Installation Exit Module

The load module for CATALOG contains module IGG026DU. The IBM-provided IGG026DU module receives control from CATALOG and immediately passes control to module IGC0002F without performing any processing.

If you require special processing either before or after system catalog routines receive control, replace the IBM-provided module with your own module. You may replace this module in SYS1.AOSD0 *prior* to system generation, or you may replace it in SYS1.LPALIB *after* system generation. The stage I system generation macro SGIEC4DM in SYS1.AGENLIB and the appropriate link edit step of the STAGE I system generation output are other sources of information about replacing the modules with your own versions.

Your replacement module must receive control in 31-bit addressing mode and pass control to IGC0002F in 31-bit addressing mode. If the your module needs to save registers, it must provide a re-entrant save area to save and restore registers. The SVRB save area should not be used because it is used by the catalog routines.

This module must be compiled and executed on a machine with an EBCDIC character set. Your replacement module must be re-entrant and read-only. It must follow all the characteristics and programming conventions for SVC routines. For information on these characteristics and conventions, see *Application Development Guide*.

You may apply PTFs to CATALOG with SMP without modifying your own version of IGG026DU.

The Exit Environment

The input to IGG026DU is in the form of a catalog request parameter list pointed to by register 1. The list may be in the form of an OS/VS CAMLST or a VSAM request list.

The register contents on entry to IGG026DU are as follows:

Register	Contents
0	Not Used.
1	Catalog parameter list address.
2	Not used.
3	Pointer to the CVT.
4	Pointer to the TCB.
5	Pointer to the SVRB.
6 - 13	Not used.
14	Address of exit prolog
15	Not used.

Figure 76. Register Contents on Entry to Catalog Installation Exit

The output from module IGG026DU is the entry point address of of catalog module IGC0002F, stored in register 15. Registers 1 through 14 should be the same as they were on entry to IGG026DU.

DASD Calculation Services (DCS) Installation Exits

DASD calculation services (DCS) retrieves DASD data set information, performs calculations, and returns statistics to the caller of DCS. DCS provides data set information primarily for display by ISMF (Interactive Storage Management Facility). The values returned are designated in kilobytes (Kb) or bytes rather than cylinders or tracks, to eliminate device dependency.

DCS allows for two installation-written exit modules, the precalculation exit (IGBDCSX1) and the postcalculation exit (IGBDCSX2), to provide flexibility in selecting the optimum block size/CI size. Because the access methods restrict maximum block size to 32760, if an exit module returns an override or limit greater than this, DCS sets the block size to 32760. DCS also verifies that exit-supplied CI size override values do not violate VSAM restrictions.

The DCS installation exit routines receive control and execute in the calling program's key and system state (problem/supervisor). The exit CSECTs are linked together with the Common Filter Services, Device Information Services, and DCS CSECTs into a single load module. They must be programmed to run in 31-bit mode and must reside above the 16Mb line. DCS provides 1K bytes of working storage for each of the exits. Figure 78 on page 151 and Figure 79 on page 155 contain sample precalculation and postcalculation exit routines to document usage and provide models for you.

Data That DCS Passes to the Exits

The IGBDCSIE macro maps the DCS pre/postcalculation exit parameter list. At entry to the exits, register 1 points to a field containing the address of the parameter list. Figure 77 shows the format of the DCS Precalculation and Postcalculation Exit parameter list.

Figure 77. Format of the DCS Precalculation and Postcalculation Exit Parameter List

Name	Offset	Bytes	Description
DCSIEPL			DCS exit parameter list
DCSIEDSN	00(X'00')	44	Data set name
DCSIEDSO	44(X'2C')	4	Data set organization
DCSIEKP	48(X'30')	4	Key position
DCSIELRL	52(X'34')	4	Logical record length (average record length if VSAM)
DCSIETC	56(X'38')	4	Track capacity
DCSIEBUF	60(X'3C')	4	Buffer space
DCSIESTG	64(X'40')	4	Exit workspace address
DCSIEKL	68(X'44')	2	Key length
DCSIEBS	70(X'46')	2	Block size (current physical block size if VSAM)
DCSIECOB	72(X'48')	2	Calculated optimum block size
DCSIEVSN	74(X'4A')	6	Volume serial number

Registers at Entry to the DCS Exits

At entry to your exit routine, register contents are as follows:

Register	Contents
1	Pointer to the address of the exit parameter list
13	Address of an 18-word save area
14	Return address to DCS
15	Address of your exit routine

Registers at Return from the DCS Exits

When you return to DCS, register contents must be as follows:

Register	Contents
0	Dependent upon which exit is returning and the return code in register 15.
1-14	Same as on entry to your exit routine
15	A return code from the exit routine

IGBDCSX1 (DCS Precalculation Installation Exit)

This installation exit routine gains control before DCS calculates the statistics you requested. You can use it to either bypass or limit the DCS-calculated optimum block size/CI size. See "Registers at Entry to the DCS Exits" on page 149 and "Data That DCS Passes to the Exits" on page 149.

Return Codes from the Precalculation Exit

The precalculation installation exit must pass a return code back to DCS in register 15. The return codes and their meanings are as follows:

Code	Meaning
00(X'00')	Indicates that DCS can proceed normally
04(X'04')	Indicates that DCS can proceed, using the unsigned value in register 0 as the maximum possible value.
08(X'08')	Indicates that DCS should bypass calculating statistics and use the block size/CI size provided in register 0.

Sample Precalculation Exit

Figure 78 on page 151 contains a sample exit for you to use as a model.

```
*************************************************************************
*                                                                       *
*  $MOD(IGBDCSX1): DASD CALCULATION SERVICES PRE-CALCULATION EXIT       *
*                                                                       *
*  DESCRIPTIVE NAME = DCS PRE-CALCULATION EXIT                          *
*                                                                       *
*  COPYRIGHT = NONE                                                     *
*                                                                       *
*  FUNCTION = THIS MODULE IS AN EXAMPLE OF THE PRE-CALCULATION EXIT     *
*     CALLED BY DASD CALCULATION SERVICES.  THE EXIT IS PROVIDED BY     *
*     THE INSTALLATION AND, IN GENERAL, IS THE MEANS BY WHICH THE       *
*     INSTALLATION TELLS DCS THAT IT WISHES TO LIMIT THE OPTIMAL CI     *
*     SIZE OR THE OPTIMAL BLOCKSIZE OR BYPASS THE DCS CALCULATION       *
*     ENTIRELY IN LIEU OF AN INSTALLATION-SUPPLIED VALUE.               *
*                                                                       *
*     THIS IS A SAMPLE EXIT WHICH RETURNS THE INSTALLATION-DEFINED      *
*     MAXIMUM BLOCKSIZE FOR NON-VSAM DATASETS WHOSE FIRST LEVEL         *
*     QUALIFIER IS SYS2 OR SYS3.  FOR NON-VSAM DATASETS WHOSE FIRST     *
*     LEVEL QUALIFIER IS SYS4 OR SYS5, THIS EXIT RETURNS THE ONLY       *
*     BLOCKSIZE ALLOWED.  FOR ALL OTHER DATASETS, NO                    *
*     RESTRICTIONS APPLY.                                               *
*                                                                       *
*  NOTE: THIS SAMPLE EXIT IS INTENDED TO BE USED AS A LEARNING          *
*        AID FOR THE INSTALLATION SYSTEM PROGRAMMER AND IS NOT          *
*        GUARANTEED TO RUN ON A PARTICULAR SYSTEM WITHOUT SOME          *
*        MODIFICATION.                                                  *
```

Figure 78 (Part 1 of 4). Sample DASD Precalculation Exit

```
*                                                                     *
*     REGISTER CONVENTIONS =                                          *
*        ON ENTRY:                                                    *
*          REGISTER  1 = PARAMETER LIST ADDRESS                       *
*          REGISTER 13 = CALLER SAVE AREA ADDRESS                     *
*          REGISTER 14 = RETURN ADDRESS                               *
*          REGISTER 15 = RETURN CODE                                  *
*        ON EXIT:                                                     *
*          REGISTER  0 = BLOCK/CI SIZE IF INDICATED BY RETURN CODE    *
*          REGISTER 15 = RETURN CODE                                  *
*                                                                     *
*     PROCESSOR = ASSEMBLER                                           *
*                                                                     *
*     ATTRIBUTES = CALLER KEY, CALLER STATE, ENABLED,                 *
*                  AMODE(31),RMODE(ANY)                               *
*                                                                     *
*  ENTRY POINT = IGBDCSX1                                             *
*                                                                     *
*  OUTPUT = RETURN CODE IN REGISTER 15, BLOCKSIZE IN REGISTER 0       *
*                                                                     *
*    0 = PROCEED WITHOUT RESTRAINT                                    *
*    4 = PROCEED WITH CALCULATION BUT USE THE UNSIGNED VALUE IN       *
*        REGISTER 0 AS THE MAXIMUM POSSIBLE VALUE                     *
*    8 = BYPASS CALCULATION AND USE BLOCKSIZE PROVIDED IN             *
*        REGISTER 0                                                   *
*                                                                     *
***********************************************************************
          EJECT
IGBDCSX1  CSECT
IGBDCSX1  AMODE 31
IGBDCSX1  RMODE ANY
*
*   SET UP ADDRESSABILITY
*
          STM     REG14,REG12,12(REG13)  SAVE CALLER'S REGS
          LR      REG12,REG15        LOAD IGBDCSX1 ADDR INTO BASE REG
          USING   IGBDCSX1,REG12
          L       REG1,0(,REG1)      SET UP PARM LIST ADDRESSABILITY
          USING   DCSIEPL,REG1
*
*   CHECK FOR VSAM DATA SET
*
          TM      DCSIEDSO+1,DCSIEAM  TEST DATA SET ORGANIZATION
          BO      EXIT               BRANCH IF VSAM DATA SET
*
*   CHECK FIRST-LEVEL QUALIFIER OF DATA SET NAME FOR SYS2
*
          CLC     DCSIEDSN(4),SYS2   IS DSN SYS2?
          BNE     SYS3CHK            NO - CHECK FOR SYS3
          L       REG0,MAXSYS2       SET MAX BLOCKSIZE FOR SYS2 DS
          LA      REG15,4            SET RETURN CODE INDICATING A
*                                    LIMIT WAS SET
          B       EXIT
```

Figure 78 (Part 2 of 4). Sample DASD Precalculation Exit

```
*
*   CHECK FIRST-LEVEL QUALIFIER OF DATA SET NAME FOR SYS3
*
SYS3CHK   CLC     DCSIEDSN(4),SYS3   IS DSN SYS3?
          BNE     SYS4CHK            NO - CHECK FOR SYS4
          L       REG0,MAXSYS3       SET MAX BLOCKSIZE FOR SYS3 DS
          LA      REG15,4            SET RETURN CODE INDICATING A
*                                    LIMIT WAS SET
          B       EXIT
*
*   CHECK FIRST-LEVEL QUALIFIER OF DATA SET NAME FOR SYS4
*
SYS4CHK   CLC     DCSIEDSN(4),SYS4   IS DSN SYS4?
          BNE     SYS5CHK            NO - CHECK FOR SYS5
          L       REG0,SYS4BSZ       SET ONLY BLOCKSIZE FOR SYS4 DS
          LA      REG15,8            SET RETURN CODE INDICATING TO
*                                    BYPASS CALCULATION
          B       EXIT
*
*   CHECK FIRST-LEVEL QUALIFIER OF DATA SET NAME FOR SYS5
*
SYS5CHK   CLC     DCSIEDSN(4),SYS5   IS DSN SYS5?
          BNE     NOLIMIT            NO - NO LIMITS SET
          L       REG0,SYS5BSZ       SET ONLY BLOCKSIZE FOR SYS5 DS
          LA      REG15,8            SET RETURN CODE INDICATING TO
*                                    BYPASS CALCULATION
          B       EXIT
          EJECT
*
*   INDICATE NO LIMITS SET
*
NOLIMIT   LA      REG15,0
*
*   RETURN TO DCS
*
EXIT      EQU     *
          L       REG14,12(,REG13)   RESTORE CALLER'S REG 14
          LM      REG1,REG12,24(REG13)  RESTORE REST OF CALLER'S REGS
          BR      REG14              BRANCH BACK TO CALLER
          EJECT
*
*   DEFINE VARIABLES
*
SYS2      DC      C'SYS2'
SYS3      DC      C'SYS3'
SYS4      DC      C'SYS4'
SYS5      DC      C'SYS5'
*
REG0      EQU     0
REG1      EQU     1
REG12     EQU     12
REG13     EQU     13
REG14     EQU     14
REG15     EQU     15
```

Figure 78 (Part 3 of 4). Sample DASD Precalculation Exit

```
*
MAXSYS2   DC      F'5120'
MAXSYS3   DC      F'10240'
SYS4BSZ   DC      F'4096'
SYS5BSZ   DC      F'8192'
*
*
          IGBDCSIE
*
          END     IGBDCSX1
```

Figure 78 (Part 4 of 4). Sample DASD Precalculation Exit

IGBDCSX2 (DCS Postcalculation Installation Exit)

This installation exit routine gains control after DCS calculates the statistics you requested. You can use it to override the DCS-calculated optimum block size/CI size with a value of your own. See "Registers at Entry to the DCS Exits" on page 149 and "Data That DCS Passes to the Exits" on page 149.

Return Codes from the Postcalculation Exit

The postcalculation installation exit must pass a return code back to DCS in register 15. The return codes and their meanings are as follows:

Code	Meaning
00(X'00')	Indicates that the exit accepts the calculated block size/CI size.
08(X'08')	Indicates that the exit wants to override the DCS-calculated block size/CI size with the value specified in register 0.

Sample Postcalculation Exit

Figure 79 on page 155 contains a sample exit for you to use as a model.

```
*********************************************************************
*                                                                   *
*  $MOD(IGBDCSX2):  DASD CALCULATION SERVICES POST-CALCULATION EXIT *
*                                                                   *
*  DESCRIPTIVE NAME = DCS POST-CALCULATION EXIT                     *
*                                                                   *
*  COPYRIGHT = NONE                                                 *
*                                                                   *
*  FUNCTION = THIS MODULE IS AN EXAMPLE OF THE POST-CALCULATTION EXIT *
*     CALLED BY DASD CALCULATION SERVICES.  THE EXIT IS PROVIDED    *
*     BY THE INSTALLATION AND, IN GENERAL, IS THE MEANS BY WHICH THE *
*     INSTALLATION TELLS DCS THAT IT WISHES TO OVERRIDE THE DCS     *
*     CALCULATION ENTIRELY IN LIEU OF AN INSTALLATION-SUPPLIED VALUE. *
*                                                                   *
*     THIS IS A SAMPLE EXIT WHICH RETURNS A BLOCKSIZE TO OVER-      *
*     RIDE THE CALCULATED OPTIMAL BLOCKSIZE OF A NON-VSAM DATA SET  *
*     IF THE CALCULATED BLOCKSIZE EXCEEDS AN INSTALLATION LIMIT AND *
*     THE DATA SET RESIDES ON A PARTICULAR VOLUME.                  *
*                                                                   *
*  NOTE: THIS SAMPLE EXIT IS INTENDED TO BE USED AS A LEARNING      *
*        AID FOR THE INSTALLATION SYSTEM PROGRAMMER AND IS NOT      *
*        GUARANTEED TO RUN ON A PARTICULAR SYSTEM WITHOUT SOME      *
*        MODIFICATION.                                              *
*                                                                   *
*     REGISTER CONVENTIONS =                                        *
*        ON ENTRY:                                                  *
*          REGISTER  1 = PARAMETER LIST ADDRESS                     *
*          REGISTER 13 = CALLER SAVE AREA ADDRESS                   *
*          REGISTER 14 = RETURN ADDRESS                             *
*          REGISTER 15 = RETURN CODE                                *
*        ON EXIT:                                                   *
*          REGISTER  0 = BLOCK/CI SIZE IF INDICATED BY RETURN CODE  *
*          REGISTER 15 = RETURN CODE                                *
*                                                                   *
*     PROCESSOR = ASSEMBLER                                         *
*                                                                   *
*     ATTRIBUTES = CALLER KEY, CALLER STATE, ENABLED,               *
*                  AMODE(31),RMODE(ANY)                             *
*                                                                   *
*  ENTRY POINT = IGBDCSX2                                           *
*                                                                   *
*  OUTPUT = RETURN CODE IN REGISTER 15, BLOCKSIZE IN REGISTER 0     *
*                                                                   *
*    0 = THE CALCULATED BLOCKSIZE IS ACCEPTED                       *
*    8 = THE CALCULATED BLOCKSIZE IS TO BE OVERRIDDEN WITH THE      *
*        VALUE SPECIFIED IN REGISTER 0                              *
*                                                                   *
*********************************************************************
         EJECT
IGBDCSX2  CSECT
IGBDCSX2  AMODE 31
IGBDCSX2  RMODE ANY
```

Figure 79 (Part 1 of 2). Sample DASD Postcalculation Exit

```
*
*   SET UP ADDRESSABILITY
*
          STM     REG14,REG12,12(REG13)  STORE CALLER'S REGS
          LR      REG12,REG15        LOAD IGBDCSX1 ADDR INTO BASE REG
          USING   IGBDCSX2,REG12
          L       REG1,0(,REG1)      SET UP PARM LIST ADDRESSABILITY
          USING   DCSIEPL,REG1
*
*   CHECK FOR BLOCKSIZE GREATER THAN INSTALLATION LIMIT IF THE DATA
*   SET IS NON-VSAM AND RESIDES ON THE SPECIFIED VOLUME
*
          TM      DCSIEDSO+1,DCSIEAM CHECK FOR VSAM DATA SET
          BO      EXIT               BRANCH IF VSAM DATA SET
          SPACE
          CLC     DCSIEVSN(6),VOLSER DOES DS RESIDE ON THIS VOLUME?
          BNE     BLKSZOK            NO - BLOCK SIZE IS OK
          SPACE
          CLC     MAXBLKSZ(2),DCSIECOB  IS CALCULATED BIGGER?
          BNH     BLKSZOK            NO - BLOCK SIZE IS ACCEPTABLE
          SPACE
          LH      REG0,MAXBLKSZ      SET MAX BLOCKSIZE
          LA      REG15,8            SET RETURN CODE INDICATING AN
*                                    OVERRIDE VALUE
          B       EXIT
*
*   CALCULATED BLOCK SIZE IS OK
*
BLKSZOK   EQU     *
          SR      REG15,REG15
*
*   RETURN TO DCS
*
EXIT      EQU     *
          L       REG14,12(,REG13)  RESTORE CALLER'S REG14
          LM      REG1,REG12,24(REG13)  RESTORE REST OF CALLER'S REGS
          BR      REG14             BRANCH BACK TO CALLER
          EJECT
*
*   DEFINE VARIABLES
*
REG0      EQU     0
REG1      EQU     1
REG12     EQU     12
REG13     EQU     13
REG14     EQU     14
REG15     EQU     15
*
MAXBLKSZ  DC      H'31744'
VOLSER    DC      C'SPECIL'
*
*
          IGBDCSIE
*
          END     IGBDCSX2
```

Figure 79 (Part 2 of 2). Sample DASD Postcalculation Exit

Data Management Abend Installation Exit

The abend installation exit provides the ability to recover from abnormal conditions that may occur during the opening, closing, or handling of an end-of-volume condition for a non-VSAM data set associated with the user's task.

When an abnormal condition occurs, control passes to the DCB abend user exit routine if one is provided, and processing continues as specified in the DCB abend user exit routine. (The DCB abend user exit routine gives you some options regarding the actions you want the system to take when a condition arises that may result in abnormal termination of your task. For additional information about the DCB abend user exit routine, see "DCB Abend Exit" on page 62.) However, if the DCB abend user exit routine is not specified, or if it specifies immediate abnormal termination of the task, the system passes control to the abend installation exit. If a DCB abend user exit routine is not provided, control immediately passes to the abend installation exit.

IBM supplies an installation exit module, IFG0199I, in SYS1.LPALIB, that handles abend situations caused by tape positioning errors. IFG0199I allows you to retry tape positioning when you receive a system completion code 613 with return code 08 or 0C. To perform recovery actions for data management abend situations (other than those caused by tape positioning errors), you can replace installation exit module IFG0199I by modifying the source code supplied in member OPENEXIT of SYS1.SAMPLIB. IFG0199I receives control in protection key zero, supervisor state. IFG0199I checks the system completion code and the return code to determine whether the abend situation is the result of a tape positioning error. If the system completion code is other than 613 with return code 08 or 0C, control returns to the calling module with return code 0, indicating that the abend should continue. Otherwise, IFG0199I checks the counter in the 4-byte work area to determine whether one attempt to reposition the tape has been made. If no attempt to reposition the tape has been made, IFG0199I issues a return code of 4, indicating that positioning should be retried. If one attempt to reposition the tape has been made, IFG0199I issues message IEC613A to the operator to determine whether to attempt repositioning. If the operator specifies that tape positioning is to be attempted again, a return code of 4 is set, indicating that OPEN is to rewind the tape and attempt positioning. If the operator specifies that tape positioning is not to be retried, control is returned to the calling module with a 0 return code.

Data That OPEN/EOV Passes to the Exit

The format of the parameter list (OAIXL) is shown in Figure 80 on page 158.

Word Boundary

+0(00)	User Prot Key	Option Flags	Reserved	Reserved
+4(04)	Address of the protected copy of the DCB			
+8(08)	Address of the user's DCB related to the abend			
+12(0C)	Address of the UCB related to the abend			
+16(10)	Address of the JFCB related to the abend			
+20(14)	Address of the TIOT entry related to the abend			
+24(18)	Abend code - Example '6130000C'			
+28(1C)	4-byte installation work area			

1(01) Option flags:

Bits

0 indicates whether the DCB abend user exit was taken

On exit was taken
Off exit was not taken

1 indicates whether to rewind the tape volume

On rewind the tape volume
Off do not rewind the tape volume

Figure 80. Format of the Parameter List OAIXL

Registers at Entry to the Data Management ABEND Exit

At entry to the exit routine, register contents are as follows:

Register	Contents
1	Address of the parameter list (OAIXL)
13	Address of an 18-word save area
14	Return address to OPEN/EOV
15	Address of the entry point to IFG0199I

Registers at Return from the Data Management ABEND Exit

When you return to OPEN/EOV, register contents must be as follows:

Register	Contents
2-12	Same as on entry to the exit
15	A return code from the exit

Return Codes from the Data Management ABEND Exit

The data management ABEND exit must pass a return code back to OPEN/EOV as follows:

Code	Meaning
00(X'00')	Continue with the abend in process.
04(X'04')	If the bit 1 option flag is on, rewind the tape volume, set the UCBFSCT and UCBFSEQ fields in the UCB to zero, and try to recover from the abend. If the bit 1 option flag is off, try to recover from the abend.

For abend codes for which the installation is allowed to try to recover, see "DCB Abend Exit" on page 62

Modifying the IBM-Supplied Installation Exit Module: Because the IBM-supplied installation exit module handles only a particular abend situation, you may want to modify the source code of that module to perform corrective actions for other abend situations.

You can obtain a copy of the source code from member OPENEXIT of SYS1.SAMPLIB for modification, using the editing function that is available to you. After you have modified the source code, link-edit it into SYS1.LPALIB. The source program is written in Assembler language. If you replace the supplied installation module, the exit module you supply must have the entry point name IFG0199I and it must be reenterable.

DCB OPEN Installation Exit (IFG0EX0B)

The OPEN exit enables an installation-written module to gain control during Open for a DCB. OPEN uses an exit parameter list to communicate with exit module. The format of the parameter list is shown in Figure 81 on page 161.

The Exit Module

OPEN has an exit module that the installation can replace. The module name is IFG0EX0B and it is part of load module IGC0001I. IGC0001I resides in SYS1.LPALIB. You can use System Modification Program (SMP) to replace the IBM-supplied exit module with an installation exit you write.

The Exit Environment

IFG0EX0B is given control in supervisor state and protect key zero with no locks held. System enqueues are issued to serialize system functions. These enqueues may prevent other system services from being invoked. In particular, dynamic allocation, OPEN, CLOSE, EOV, and DADSM functions should not be invoked because of an enqueue on the SYSZTIOT resource. If the exit requires access to an installation data set, the control blocks required to access that data set (DCB, DEB) should be built during system initialization (IPL/NIP). RACF macros may be invoked from the exit.

Open Processing before the DCB OPEN Exit Gets Control

The exit module, IFG0EX0B, is given control whenever OPEN processes a DCB. The exit is taken after the following functions have been performed for the DCB.

- DASD data sets
  - Volume mounted
  - Format-1, -2, and -3 DSCBs read
  - Forward merge from format-1 DSCB to JFCB
- Tape data sets
  - Volume mounted
  - Header labels verified
  - Forward merge from header labels to JFCB
- All data sets
  - Forward merge from JFCB to DCB
  - User DCB OPEN installation exit (if any) taken
  - RACF or password verification processing

Open Processing after the DCB OPEN Exit Gets Control

The following functions have not yet been performed at the time the exit is given control for the DCB.

- Reverse merge from DCB to JFCB (not all fields are merged)
- Reverse merge from JFCB to format-1 DSCB for DASD data sets (not all fields are merged)
- Header labels written (for output tape data set)
- Access-method-dependent processing (obtain buffers, GETMAIN, and build IOBs and DEB)
- Write JFCB
- Write format-1 DSCB
- Obtain system block size if block size in DCB is zero.

Getting Control from Open

The exit is given control for each DCB being opened, even when two or more DCBs are being opened in parallel with one invocation of OPEN.

The exit is given control from OPEN and OPEN TYPE=J. The exit is given control from end-of-volume (EOV) and from force-end-of-volume (FEOV) when a concatenation of two *unlike* sequential data sets is being processed. (*Unlike* meaning that the user program considers the data sets to have unlike attributes.) The exit is not given control when a concatenation of two *like* sequential data sets is being processed. (*Like* meaning that the user program considers the data sets to have like attributes.) Such data sets might be a combination of disk, tape, and sysin data sets. A request by the user program for concatenation with unlike attributes is shown in the DCB by flag DCBOFPPC (bit 4; mask X'08') in field DCBOFLGS being set to one.

Data That Open Passes to the Exit

The parameter list mapped by macro IECOIEXL is supplied to the installation exit. It contains data and the addresses of control blocks that may be of interest to the exit.

The format of the parameter list is shown in Figure 81.

Name	Offset	Bytes	Description
OIEXL	00(X'00')	0	DCB Open installation exit parameter list
OIEXOOPT	00(X'00')	1	Open option (last 4 bits)
OIEXRSVD		1111	X'F0' first 4 bits reserved
OIEXOOUT		 1111	15 output
OIEXOOIN		 0111	7 outin
OIEXOUPD		 0100	4 update
OIEXOINO		 0011	3 inout
OIEXORDB		 0001	1 read backward
OIEXOINP		 0000	0 input
OIEXUKEY	01(X'01')	1	User protect key-key of user DCB
OIEXLTH	02(X'02')	2	Length of OIEXL
OIEXUDCB	04(X'04')	4	Address of user DCB in user protect key (OIEXUKEY)
OIEXPDCB	08(X'08')	4	Address of protected copy of DCB used by OPEN
OIEXJFCB	12(X'12')	4	Address of JFCB
OIEXDSCB	16(X'16')	4	Address of data portion of format-1 DSCB
OIEXTIOT	20(X'20')	4	Address of TIOT entry
OIEXUCB	24(X'24')	4	Address of UCB

Figure 81. Format of DCB OPEN Installation Exit Parameter List (OIEXL)

Note that two DCB addresses are supplied. OPEN maintains a protected copy of the user DCB. You can use OPEN's copy of the DCB to test the DCB fields. If you modify your copy of the DCB, OPEN updates its protected copy when it regains control from the exit. The protect key of the user DCB is supplied in the exit parameter list. You must use this key to either get information from or modify the user DCB.

When using this exit to change values in the DCB for a data set that has been allocated to an SMS-managed volume, do not specify values that would change the data set to a type which cannot be SMS managed, such as unmovable. Changing an SMS-managed data set to a type such as unmovable results in abnormal termination of the task.

Be sure you determine the type of DCB and device passed to the exit before testing access-method or device-dependent fields in the DCB. The sample exit shown in Appendix A, "Example of an OPEN Installation Exit Module" on page 229 gives an example of isolating a QSAM DCB being opened to a DASD or tape device.

The JFCB address supplied to the exit points to a copy of the JFCB that is in the OPEN work area. There may be other JFCBs associated with the OPEN if ISAM or concatenated partitioned data sets are being opened.

In the case of BDAM, ISAM, and concatenated partitioned data sets, the UCB, whose address is supplied to the exit, may not be the only UCB associated with the DCB being opened. The UCB should not be modified.

The TIOT address supplied is of a TIOT entry (TIOENTRY label in the IEFTIOT1 macro). In the cases of ISAM and concatenated partitioned data sets, other TIOT entries may be associated with the DCB being opened. If concatenation of unlike attributes is being processed, the TIOT entry may have a blank DDNAME field.

The format-1 DSCB passed to the exit is in the OPEN work area. The address is that of the field DS1FMTID. There may be format-2 and -3 DSCBs associated with the format-1 DSCB. There may be other format-1 through -3 DSCBs associated with the DCB being opened in the cases of ISAM, BDAM, and concatenated partitioned data sets. If the OPEN is to the VTOC, a format-4 DSCB address is passed to the exit; this can be determined by testing field DS1FMTID for a value of X'F4', or the data set name in the JFCBDSNM field of 44X'04'.

Defaulting the DCB Buffer Number

If a value has not yet been supplied, the exit may be used to supply an installation-determined value for DCBBUFNO (number of buffers) for QSAM DCBs.

A sample exit program that does this is shown in Appendix A, "Example of an OPEN Installation Exit Module" on page 229.

You should not override a nonzero value in DCBBUFNO for QSAM DCBs without knowing what dependency the user program has on that value. When a buffer pool control block address is in the DCB field DCBBUFCA, you cannot override DCBBUFNO; this indicates that buffers have been acquired before OPEN. If no buffer pool control block address exists, DCBBUFCA is set to one (not zero)

You should not override a zero value in DCBBUFNO for BSAM DCBs when DCBBUFCA is set to one without knowing what dependency the user program has on these values. If the user program does not want OPEN to acquire buffer storage space, it indicates this by setting DCBBUFNO to zero and DCBBUFCA to one. If the user program wants OPEN to acquire buffer storage space, it can override DCBBUFNO with a nonzero value. The user program is then responsible for freeing that space after closing the DCB.

Forcing the System to Determine Block Size

OPEN calls DASD Calculation Services to obtain a system-determined block size if the block size in the DCB (DCBBLKSI) is zero upon return from this exit. If you want the system to determine the block size for a data set, set DCBBLKSI to zero before returning from this exit.

Note: If a system-determined block size is used, OPEN turns on the reblockable indicator in the format-1 DSCB (bit 2 of the DS1SMSFG field). When the data set is opened, and the merging of the block size occurs, OPEN checks the indicator. If the block size is a system-determined block size, and the LRECL or RECFM have changed from those specified in the data set label, OPEN will rederive the block size.

Modifying the JFCB

Whenever the JFCB is modified, code 4 should be returned to OPEN. This causes OPEN to rewrite the JFCB. The JFCB should not be modified if the user program has set JFCNWRIT (bit 4) in byte JFCBTSDM because it indicates the JFCB should not be written.

A sample exit program that modifies the JFCB is shown in Appendix A, "Example of an OPEN Installation Exit Module" on page 229.

Requesting Partial Release

An example of modifying the JFCB in OPEN's work area to request partial release is shown in Appendix A, "Example of an OPEN Installation Exit Module" on page 229. It sets the following bits to 1, indicating a partial release request: JFCRLSE (bits 0 and 1; mask X'C0') in byte JFCBIND1. This should be done only for DASD physical-sequential or partitioned data sets opened for OUTPUT or OUTIN and processed by either (1) EXCP with a 5-word device-dependent section present in the DCB, (2) BSAM, or (3) QSAM.

Care should be taken in modifying the JFCB release bits. For example, a data set that is opened for output many times, writing varying amounts of data each time, may have to extend after each OPEN, resulting in many small extents and, perhaps, reaching the 16-extent limit. This could result in a B37 abend.

Care should also be taken in setting the JFCBSPAC bits to define the space quantity units when the partial release flag, JFCBRLSE, is also set on. A cylinder allocated extent may be released on a track boundary when JFCBSPAC does not indicate cylinder units or average block length units with ROUND specified. This causes the cylinder boundary extent to become a track boundary extent, thereby losing the performance advantage of cylinder boundary extents. Zeroing the release indicator and increasing secondary allocation quantity (for example, when the data set has extended a large number of times) may prevent such a B37 abend. Setting the release indicator could result in more space being made available to other users sharing the volume.

Updating the Secondary Space Data

The JFCB may also be modified by updating the secondary space data. Byte JFCBCTRI contains the space request type coded in the DD statement or merged from the format-1 DSCB. Field JFCBSQTY contains the amount of secondary space (in either tracks, cylinders, or average block units). Field JFCBPQTY contains the amount of primary space (in either tracks, cylinders, or average block units).

Setting the contiguous bit (JFCONTIG) to zero may prevent an out-of-space abend where there is enough space, but not enough contiguous space, to satisfy a request to extend the data set.

Registers at Entry to the DCB OPEN Exit

At entry to the exit, register contents are as follows:

Register	Contents
1	Address of the DCB OPEN installation exit parameter list
13	Address of an 18-word save area
14	Return address to OPEN
15	Address of the entry point to IFG0EX0B

Registers at Return from the DCB OPEN Exit

When you return to OPEN, register contents must be as follows:

Register	Contents
0-14	Same as on entry to the exit
15	A return code from IFG0EX0B

Return Codes From the DCB OPEN Exit

The DCB OPEN exit must pass a return code back to OPEN in register 15. The return codes and their meanings are as follows:

Code	Meaning
00(X'00')	Indicates that the JFCB has not been modified
04(X'04')	Indicates that the JFCB has been modified

Open/EOV Installation Exit for Format-1 DSCB Not Found

The function of the format-1 DSCB-not-found installation exit in OPEN and EOV is to determine whether a missing DSCB (such as a data set that has been migrated to another volume) can be restored to the volume. If your exit module restores the DSCB, it indicates this when it returns control to the control program. The exit module, IFG0EX0A, is given control whenever OPEN or EOV fails to find a format-1 DSCB on a volume. There is an IBM-supplied exit module, IFG0EX0A, in SYS1.LPALIB. If you want to use your own exit module, you must replace IFG0EX0A. Your exit module must have an entry point name of IFG0EX0A. If you do not write your own exit module, processing continues normally because the IBM-supplied exit returns a zero return code.

The exit is taken even under conditions under which abnormal termination ordinarily would not occur. Two examples of these conditions follow:

1. When you have specified DISP=MOD and error recovery processing is taking place because the last volume specified in the JFCB does not contain the DSCB, but an earlier volume does. For this case, if your return code from IFG0EX0A is 0 or if your return code is 4 and the DSCB has not been restored, OPEN and EOV search the other volumes for the DSCB after the exit is taken.
2. Another condition occurs during EOV output when space has not yet been allocated on the new volume. Space is allocated after the exit is taken if your return code from IFG0EX0A is 0 or if your return code is 4 and the DSCB has not been restored.

When a DSCB is not found, IFG0EX0A is given control as follows:

- In system protect key 5 (data management key).
- In supervisor state.
- The system resource represented by the SYSZTIOT major name is enqueued for shared control. (This ENQ prevents the exit from invoking system functions such as SCRATCH, RENAME, dynamic allocation, or LOCATE.)

Data That OPEN/EOV Passes to the Exit

The parameter list pointed to by register 1 consists of two fullwords. The first fullword contains the address of the UCB of the volume for which the format-1 DSCB was not found. The second fullword contains the address of the 44-byte data set name, left justified, and padded with blanks. Bit zero of the second fullword is set to one, indicating the last word in the parameter list.

The data set name must not be modified by the exit. The parameter list, save area, and data set name are in protect key 5 virtual storage, which is not fetch protected. IFG0EX0A must be reenterable. All work areas obtained through GETMAIN must be released through FREEMAIN.

Registers at Entry to the Format-1 DSCB Not Found Exit

At entry to your exit routine, register contents are as follows:

Register	Contents
0	If X'00', entry was from OPEN (single volume data set). If X'0C', entry was from OPEN (multivolume data set). If X'0F', entry was from EOV.
1	Address of the parameter list
2-12	Unpredictable
13	Address of an 18-word save area
14	Return address to OPEN/EOV
15	Address of entry point to IFG0EX0A

Registers at Return from the Format-1 DSCB Not Found Exit

When you return to OPEN/EOV, register contents must be as follows:

Register	Contents
2-12	Same as on entry to the exit
15	A return code from the exit

Return Codes from the Format-1 DSCB Not Found Exit

The format-1 DSCB not found exit must pass a return code back to OPEN/EOV as follows:

Code	Meaning
00(X'00')	Processing continues normally. This return code is given if the exit does not restore the DSCB. The IBM-supplied exit module always gives return code 0.
04(X'04')	The volume is searched one more time by OPEN or EOV for the DSCB. This return code is given if IFG0EX0A restores the DSCB to the volume. If the DSCB is again not found, IFG0EX0A is not given control and processing continues normally.
08(X'08')	The task is abnormally terminated without attempting to determine if DISP=MOD error recovery or allocation on the new volume should occur. This return code is given if IFG0EX0A encounters an error and you do not want to continue processing.

You should have IFG0EX0A establish its own error recovery environment (for example, through an ESTAE), intercept any indeterminate errors, and return to the control program with return code 8. Problem determination is the responsibility of your exit module. A write-to-programmer (WTO with routing code 11) or a TPUT (if a TSO region) may be used to issue an informative message.

During a parallel OPEN when two or more DCBs are being opened at the same time and two of the DCBs are opening the same data set, the DSCB may be missing. If IFG0EX0A is called for the first of the two DCBs and restores the DSCB, the channel program attempting to read the DSCB for the second DCB may have been executed before the restoration of the DSCB was complete. IFG0EX0A is then called for the second DCB, even though the DSCB has already been restored. Return from IFG0EX0A with a return code 4 is appropriate in this case.

IFG0EX0A is not given control when you are processing a VSAM data set with an ACB; however, it is given control when you are processing a VSAM data space with a DCB. IFG0EX0A is bypassed if the format-4 DSCB is not found on a volume, even if the OPEN is to the VTOC data set name (data set name of 44 bytes of X'04').

IDATMSTP Datestamp Routine

The datestamp control module (IDATMSTP) is provided as a module that you can use to cause datestamp processing to be skipped. It sets a return code of 4 that causes VSAM to process the last-referenced date (DS1REFD) in the format-1 DSCB for VSAM data sets cataloged under the integrated catalog facility.

The datestamp control module cannot be used to change the date last referenced in the DSCB. The DSCB can be modified using the DCB OPEN

installation exit, IFG0EX0B. It is described in "DCB OPEN Installation Exit (IFG0EX0B)" on page 159.

Register Contents at Entry to IDATMSTP

Figure 82 shows the contents of the registers when VSAM gives control to IDATMSTP.

Register	Contents
0	Unpredictable.
1	Address of a parameter list that contains the following addresses of the offset: 0 Data set name. 4 List of five volume serial numbers that contain the data set. 8 Address of a 1-byte indicator set to X'D' to indicate the data set is a base data component.
2-12	Unpredictable.
13	Save area address.
14	Return address in VSAM OPEN.
15	Entry address to IDATMSTP.

When IDATMSTP returns to VSAM OPEN, the desired datestamp option is indicated in register 15:

- 0 No datestamp processing.
- 4 Datestamp processing desired; updated indicator (DS1IND02) maintained.

Figure 82. Communication with the Datestamp Routine

Programming Considerations

IDATMSTP is packaged as a single load module.

It is entered in 31-bit addressing mode and must return in 31-bit addressing mode. IDATMSTP may be replaced with another module you select that sets a return code of 0 and causes datestamp processing to be skipped on all specified data sets.

Products or installations which replace the VSAM version of IDATMSTP with their own module must link-edit their version into LPALIB as a separate load module named IDATMSTP.

Your module can include code to cause checking of some or all the base cluster data components of VSAM data sets cataloged in an integrated catalog facility catalog for periodic migration to other storage media and maintenance of the updated indicator.

Parameters Passed to IDATMSTP

Parameters passed to the IDATMSTP module may reside above 16 megabytes.

IDATMSTP is passed the addresses of the data set name, first 5 volume serial numbers, and a character 'D' which indicates that the object is a data component of a data set cataloged in an integrated catalog facility catalog. This information is available to your module and can be used to further qualify which data sets should have datestamp processing.

Requesting Datestamp Processing

If you don't modify the IDATMSTP module to skip datestamp checking (return code 0 in register 15), VSAM datestamp processing is performed if the following conditions are met:

- The data set is the data component of a base cluster or alternate index.
- The field DS1REFD in the format-1 DSCB is earlier than today's date.

Returning to VSAM

When IDATMSTP returns to VSAM OPEN, your routine must indicate the datestamp option in register 15 (see Figure 82 on page 168).

IGXMSGEX Message Display Exit

The message display installation exit (IGXMSGEX) is provided as a module that you can use to customize messages for display on an IBM 3480 tape drive. The exit is optional, and is not invoked if it does not exist. You can use the exit to modify either the parameter list (EXITPLST, shown in Figure 84 on page 170) or the message text pointed to by the parameter list, based on jobname, step name, or some other means. You can also use the exit to specify no automatic cartridge load.

Register Contents at Entry to IGXMSGEX

Figure 83 shows the contents of the registers when the message display execution module gives control to IGXMSGEX.

Register	Contents
0	Unpredictable.
1	Address of a two word parameter list (EXITPLST)
2-12	Unpredictable.
13	Save area
14	Return address
15	Base register.

Figure 83. Registers on Entry to IGXMSGEX

Parameters Passed to IGXMSGEX

Figure 84 shows the contents of the parameter list when the message display execution module gives control to IGXMSGEX.

Field	Description
EXITPLST	CHAR(8) Eye-catcher
UCBADDR	PTR(31) • CHAR(1) which is reserved • PTR(24) address of the UCB
MSGTXADR	PTR(31) Address of MSGTEXT

Figure 84. IGXMSGEX Parameter List

MSGTEXT, which is pointed to by MSGTXADR, consists of a format control byte followed by two 8-byte display data fields. You can modify the messages in these display data fields to meet your requirements. You can modify bit 7 in the format control byte to request no automatic cartridge load. Changes to any other bits in the format control byte are ignored. The format control byte is described below in Figure 85:

Bit	Contents
0-2	New message overlay
3	Alternate messages
4	Blink message
5	Display Low/High message
6	Reserved
7	Automatic cartridge load (ACL) request

Figure 85. MSGTEXT Format Control Byte

Programming Considerations

If you want to use the exit, IGXMSGEX must be link-edited with the message display execution module (IGX00030). If it is not link-edited with IGX00030, the exit will not be called.

If IGXMSGEX is link-edited with IGX00030, then IGXMSGEX is called for MSGDISP requests of MOUNT, DEMOUNT, VERIFY, and GEN.

The PSA points to the ASCB, which points to the CSCB. The CSCB contains the jobname and the stepname.

On entry to the exit, IGXMSGEX is given control as follows:

- In system protect key 5 (data management key)
- In supervisor state

- AMODE 31
- RMODE any

The exit routine you code for IGXMSGEX must be re-entrant.

On return from the exit, the message display execution module (IGX00030) uses any modifications of the two display data fields and bit 7 in MSGTEXT when issuing the load display channel command.

Controlling the Automatic Cartridge Load

If you want to request that no automatic cartridge load be performed, you can set bit 7 in the format control byte to zero. On return from the exit, the message display execution module unconditionally accepts the new zero value of bit 7. If you set bit 7 to a value of one, the message display execution module verifies that the device supports automatic cartridge load and is active. If the device does not support automatic cartridge load, and your exit set bit 7 on, then the message display execution module resets bit 7 to zero before issuing the load display channel command.

If the exit is not called (IGXMSGEX is not link-edited with IGX00030), MSGDISP enables automatic cartridge load only when:

- the device supports automatic cartridge load,
- the automatic cartridge load is ready,
- and the VOLSER is either SCRTCH or PRIVATE.

Chapter 10. Tape Label Processing Installation Exit Routines

General Guidance

This chapter discusses installation-written replaceable modules for specialized tape processing. With replaceable modules you can:

- Create and process nonstandard tape labels.
- Edit labels when versions, label types, density, or volume serial number conflicts are detected.
- Control volume access, file access and label validation for ISO/ANSI/FIPS Version 3 volumes.
- Selectively convert non-Version 3 volumes to Version 3 volumes.

The replaceable modules available for tape label processing are listed in Figure 86.

Figure 86 (Page 1 of 2). Tape Label Processing Modules

Module Name	Description	When Available
NSLOHDRI NSLEHDRI	Nonstandard label processing routines for input headers	At open/EOV
NSLOHDRO NSLEHDRO	Nonstandard label processing routines for output headers	At open/EOV
NSLETRLI	Nonstandard label processing routine for input trailers	At open
NSLETRLO NSLCTRLO	Nonstandard label processing routines for output trailers	At EOV/close
NSLRHDRI	Nonstandard label processing routine for restarting after a checkpoint	At restart from a checkpoint
IEFXVNSL	Automatic volume recognition (AVR) nonstandard label processing	When AVR cannot identify the first record on a magnetic tape volume as a standard label
NSLREPOS	Volume verification using the dynamic device reconfiguration (DDR) option for nonstandard label processing	When DDR is used for nonstandard labels

Figure 86 (Page 2 of 2). Tape Label Processing Modules		
Module Name	**Description**	**When Available**
OMODVOL1 EMODVOL1	Volume label editor routines for open and EOV	At open/EOV
IFG0193G	ISO/ANSI/FIPS Version 3 label installation exits for volume access, file access, label validation, and label validation suppression	At open/EOV; file access: after positioning to a requested data set
IEECVXIT	ISO/ANSI/FIPS Version 3 label WTOR installation exit	Label version conflict

Programming Considerations

In general, your replaceable module must:

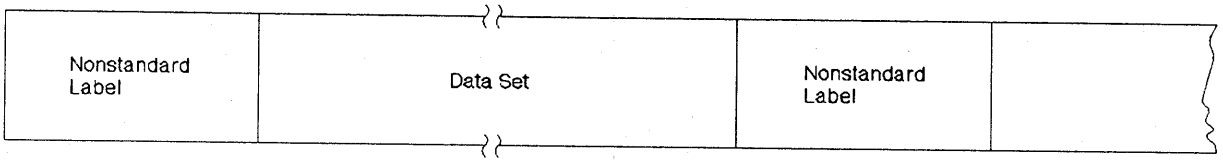
- Follow the naming conventions for the particular module you are replacing
- Save and restore registers
- Reside in SYS1.LPALIB.

Nonstandard Labels

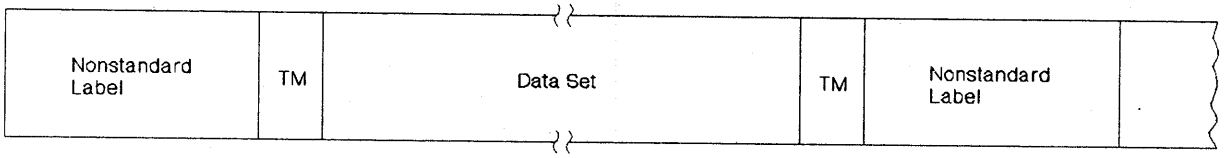
Nonstandard labels do not conform to the IBM or ISO/ANSI/FIPS standard label formats. They are labels which you design, and you provide routines to write and process them. There are no requirements as to the length, format, contents, and number of nonstandard labels, except that the first record on a BCD, EBCDIC, or ISCII/ASCII tape cannot be a standard volume label.

Figure 87 on page 175 shows some examples of how you can organize tape volumes with nonstandard labels. Other variations are possible. Because your routines do the positioning, there are no special requirements for multivolume or multiple data set organizations. All labels and tapemarks are written by your routines. If an operating system access method is used to retrieve the data, tapemarks should precede and follow the data set to indicate the end-of-data-set condition for forward and backward read operations.

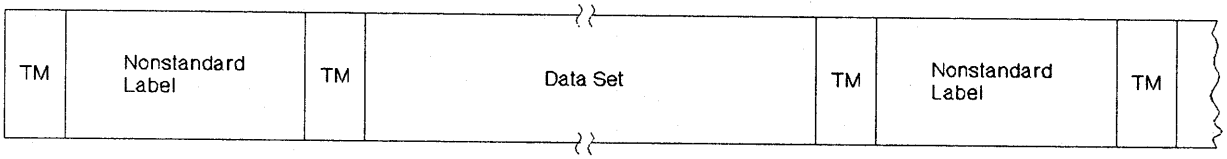
Example 1 -- No Tapemarks



Example 2 -- Tapemarks Delimiting the Data Set



Example 3 -- Tapemarks Delimiting the Labels and the Data Set



1. No Tapemarks: This type of organization can be created by your nonstandard label processing routines, and read with the EXCP technique. It should not be used with an operating system access method because there is no tapemark to signal end-of-data.
2. Tapemarks Delimiting the Data Set: This is the recommended organization. The tapemarks are written by your nonstandard label processing routines. When the tape is read by an operating system access method, the tapemark following the data set signals end-of-data for forward read operations, and the tapemark preceding the data set signals end-of-data for backward read operations.
3. Tapemarks Delimiting the Labels and the Data Set: This is an expansion of the preceding organization. The additional tapemarks that precede and follow the labels are not handled by the operating system. They are written and used by your nonstandard label processing routines.

Figure 87. Examples of Tape Organizations with Nonstandard Labels

If you want to use nonstandard labels for tape volumes you must:

1. Create nonstandard label processing routines for input header trailer labels, input trailer labels, output header labels, and output trailer labels.
2. Insert your routines into the link pack area (LPA) library (SYS1.LPALIB).
3. Code NSL in the LABEL parameter of the DD statement at execution time.

This section explains how your nonstandard label processing routines work with the operating system control program, how to write your routines, and how to insert your routines into the operating system.

Processing Tapes with Nonstandard Labels

Your appropriate nonstandard label processing routine is selected, brought into virtual storage, and executed when a data set is opened or closed, when an end-of-volume or end-of-data-set condition occurs, or for repositioning a volume when a job step is restarted from a checkpoint. When your routine has completed its processing, it must return control to data management's open, close, EOV, or restart routine, which then continues its normal processing. For input, the EOV routine handles both end-of-volume and end-of-data-set conditions. For output, the EOV routine handles the end-of-volume condition, and the close routine handles the end-of-data-set condition.

Your routines must provide for reading labels, processing labels, writing labels, writing tapemarks, identifying volumes, and positioning volumes. Your

nonstandard label processing routines are responsible for setting the UCB file sequence (UCBFSEQ) and UCB file count (UCBFSCT) fields, based upon the user's processing. The control program assumes that a tape with nonstandard labels is properly positioned upon completion of a nonstandard label processing routine.

If you want the control program to maintain a block count, your header label routines that receive control from open or EOV must properly initialize the block count field of the DCB. During EOV processing in BSAM and QSAM, the DCBIOBA field of the DCB points to an IOB. The DCBBLKCT field must be decreased by the value in the IOBINCAM field of that IOB. If chained scheduling is being used, the block count in the DCB is correct and need not be decreased.

When processing is completed, the control program handles volume disposition in accordance with the parameters of the DD statement. Your nonstandard label processing routines are responsible for any positioning specified by the OPEN or CLOSE macro instructions. If you need to process a data set more than once in a job, or if you want to handle multiple data set volumes, your routines must control the positioning. If you handle volume disposition in your nonstandard label processing routines, you must issue volume disposition messages to the console operator. Data management checks to see if your routine has handled disposition, and it bypasses disposition and message handling if volume disposition is verified. Be extremely careful when verifying a tape under one processing technique and then accessing the tape under a second technique (for example, changing from NSL to NL with a verified tape).

Following paragraphs explain the flow of control between the control program and each type of nonstandard label processing routine. Information on tape positioning and volume identification is also provided.

Support for RACF protection of tape volumes may be a part of nonstandard label processing routines.

Input Header Label Routines

When nonstandard labels are specified, the control program checks the input tape to make sure that the first record is not a standard volume label. If the first record contains the identifier VOL1 in the first 4 bytes, is recorded in EBCDIC, and is 80 bytes long, or it is recorded in ISCII/ASCII and is 80 bytes or more in length, the tape is rejected by a message from the control program directing the operator to demount the current volume and mount the correct volume. The various error conditions that can occur during verification of the first record are explained under "Volume Label Verification and Volume Label Editor Routines" on page 196.

When it is determined that the tape does not contain a standard volume label, the open or EOV routine gives control to your routine for processing input header labels. Control comes from the open routine for the first or only volume of a data set, or for a concatenated data set with unlike characteristics. (Data sets with like characteristics can be processed correctly using the same DCB, IOB, and channel program. Any exception in processing makes the data sets unlike.) Control comes from the EOV routine for the second and subsequent volumes of a data set, or for a concatenated data set with like characteristics.

When your routine receives control, the tape has been rewound; the tape is positioned at the interrecord gap preceding the nonstandard label.

Note: If the control program finds that the tape volume has been previously verified in the job, the tape has not been rewound.

If your routine determines that the wrong volume is mounted, you must place a 1 in the high-order bit position of the UCBDMCT field of the UCB, and return control to the control program. The control program then issues a message directing the operator to mount the correct volume. When the volume is mounted, the control program again checks the initial label on the tape before giving control to your routine.

Before returning control to the control program, your input header label routine must position the tape at the appropriate data set:

- For forward read operations, position the tape at the interrecord gap that precedes the initial record of the data set.
- For backward read operations, position the tape after the last record of the data set.

Input Trailer Label Routines

When a tapemark is encountered on an input tape, data management's EOV routine gives control to your routine for processing input trailer labels, with two exceptions. The EOV routine does not give control to your input trailer label routine when:

- The FEOV macro instruction is used to force an end-of-volume condition.
- At the end of the data set, the DCB exit list indicates deferred nonstandard input trailer label processing.

When your routine receives control, the tape is already positioned for label processing:

- For forward read operations, the tape has been positioned immediately after the tapemark at the end of the data set.
- For backward read operations, the tape has been positioned immediately before the tapemark at the beginning of the data set.

Your routine need not reposition the tape before returning control to the control program.

If additional volumes are specified in the JFCB the control program uses the next-specified volume serial number and performs volume switching. (You specify the volume serial numbers in forward sequence, regardless of whether the tapes are to be read forward or backward.) If the new volume is not already mounted, the control program issues a mount message to the operator. The new volume is then processed by the EOV routine and your input header label processing routine.

If another volume is not specified in the JFCB, the control program gives control to your end-of-data-set (EODAD) routine that is specified in the DCB. Subsequently, the processing program or the operating system closes the data set. When an input data set is closed, your output trailer label routine is given control. This allows you to position the tape if necessary. When an

end-of-data-set condition occurs and the DCB exit list (EXLST) indicates deferred nonstandard input trailer label processing, the close routine passes control to your input trailer label routine before passing control to your output trailer label routine. The DCB exit list is described in "EXLST Exit List" on page 58.

Output Header Label Routines

When nonstandard labels are specified for output, the control program checks the tape to make sure that the existing first record is not a standard volume label. If the first record is 80 bytes in length and contains the identifier VOL1 in the first 4 bytes, the tape is not accepted, as is. If an IBM standard label exists, it is overwritten with an IBM tapemark, if possible. If an ISO/ANSI/FIPS standard label exists, the console operator must confirm that it can be destroyed. The various error conditions that can occur during verification of the first record are explained under "Volume Label Verification and Volume Label Editor Routines" on page 196.

When the control program ensures that the first record on the tape is not a standard volume label, the open or EOV routine gives control to your routine for processing output header labels. Control comes from the open routine for the first or only volume of a data set. Control comes from the EOV routine for the second and subsequent volumes of a data set. When your routine receives control, the tape has been positioned at the interrecord gap preceding the nonstandard label (the tape has been rewound). However, the tape has not been rewound if the control program found that this volume has been previously verified during the job.

If your routine determines that the wrong volume is mounted, you must place a 1 in the high-order bit position of the UCBDMCT field of the UCB and return control to the control program. The control program then issues a message directing the operator to mount the correct volume. When the new volume is mounted, the control program again checks the initial label on the tape before giving control to your routine.

The volume serial number in the UCBVOLI field of the UCB and the volume serial number in the JFCB must be the same as on entry unless a request is being made for a nonspecific volume. The control program recognizes a nonspecific request by the volume serial number requested in the JFCB being blank or SCRTCH. In this case, UCBVOLI will be set to Lxxxxx, where xxxxx is a 5-digit decimal number. This volume serial number is generated by the control program. It may be replaced in your NSL routine by the volume serial number present in the volume label read from the tape, or the volume serial number of the volume label written on the tape.

When control is returned to the control program from NSLOHDRO or NSLEHDRO for a nonspecific request (as defined in the preceding paragraph) and the UCBVOLI field of the UCB has been modified, the control program will ENQ on the volume serial number to effect volume integrity and will place the volume serial number in the JFCB or JFCB extension. If some other technique than that just described is used to support nonspecific requests, the NSL routine must update the JFCB and ENQ on the volume serial number (system ENQ; major name; SYSZVOLS; minor name: 6-byte volume serial number;

exclusive ENQ). If the result of the control program's ENQ is that the resource is unavailable (either the current task previously obtained the resource or some other task holds the resource), the volume will be rejected.

Your routine need not reposition the tape before returning control to the control program.

When tapes are first received at your installation, they should be initialized with a tapemark or other record. If a blank tape is mounted for an output data set, it is read through and removed from its reel when the control program looks for an existing standard volume label.

Restart Label Processing Routine

If you restart at checkpoints and use tapes with nonstandard labels, you must provide a routine to process nonstandard labels at restart time. You need only a routine to check existing header labels. You do not need separate routines for input and output, because output tapes will contain the header labels that were written when the data sets were opened (before checkpoint).

At restart time, the control program checks the tape to make sure that the first record is not a standard volume label. If the first record is 80 bytes long and contains the identifier VOL1 in the first 4 bytes, the tape is rejected by a message from the control program directing the operator to mount the correct tape.

When it is determined that the tape does not contain a standard volume label, the control program's restart routine gives control to your routine for processing nonstandard labels. When your routine receives control, the tape has been positioned at the interrecord gap preceding the nonstandard label (the tape has been rewound).

If your routine determines that the wrong volume is mounted, you must place a 1 in the high-order bit position of the UCBDMCT field of the UCB, and return control to the control program. The control program then issues a message directing the operator to mount the correct volume. When the new volume is mounted, the control program again checks the initial label on the tape before giving control to your routine.

Before returning control to the control program, your routine must position the tape at the interrecord gap that precedes the initial record of the appropriate data set. This applies to both forward and backward read operations. The control program then uses the block count shown in the DCB to reposition the tape at the appropriate record within the data set. This positioning is always performed in a forward direction. If the block count is zero or a negative number, the control program does no positioning. (If you want the control program to reposition the tape, your normal header label routines—open and EOV—must properly initialize the block count field of the DCB. The block count field of the DCB must not be altered at restart time.)

Output Trailer Label Routines

Your routine for writing output trailer labels receives control from data management's EOV or close routines. The EOV routine handles end-of-volume conditions (reflective strip or FEOV macro instruction). The close routine handles end-of-data-set conditions (CLOSE macro instruction). When your routine receives control, the tape has been positioned at the interrecord gap following the last data set record that was written.

Your routine need not reposition the tape before returning control to the control program.

Your output trailer label routine is also given control when input data sets are closed. This allows you to position the tapes if necessary.

Data Recovery

Recovery routines are given control when an error occurs during open, close, and end-of-volume processing. One of the purposes of these routines is to provide data recoverability in case of an error that results in abnormal termination of your task. Data recoverability is provided in conjunction with your output trailer label routines by writing a tapemark after the last data written to the tape. The tapemark serves to indicate the end of the output data set, so that you can save the records written before the error occurred. The tapemark will only be written if an unrecoverable error occurs before your output trailer label routines have received control. If the error occurs during or after the execution of your trailer label routines, no tape mark will be written.

Writing Nonstandard Label Processing Routines

The following paragraphs describe conventions, requirements, and techniques for writing your nonstandard label processing routines.

Programming Conventions

The programming conventions to be observed when writing your routines are:

- *Size of the routine*: Nonstandard label processing routines are not limited in size.
- *Design of the routine*: Nonstandard label processing routines must be read-only. You cannot store into the routine, nor can you use macro instructions that store into the routine.
- *Register usage*: When your routine receives control, it must save the contents of registers 2 through 14 (in your own work area). Before returning control, your routine must restore the contents of these registers.
- *Entry point of the routine*: The entry point of the routine must be the first byte of the load module and must be on a doubleword boundary.
- *Exit from the routine*: You must use the XCTL macro instruction (E-form) to exit from your routine and return control to a specific control program module. These modules differ depending on the control program routine from which control was received and the type of label processing being performed. Module names are shown below for each control program routine and for each type of label processing routine.

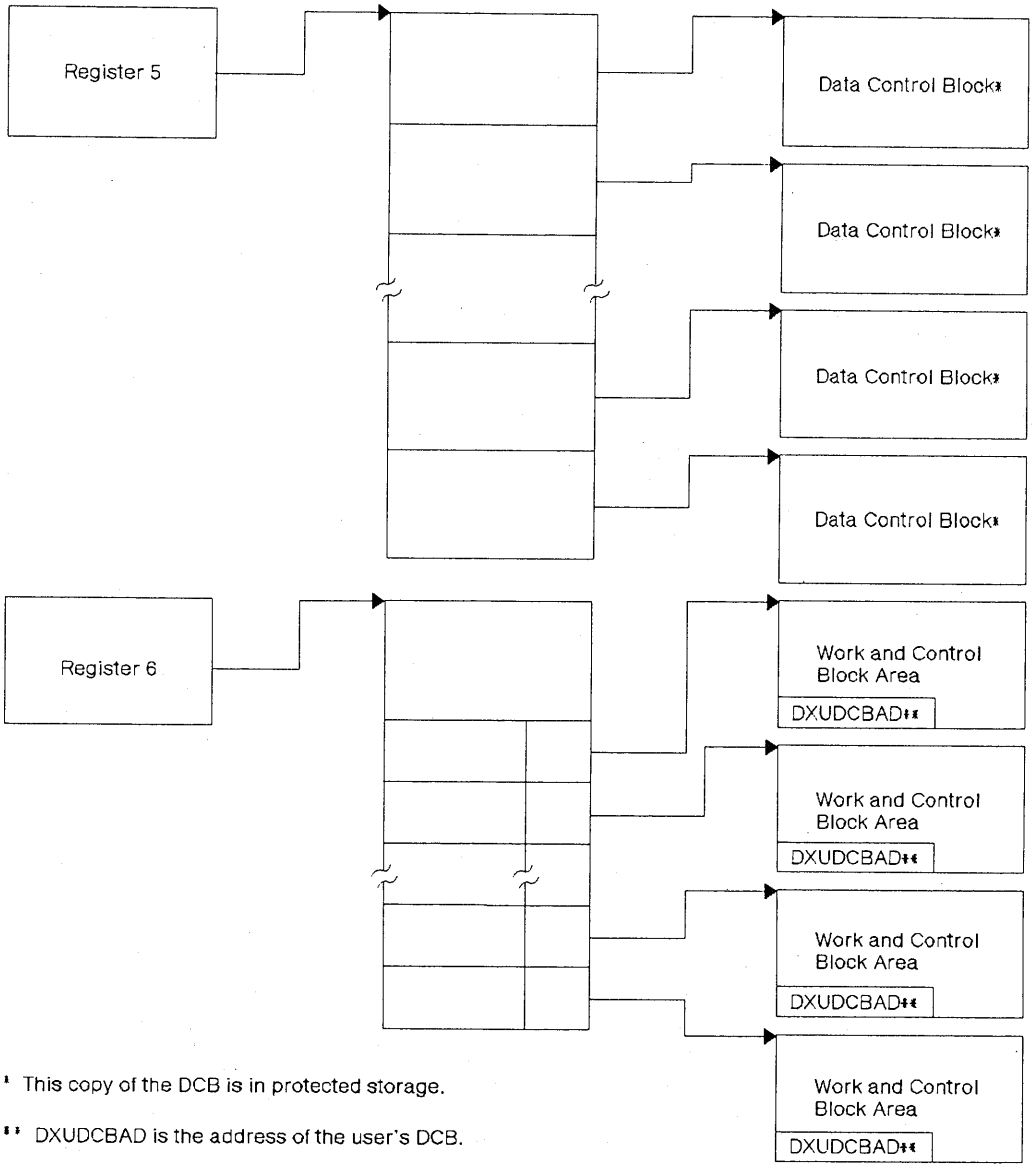
Label Processing Routine	Control Program Routine	Control Program Module Name
Input Header	Open	IGG0190B
	EOV	IGG0550D
Input Trailer	EOV	IGG0550B
Output Header	Open	IGG0190R
	EOV	IGG0550H
Output Trailer	EOV	IGG0550F
	Close	IGG0200B
Restart Header	Restart	IGC0K05B

- *Work areas*: You must use the GETMAIN macro instruction to obtain virtual storage for all your work areas, including areas used to read in or create a label. You must use the FREEMAIN macro instruction to release this virtual storage.

Program Functions

In processing nonstandard labels, you must perform many of the functions that the control program performs in processing standard labels. All input/output operations, such as reading labels, writing labels, and positioning volumes, must be performed by using the EXCP (execute channel program) macro instruction. The use of EXCP normally requires that you build several control blocks in your work area. However, you can save coding effort and virtual storage space by using control blocks already established by the control program.

- When your routine receives control from the open or close routine, the status of control information and pointers is as shown in Figure 88 on page 182.
- When your routine receives control from the EOV routine, register 2 contains the address of a DCB, and register 4 contains the address of a combined work and control block area. The format of this area is shown in Figure 89 on page 183.
- When your routine receives control from the restart routine, register 9 contains the address of a restart table entry. The table entry contains the address of a control block area. This status is as shown in Figure 90 on page 184.
- The nonstandard label routines receive control in protect key zero.
- The DCB is copied into protected storage during open/close/EOV processing. During open and close processing, register 5 points to a parameter list that contains the address of the DCB in protected storage. During EOV processing, register 2 points to the DCB in protected storage. The address of the user's DCB is in the combined work and control block area at the label DXUDCBAD. If you want to change the DCB, both copies, the user's DCB and the DCB in protected storage, must receive the same change.



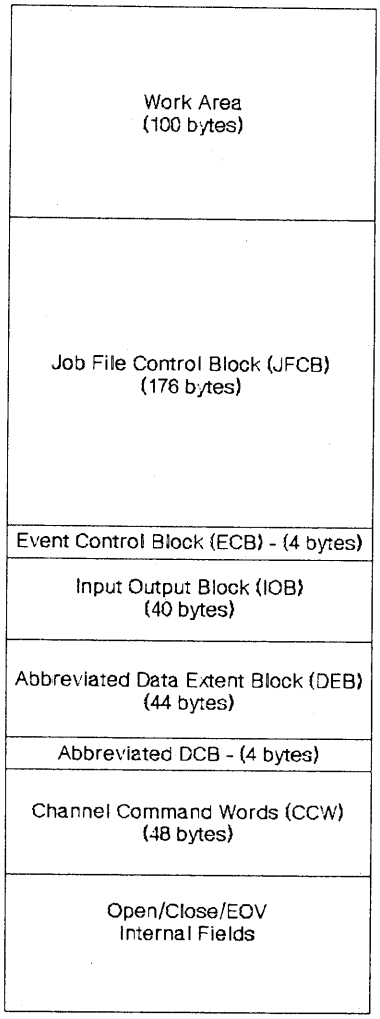
* This copy of the DCB is in protected storage.

** DXUDCBAD is the address of the user's DCB.

Register 5 contains the starting address of a list of DCB addresses. Each DCB specified in the OPEN or CLOSE macro instruction has a 4-byte entry in the list. The DCBs to which the entries point are in the problem program. The list may also include one or more ACB addresses. The list, the DCBs, and any ACBs will reside below the 16 M line.

For each DCB specified in the OPEN or CLOSE macro instruction, a combined work and control block area is built. Register 6 contains the starting address of a table that contains an address for each work and control block area. The addresses of the areas are contained in the low-order 3 bytes of 8-byte entries. The list of 8-byte entries begins 32 bytes from the starting address of the table. The format of the combined work and control block area is shown in Figure 89 on page 183.

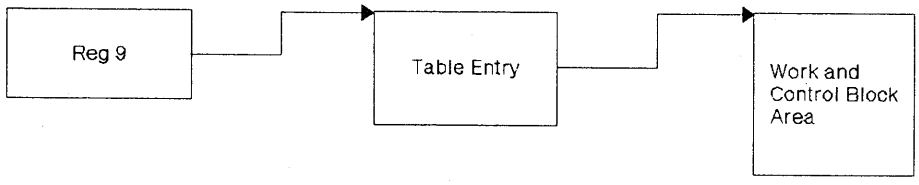
Figure 88. Status of Control Information and Pointers



Each of the fields within the work and control block area can be addressed by your nonstandard label processing routines. The IECDSECT macro instruction defines the symbolic names of all these fields. Code this macro instruction (with a null operand field and immediately preceded by a DSECT statement) in the list of constants for each of your nonstandard label processing routines. Using the starting address of the work area as a base, you are able to address any field symbolically.

When your nonstandard label processing routine receives control from the close or EOV routine, some of the information shown in the work area DEB is not the same as contained in the actual DEB. If you need actual DEB information at these times, you can get the address of the DEB from the DCBDEBAD field in the DCB.

Figure 89. Format of Combined Work and Control Block Area



Register 9 contains the starting address of a 48-byte table entry. Five bytes from the starting address of this table entry, a 3-byte field (TABSEGAD) contains the starting address of a work and control block area that is associated with the data set.

Figure 90. Status of Control Information and Pointers from the Control Program's Restart Routine

Mapping the Common O/C/EOV Workarea

The IECDSECT macro maps most of the main work area that is used by OPEN, CLOSE, EOV, access methods, and related components of the system. The general format of this area is described in Figure 89 on page 183.

The IECDSECT macro does not generate a DSECT statement. You should code a DSECT statement before calling IECDSECT. The format of the macro instruction is as follows:

[*symbol*]	**IECDSECT**	[**IOB**={**NO**|**YES**}]

IOB={**NO**|**YES**}

NO
specifies that you do not want symbols for a 32-byte basic IOB generated in the work area. **NO** is recommended.

YES
specifies that you want symbols for a 32-byte basic IOB generated in the work area.

Flowcharts for Sample Routines

General flowcharts of nonstandard label processing routines are shown in Figure 91 on page 185, Figure 92 on page 186, Figure 93 on page 187, and Figure 94 on page 191. These flowcharts suggest the logic that you could use in your routines. The logic is shown separately for routines receiving control from the open, close, EOV, or restart routines of the control program. Each block in the flowcharts is numbered, and the number corresponds to an item in the list of explanations that follows.

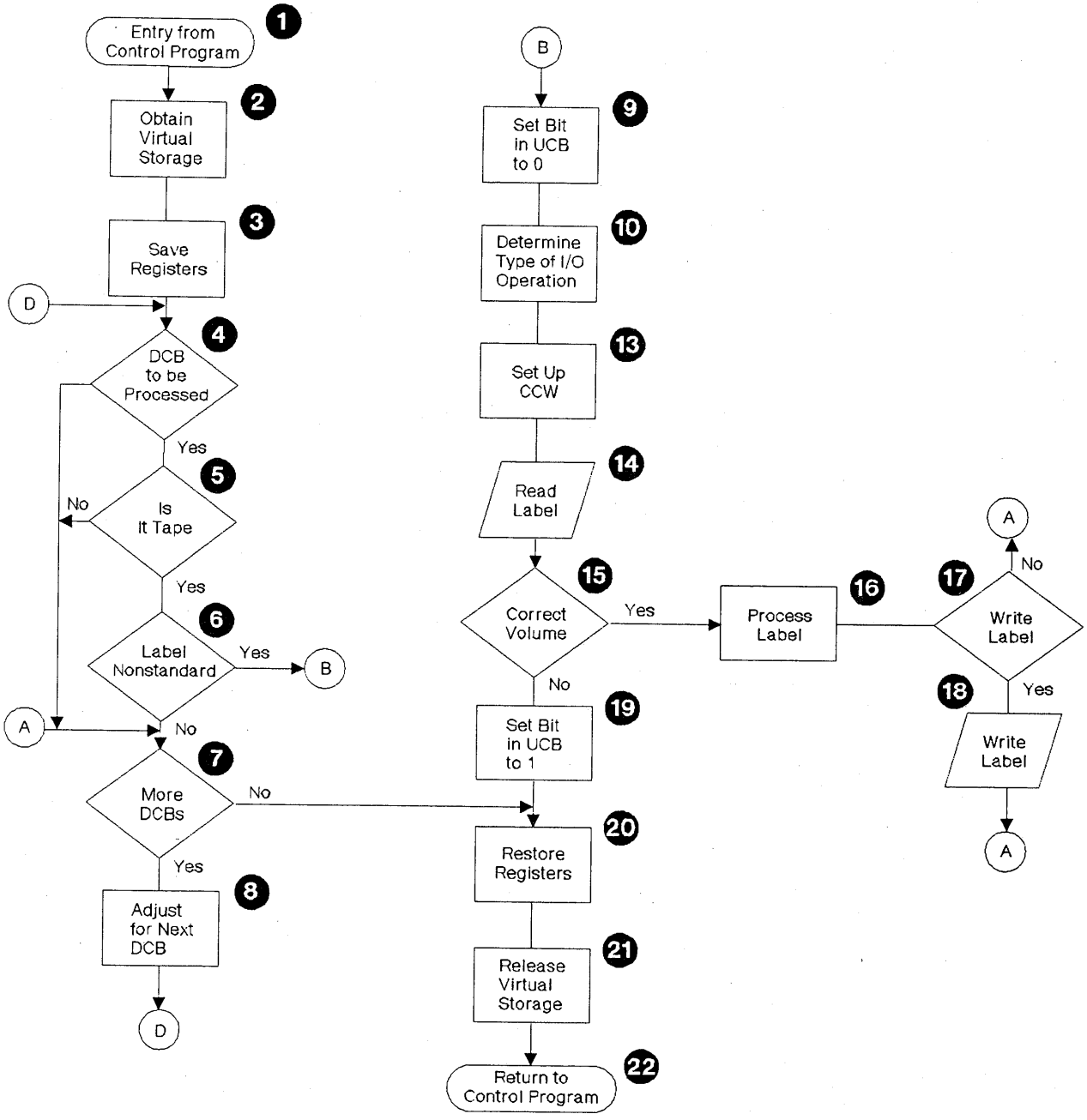


Figure 91. General Flow of a Nonstandard Label Processing Routine After Receiving Control from the Open Routine

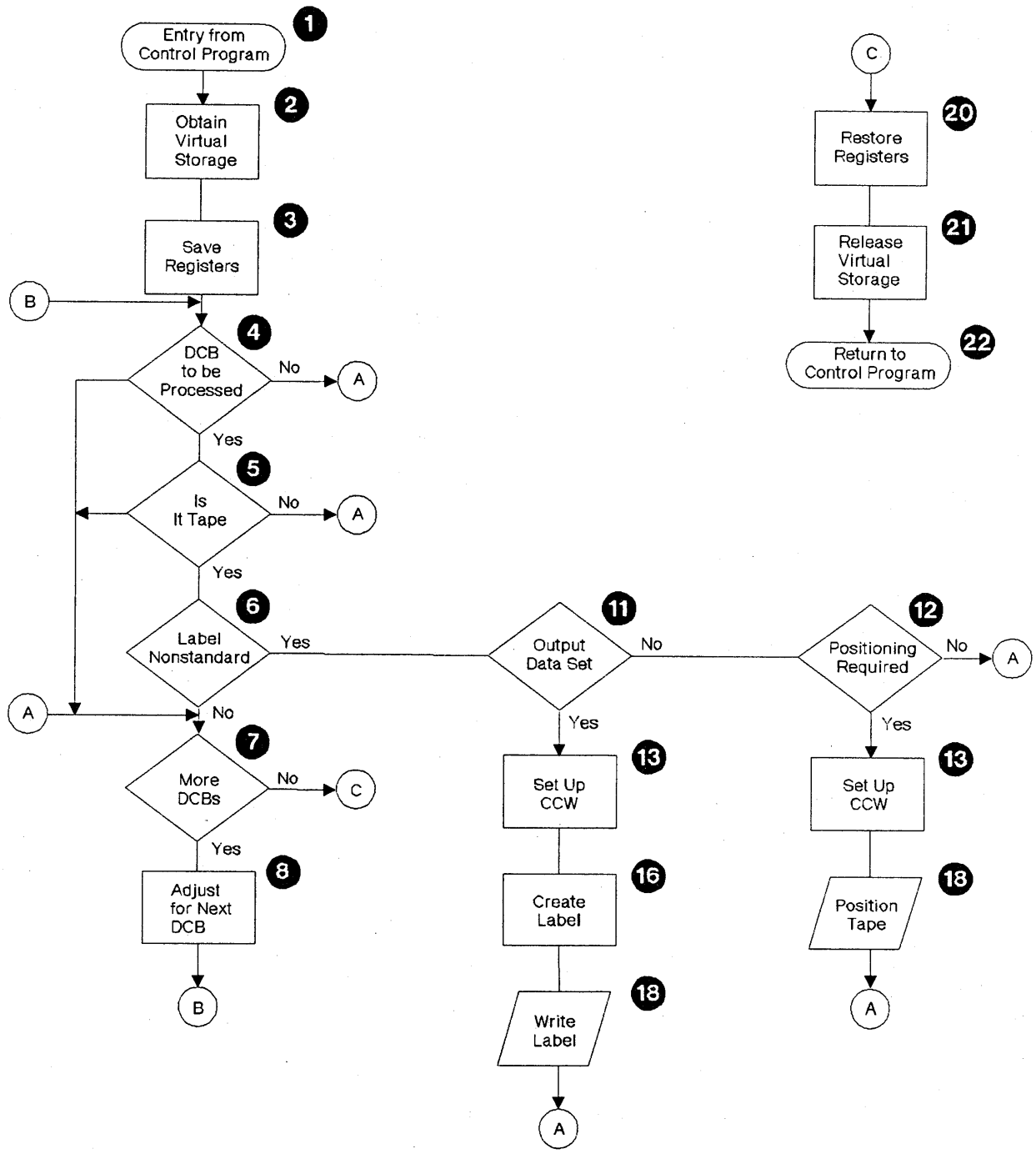


Figure 92. General Flow of a Nonstandard Label Processing Routine After Receiving Control from the Close Routine

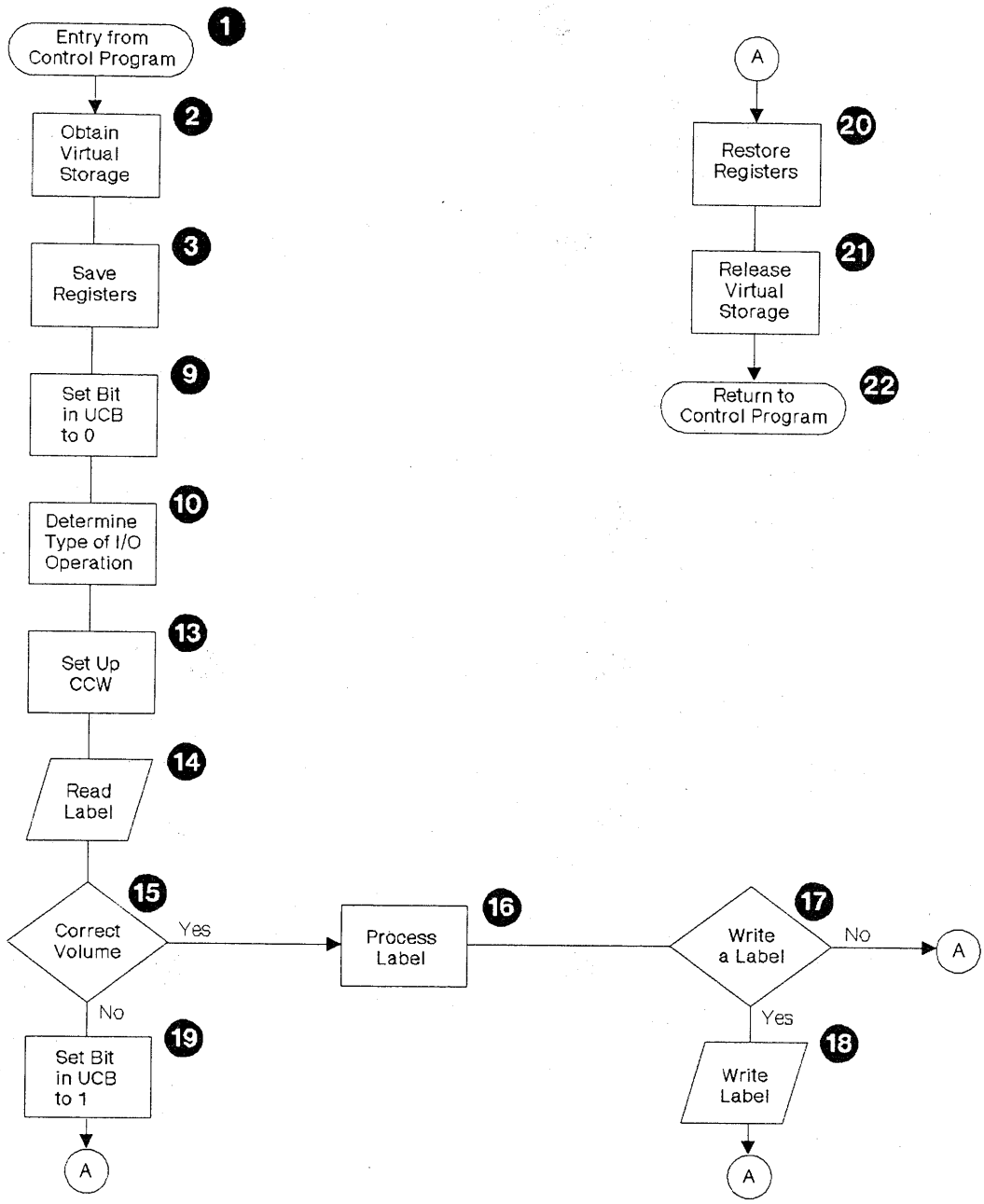


Figure 93. General Flow of a Nonstandard Label Processing Routine After Receiving Control from the EOV Routine

Explanation of Logic Blocks—Figures 91, 92, and 93

1 The entry is in the form of an XCTL macro instruction issued by the control program.

2 Use the GETMAIN macro instruction to obtain virtual storage.

3 Use the store multiple (STM) instruction.

4 To locate the address of the DCB, use the contents of register 5. To determine if the DCB is to be processed, test bits 6 and 7 of the DCBOFLGS field of the DCB; if these bits are 1, the DCB is to be processed. (The symbolic names of all fields in the DCB are defined by the DCBD macro instruction.) The user's DCB is pointed to by the DXUDCBAD field on the combined work and control block area.

   If a DCB in the CLOSE parameter list is not open at entry to CLOSE, it will not be processed, and its entry in the where-to-go table will be all zeros.

5 To determine if a tape data set is being processed, test the UCB3TAPE field of the UCB; this bit is 1 for a tape data set. The symbolic names of all fields in the UCB are defined by the IEFUCBOB macro instruction. The address of the UCB is contained in the DXDEBUCB field of the DEB as defined by the IECDSECT macro instruction. (The IEFUCBOB macro definition and how to add it to the macro library is described in *System—Data Administration*. The IECDSECT macro is described in "Mapping the Common O/C/EOV Workarea" on page 184.)

6 To determine if nonstandard labels have been specified, test the JFCBLTYP field of the JFCB; this field contains a hexadecimal 04 when nonstandard labels have been specified.

7 The final DCB entry in the list of DCB addresses contains a 1 in its high-order bit position.

8 Add 4 to the contents of register 5; add 8 to the contents of register 6.

9 Set the high-order bit to 0 in the UCBDMCT field of the UCB.

10 To determine the type of I/O operation specified in the OPEN macro instruction, check the bit configuration of the high-order byte of the DCB entry in the list of DCB addresses. The bit configuration for each type of I/O operation is shown below. (The high-order 4 bits correspond to the disposition of the data set; the low-order 4 bits correspond to the I/O operation itself. For example, the bit configuration x0110000 indicates a data set opened for input whose disposition is LEAVE.)

0	1	2	3	4	5	6	7	Bits
x	0	0	1	x	x	x	x	REREAD
x	0	1	1	x	x	x	x	LEAVE
x	0	0	0	x	x	x	x	Neither REREAD nor LEAVE
x	x	x	x	0	0	0	0	INPUT
x	x	x	x	1	1	1	1	OUTPUT
x	x	x	x	1	1	1	0	EXTEND
x	x	x	x	0	1	1	1	OUTIN
x	x	x	x	0	1	1	0	OUTINX
x	x	x	x	0	1	0	0	UPDAT
x	x	x	x	0	0	1	1	INOUT
x	x	x	x	0	0	0	1	RDBACK

11 To determine the mode of the data set, test the high-order bit of the DCBOFLGS field of the DCB. If this bit is 1, the data set mode is output; if this bit is 0, the data set mode is input. (The symbolic names of all fields in the DCB are defined by the DCBD macro instruction.)

12 You may want to position the tape if you have closed an input data set before all data has been read.

13 Move your CCW into the channel program area of the control program's work area. (The symbolic name of the first entry in the channel program area is DXCCW.) You can use the first six entries.

14 Issue an EXCP macro instruction specifying the address of the control program's IOB. (The symbolic name of the IOB is DXIOB.)

15 Techniques used to check for correct volume differ depending on the label formats used in the installation.

16 Label processing routines differ by label format.

17 If a write operation is required, this block can be used.

18 Issue an EXCP macro instruction specifying the address of the control program's IOB. (The symbolic name of the IOB is DXIOB.)

If the command is a rewind, set the rewind-issued bit in the UCB (UCBWGT field, bit 3) before issuing the EXCP.

If the command is a rewind-unload, set the unit-not-ready bit in the UCB (UCBFL1 field, bit 1) and zero out the UCB volume serial number field (UCBVOLI) after the channel program is complete.

19 Set the high-order bit to 1 in the UCBDMCT field of the UCB.

20 Use the load multiple (LM) instruction.

21 Use the FREEMAIN macro instruction to free the work area obtained in step 2.

22 Use the XCTL macro instruction, specifying the appropriate operand.

The following coding sequence illustrates an exit from your routine during open or close operations. Register 4 contains the address of the control program's open/close work area.

```
         USING     IECDSECT,4
         LR        1,SAVEBASE          put work area pointer in
                                       register 1 for FREEMAIN
         LM        2,14,REGSAVE        restore caller registers
         FREEMAIN  R,LV=size,A=(1)
         BALR      15,0                use 15 as temporary base
         USING     *,15
         MVC       0(8,6),MODNAME      module name to
                                       open/close area
         LA        15,DXCCW12          use open work area
         XCTL      EPLOC=(6),SF=(E,(15))
MODNAME  DC        C'IGGxxxxx'
```

The following coding sequence illustrates an exit from your routine during end-of-volume operations. Register 4 contains the address of the control program's EOV work area.

```
         USING   IECDSECT,4
         LR      1,SAVEBASE          put work area pointer in
                                     register 1 for FREEMAIN
         LM      2,14,REGSAVE        restore caller registers
         FREEMAIN R,LV=size,A=(1)
         BALR    15,0                use 15 as temporary base
         USING   *,15
         MVC     DXXMODNM,MODNAME    module name to EOV area
         LA      15,DXCCW12          use EOV work area
         LA      5,DXXMODNM          address of module name
         XCTL    EPLOC=(5),SF=(E,(15))
MODNAME  DC      C'IGG055xx'
```

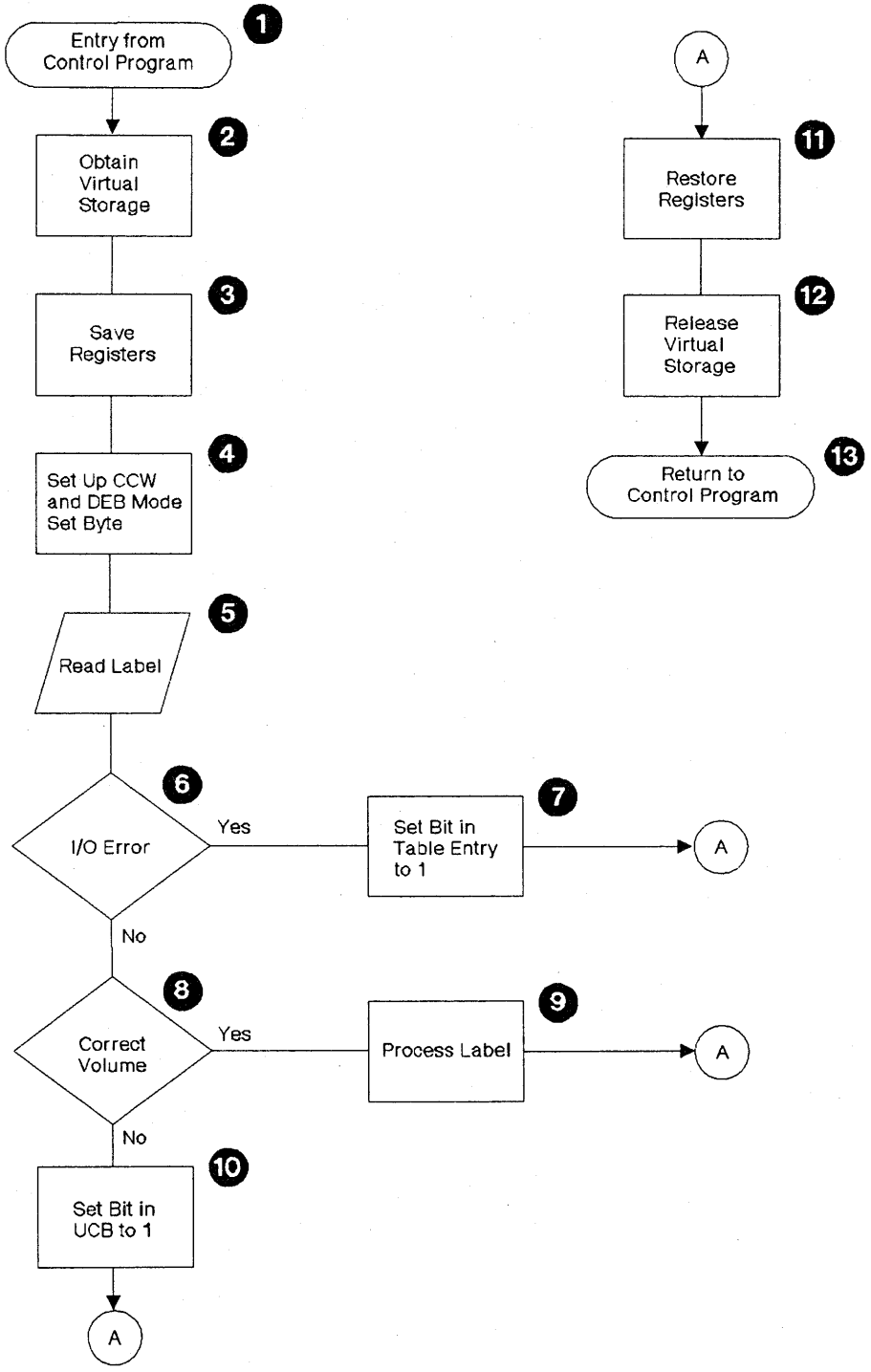


Figure 94. General Flow of a Nonstandard Label Processing Routine After Receiving Control from the Restart Routine

Explanation of Logic Blocks—Figure 94

1 The entry is in the form of an XCTL macro instruction issued by the control program.

2 Use the GETMAIN macro instruction to obtain virtual storage.

3 Use the store multiple (STM) instruction.

4 Move your CCW into the channel program area of the control program's work area. (The symbolic name of the first entry in the channel program area is RSCCW1.)

The device modifier byte, RSDEBMOD, in the DEB portion of the restart work area, is provided by the control program and will contain the mode-set command for the data portion of the tape. If the nonstandard labels at your installation are recorded in a mode different than the data, your NSL routine must set the device modifier byte (RSDEBMOD) to the density and recording technique of the labels. (See "Tape Characteristics" in *Magnetic Tape Labels and File Structure*.)

5 Issue an EXCP macro instruction specifying the address of the control program's IOB. (The symbolic name of the IOB is RSIOB.)

6 Determine if an uncorrectable I/O error occurred. This can be any type of error that you do not want to accept.

7 Set the high-order bit to 1 in the TABTPLBL field of the table entry.

8 Techniques used to check for correct volume differ depending on the label formats used in the installation. The volume serial number for the mounted volume is shown in the UCB.

9 Perform any necessary label processing and tape positioning.

10 Set the high-order bit to 1 in the UCBDMCT field of the UCB.

11 Use the load multiple (LM) instruction.

12 Use the FREEMAIN macro instruction to free the work area obtained in step 2.

13 Use the XCTL macro instruction. The following coding sequence illustrates an exit from your routine.

```
         LR       1,SAVEBASE          put work area pointer in
                                      register 1 for FREEMAIN
         LM       2,14,REGSAVE        restore caller registers
         FREEMAIN R,LV=size,A=(1)
         BALR     15,0                use 15 as temporary base
         USING    *,15
         L        1,4(,9)             put pointer to restart
                                      work area into register 1
                                      (see Figure 12)
         MVC      128(8,1),MODNAME    put module name in
                                      restart work area
         LA       15,120(,1)          set up XCTL
                                      parameter pointers
         LA       5,128(,1)           set up XCTL
                                      parameter pointers
         XCTL     EPLOC=(5),SF=(E,(15))
MODNAME  DC       C'IGC0K05B'
```

Inserting Nonstandard Label Routines Into the Control Program

Because they are type 4 SVC routines, nonstandard label processing routines must be included in the control program as part of LPALIB. This is done during the system generation process. (The routines may also be inserted after system generation by link-editing them into LPALIB. This procedure is similar to replacing volume label editor routines, which is described in "Volume Label Verification and Volume Label Editor Routines" on page 196. The routines may also be added after system generation by using the SVCUPDTE macro. For information on the SVCUPDTE macro, see *SPL: Application Development Macro Reference*.)

Before your routines can be inserted into the control program, each load module must be a member of a cataloged, partitioned data set. You must name this data set with the SYS1 prefix (for example, SYS1.name).

To insert your load modules into the SVC library during system generation, you use the SVCLIB macro instruction. With this macro instruction, you must specify the name of the partitioned data set and the names of members to be included in the SVC library. Member names for the first load module of each type of label processing routine are listed below. Member names for additional load modules must begin with the letters NSL or IGC. The format and specifications of the SVCLIB macro instruction are in *System Generation*.

Nonstandard Label Processing Routine	Control Program Routine	Member Name
Input Header	Open	NSLOHDRI
	EOV	NSLEHDRI
Output Header	Open	NSLOHDRO
	EOV	NSLEHDRO
Input Trailer	EOV	NSLETRLI
Output Trailer	EOV	NSLETRLO
	Close	NSLCTRLO
Restart Header	Restart	NSLRHDRI

Automatic Volume Recognition (AVR) Nonstandard Label Processing Routine

To enable the AVR option to process nonstandard magnetic tape labels, you must write a routine to supply AVR with information concerning the nonstandard labels. This routine is inserted in the control program in place of an IBM-supplied routine that causes AVR to reject tape volumes that do not have standard labels. The information returned to AVR by your routine consists of a validity indication (for example, the label read is valid) and the location within the nonstandard label of the volume serial number field. Specifically, your routine must:

1. Determine if the label under consideration is a valid, nonstandard label as defined by your installation.
2. Set general register 15 to zero if a valid label is detected, or to nonzero if the label is not recognizable. (A nonzero return causes AVR to unload the tape volume and issue an error message.)

3. When a valid label is detected, place the location of the volume serial number field within the label in an area provided by AVR. (The label, or the first part of it, is read into an 80-byte work area by AVR before your routine receives control; the location is defined within this work area. Also before your routine receives control, AVR positions the tape at the interrecord gap after the nonstandard label.)
4. Return control to AVR. Register 14 contains the return address. (The SAVE and RETURN macro instructions may be used in your routine.)

Your label processing routine receives control when the AVR routine cannot identify the first record on a magnetic tape volume as a standard label. The various error conditions that can occur during verification of the first record are explained under "Volume Label Verification and Volume Label Editor Routines" on page 196.

Entry Conditions

When your routine receives control, the AVR routine has placed the nonstandard label in an 80-byte work area, and general register 1 contains the address of a 2-word area whose contents are as follows:

Word 1 The address of the beginning byte of the 80-byte work area

Word 2 The address of a 1-word area where your routine stores the beginning address of the volume serial number field within the nonstandard label

Conventions

The format of your installation's nonstandard label(s) must provide for a 6-byte volume serial number field within the first 80 bytes of the label. Otherwise, the volume serial number will not be read into the 80-byte internal work area. This does not restrict the overall nonstandard label format from being more, or less, than 80 bytes in length.

The name of your routine must be IEFXVNSL.

Inserting AVR Nonstandard Label Routines into the Control Program

You may replace the IBM-supplied routine IEFXVNSL with your routine by link editing your assembled routine into the SYS1.AOSB3 data set prior to system generation, or you may replace the IBM-supplied routine after system generation by link editing your assembled routine into the control program module. The module is IEFXVAVR and the object deck step is as follows:

	Object Deck
INCLUDE	SYSLMOD(IEFXVAVR)
ALIAS	IEFXV001
ENTRY	IEFXV001
NAME	IEFXVAVR(R)

Volume Verification and Dynamic Device Reconfiguration

If you use nonstandard tape labels and you want to use the dynamic device reconfiguration (DDR) option, you must perform your own volume verification. Note that you must be able to perform your verification within the first 48 bytes of any record in your nonstandard label.

Before system generation time, code a routine named NSLREPOS and link-edit it into a cataloged partitioned data set. Then, identify the member of the partitioned data set that contains NSLREPOS in the LPALIB system generation macro instruction. Link-edit NSLREPOS into the LPALIB after system generation.

When your NSLREPOS routine receives control from the DDR tape reposition routine, register 2 contains a pointer to an XCTL list (built by DDR) in IORMSCOM. This list contains the module name to which you transfer control when you return control to DDR. Register 5 points to a buffer (SVRBEXSA) containing the first 48 bytes of a record of your label. The serial number of the volume against which verification is made is in the STREVOL1 field of the UCB. Register 7 contains the UCB address.

Before returning control, your routine should put one of the following hexadecimal codes into register 0:

Code	Explanation
0 (X'00')	Volume verification is complete. Because a tapemark follows this label, the tape reposition routine must position the tape to that tapemark and clear the block count it has accumulated before it begins repositioning.
4 (X'04')	The NSLREPOS routine needs more information for volume verification. When the tape reposition routine receives this code, it reads the first 48 characters of the next record into the buffer and returns control to NSLREPOS.
8 (X'08')	The wrong volume has been mounted. When the tape reposition routine receives this code, it sends a message to the operator explaining that the wrong volume has been mounted.
12 (X'0C')	Volume verification is complete. Since no tapemark follows this label, the tape reposition routine repositions the volume, using the block count it has accumulated.
16 (X'10')	Volume verification is complete. Because the tapemark following the label has already been reached, the tape reposition routine clears the block count it has accumulated and repositions the volume.

If NSLREPOS uses any registers other than register 0 or 14, the routine must save the registers in subpool 245 (using a GETMAIN macro) and store them in its own area before returning control to the tape reposition routine. When your NSLREPOS routine returns control to DDR, the following sequence should be used:

```
LR      15,2
XCTL    SF=(E,(15))
```

Volume Label Verification and Volume Label Editor Routines

If you specify that an input or output tape has a standard label, the operating system checks for the standard volume label at the beginning of the tape. For ISO/ANSI/FIPS tapes, the system checks for the correct version. If you specify that the tape has nonstandard labels or no labels, the system attempts to verify that the first record is not a standard volume label.

Because of conflicting label types or conflicting tape characteristics, various error conditions can occur during this verification of the first record. Under some error conditions, the tape is accepted for use. Under other error conditions, the tape is not accepted and the system issues another mount message. For certain other error conditions, the system gives control to a volume label editor routine; your installation can use routines supplied by IBM or it can supply its own routines. The IBM-supplied volume label editor routines determine the discrepancies between the requested tape and the mounted tape and, if necessary, pass control to the appropriate data management routine to create or destroy labels, as required. Installation-supplied routines can perform other functions.

Verification of First Record

The system reads the first record on the tape in accordance with the following criteria:

- If a single-density 9-track tape unit is used, the record is read in the density (800 bpi, 1600 bpi, or 6250 bpi) of the unit. If the record cannot be read, a unit check occurs.
- If a dual-density 9-track tape unit is used, the record is read in its existing density, provided that density is available on the unit. If the density is not available, a unit check occurs. If the record is a 7-track record, a unit check occurs.
- If a 7-track tape unit is used, the first record is read in the density specified by the user and in the translate on, even parity mode. If the record is in another density or mode, or is a 9-track record, a unit check occurs. ISO and ANSI do not specify support of 7-track tape for information interchange.
- If an 18-track tape unit is used, the record is read in the density of the unit. If the record cannot be read, a unit check occurs. ISO and ANSI standards do not include a specification of 18-track magnetic tape for information interchange.

As previously explained, various error conditions can occur during the system's verification of the initial record on a tape. The system actions resulting from these error conditions are shown in Figures 95, 96, and 97. Figure 95 on page 197 shows the actions when standard labels are specified; Figure 96 on page 198 shows the actions when nonstandard labels are specified; Figure 97 on page 199 shows the actions when no labels are specified.

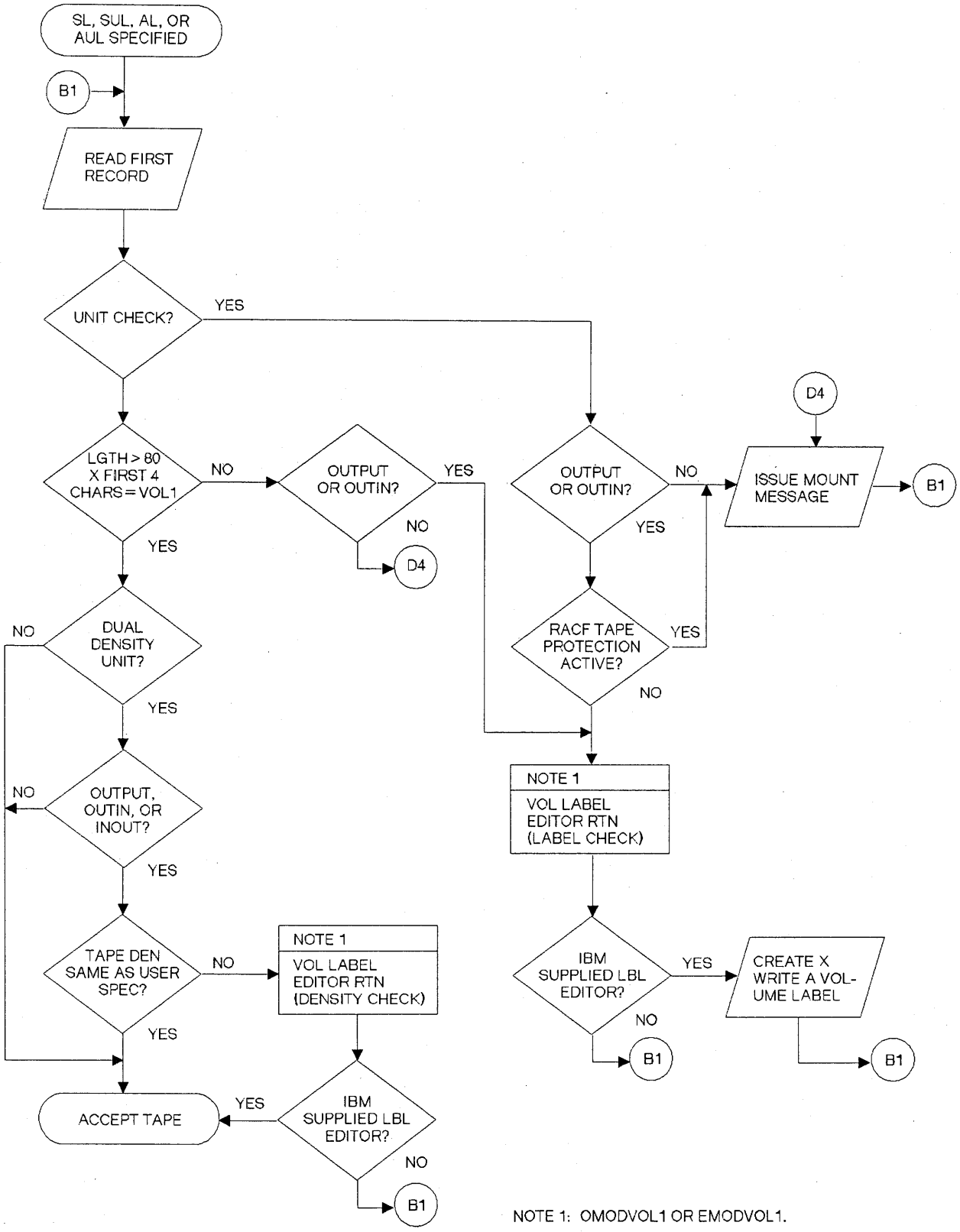


Figure 95. Verification of First Record When Standard Labels Are Specified

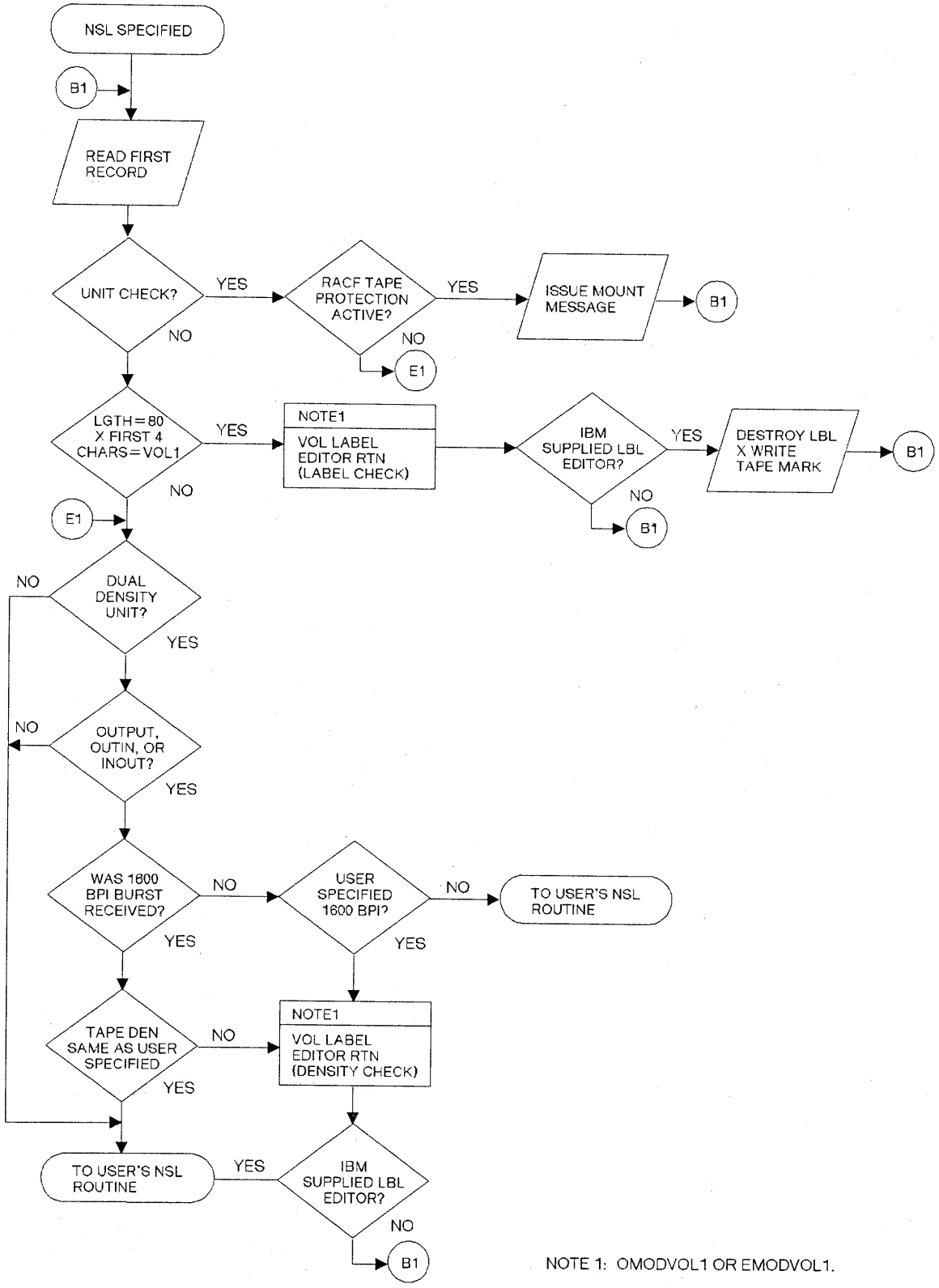


Figure 96. Verification of First Record When Nonstandard Labels Are Specified

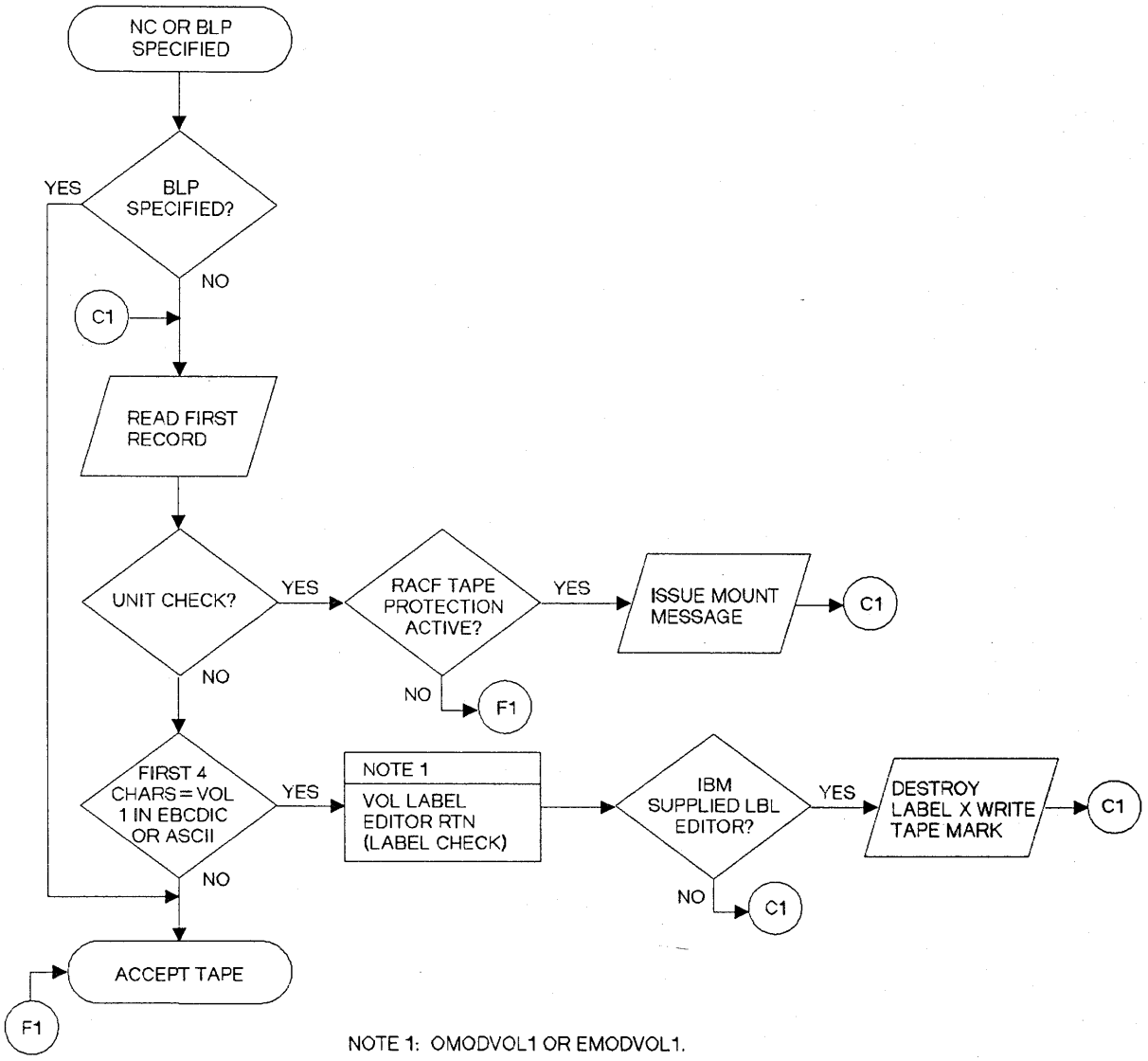


Figure 97. Verification of First Record When Unlabeled Tape Is Specified

Volume Label Editor Routines

When data sets are written on tape, data management's open or EOV routine may detect conflicts between:

- The label type specified by the user and the actual label type on the mounted output volume (OUTPUT or OUTIN).
- The recording density specified by the user and the actual density of the output volume (OUTPUT, OUTIN, or INOUT) mounted on a dual-density tape unit.
- The volume serial number specified by the user and the actual volume serial number on the mounted output volume (OUTPUT or OUTIN).
- The existing label version on the mounted output volume and ISO/ANSI/FIPS Version 3.

When such conflicts occur, control is given to the volume label editor routines. The IBM-supplied editor routines determine whether the data management routines can resolve the conflict.

If the volume label editor routines accept a conflict while opening to the first data set on an ISO/ANSI/FIPS Version 3 volume, the system will enter RACF, check the expiration date, and enter the file access exit before requesting permission from the operator to create a new VOL1 label (the volume access exit is entered prior to label conflict processing).

If a nonspecific volume request is made for a standard labeled tape, but the mounted volume does not have a standard label, data management issues a message to the operator requesting that the volume serial number and owner information be supplied or, optionally, that the use of this tape volume be refused.

Note: If a specific volume request is made and the label format of the mounted volume does not match the format specified in the processing program, data management will reject the tape and issue a message to mount another volume. However, if a specific volume request is made for an SL tape and the mounted tape is unlabeled, data management gives the operator the option of labeling or rejecting the tape.

If a nonspecific volume request is made for a nonstandard labeled or unlabeled tape, but the mounted volume has a standard label, data management gives the operator the option to allow or refuse the use of the tape under the following conditions:

- The file sequence number is not greater than 1.
- The expiration date has passed, or the operator has allowed the use of the tape.
- The volume is not password protected nor is it RACF protected and the accessor is ALTER authorized.

If the preceding conditions are not met, data management rejects the tape and issues a mount message. Data management follows the same procedure if the conditions are met, but the operator refuses the use of the tape.

If the operator accepts the tape, data management destroys the volume label by overlaying it with a tapemark and deletes the RACF definition of the volume if it was found to be RACF defined and the user is ALTER authorized.

Note: Even if the password is known, a password-protected tape that is not RACF defined is not converted to NL or NSL.

For dual-density tapes with standard labels, data management rewrites the labels in the density specified when an output request is made to the first data set on a volume. When an output request is made to other than the first data set, the labels are rewritten in the density specified in the existing labels.

For tapes with ISO/ANSI/FIPS labels, data management rewrites the VOL1 label only in the case of a density conflict.

If the existing ISO/ANSI/FIPS label is not Version 3 during an output request to the first data set on the volume, the volume label editor routines offer an option

that allows the label to be rewritten to conform to Version 3 standards. The WTOR installation exit may be used to provide label information for the new Version 3 label instead of requiring the operator to supply it via a WTOR message (see "Appendix D. Version 3 Installation Exits"). If a version conflict is detected for an output request to other than the first data set, the volume is unconditionally rejected by open/EOV after issuing an IEC512I LBL STD "VRSN" error message.

You can replace the IBM-supplied editor routines with installation routines that resolve the conflict to your own specifications. Your editor routines can resolve label and density conflicts by writing labels, by overwriting labels with a tapemark, and by performing write operations to set the correct density on a dual-density tape device. Or, your editor routines can reset the appropriate system control blocks (in effect, change the program specifications) to agree with the label type and/or density of the currently mounted volume. Or, you may desire a combination of these actions, including demounting of the volume under certain conditions. You may include all of these possible actions in the design of your editor routines.

There are two IBM-supplied editor routines. One gets control from the open routine for handling the first or only volume of a data set. The other gets control from the EOV routine for handling the second and subsequent volumes of a multivolume data set. You can replace either or both of these routines.

The remainder of this section provides the information necessary for writing editor routines and inserting them into the control program.

Programming Conventions

Your editor routines must conform to the same general programming conventions as the nonstandard label processing routines discussed under "Programming Conventions" on page 180, for size, design, register usage, entry points, and work areas. As discussed under "Nonstandard Labels" on page 174, you must use the EXCP macro instruction to perform needed input/output operations.

You must name the first (or only) module of your routines as follows:

OMODVOL1	The editor routine associated with open
EMODVOL1	The editor routine associated with EOV

If your editor routines consist of more than one load module, names for the additional modules must begin with the prefix OMODVOL for the open routine, or EMODVOL for the EOV routine. Transfer between the modules must be by name.

Note: With an IBM 3480 Magnetic Tape Subsystem, the open and EOV routines normally use EXCP appendages when processing labels. For the duration of the open or EOV, they normally save labels in virtual storage buffers to improve performance by avoiding an unnecessary change of direction on the tape. The EXCP appendages simulate most types of channel programs that read. For channel programs that they do not simulate, they move the tape to the point where your routine expects the tape to be and then allow the channel program to execute. They are designed to do simulation so as to appear to have no effect except to improve performance.

If your routine does I/O, it should use the DCB that is in the work area. The DEB appendage vector table should not be substituted or modified.

Program Functions

Figure 98 on page 203 presents the five conditions under which the open or EOV routines transfer control to your editor routines. Each condition suggests a general action that your routine could take to permit processing of the current volume to continue. The first two conditions (density checks) arise only when the tape volume is mounted on a dual-density tape device.

General flowcharts of editor routines are shown in Figure 99 on page 205 and Figure 100 on page 206. These flowcharts suggest the logic that you could use in your routines. The logic is shown separately for routines that receive control from the open or EOV routine of the control program. Each block in the flowcharts is numbered, and the number corresponds to an item in the list of explanations that follows. Other items to note are:

- The logic in the flowcharts is oriented toward resolving the label and density conflicts by altering the characteristics of the mounted volume.
- Figure 100 on page 206 (the EOV editor routine) does not contain logic blocks corresponding to blocks 5, 18, and 19 in Figure 99 on page 205 (the open editor routine). These blocks represent functions that you must program only when receiving control from the open routine. You must test all the DCBs defined by the OPEN macro instruction before returning control to the open routine. When you receive control from the EOV routine, there is only one DCB to process.
- If your installation does not support expiration date and protection checking on nonstandard label volumes, and does not desire to maintain such checking on standard label volumes, you need not implement the functions of logic blocks 6 through 14 in the flowcharts.
- The DCB is copied into protected storage during Open/Close/EOV processing. During open and close processing, register 7 points to a parameter list that contains the addresses of the DCB in protected storage. During EOV processing, register 2 points to the DCB in protected storage. The address of the user's DCB is in the Open/Close/EOV work area at the label DXUDCBAD. If the DCB is to be changed, both copies must receive the same change.

Figure 98 (Page 1 of 2). Editor Routine Entry Conditions from the EOV Routine

Program Specification	Mounted Volume Characteristics	Transfer Conditions	Possible Editor Routine Action
AL or SL-9-track tape	AL[1] or SL	Density Check[2]	Overwrite the standard label with a standard label. The first write from load point sets the recording density on a dual-density device. (See Figure 99 or 100—blocks 15B, 16, and explanation.)
NSL[3]-800 bpi	NSL or NL or 1600 bpi density	fit=8. Density Check[2]	Write a tapemark to set density. The program specification NSL will cause control to be given to your nonstandard label routines after return to Open or EOV. (See Figure 99—blocks 15, 15B, and 16. If your installation supports protection and retention date checking on NSL volumes, see block 6.)
SL or AL	NSL or NL[5]	Label Check[6]	Write a standard volume label. (See Figure 99—blocks 15, 15A, and 16. If your installation supports protection and retention date checking on NSL volumes, see block 6.)
NL or NSL	SL[4] or AL[1]	Label Check	Overwrite standard label with tapemark, for example, cancel. (See Figure 99—blocks 15, 15A, and 16.) Depending on whether NL or NSL is specified by the program, open or EOV will either position tape (NL) or transfer control to your non- standard label routines (NSL).
AL	SL	Label Check	Overwrite an IBM standard label with a Version 3 VOL1 label.
SL	AL[1]	Label Check	Overwrite ISO/ANSI/FIPS label with an IBM standard label.

Figure 98 (Page 2 of 2). Editor Routine Entry Conditions from the EOV Routine

Program Specification	Mounted Volume Character-istics	Transfer Conditions	Possible Editor Routine Action
AL	AL[1]	Compatibility Conflict	Overwrite an ISCII/ASCII label with a Version 3 label (first file output only).

Legend:

AL ISO/ANSI/FIPS standard volume label
SL IBM standard volume label
NSL Nonstandard volume label
NL No volume label

Notes:

1 The open and EOV routines position the tape at load point before transferring control to the editor routines.

2 ISO/ANSI/FIPS standard labeled tape cannot be overwritten without the permission of the console operator.

3 Dual-density devices only.

4 If NL is specified, no density check is performed. For NL volumes, tape is positioned at load point and recording density is set by the first write command.

5 If the volume is mounted on a dual-density device, a density condition may also exist. It will be corrected by the write operation.

6 When SL is specified, a label check may also indicate that the system could not recognize the first record because of a unit check condition.

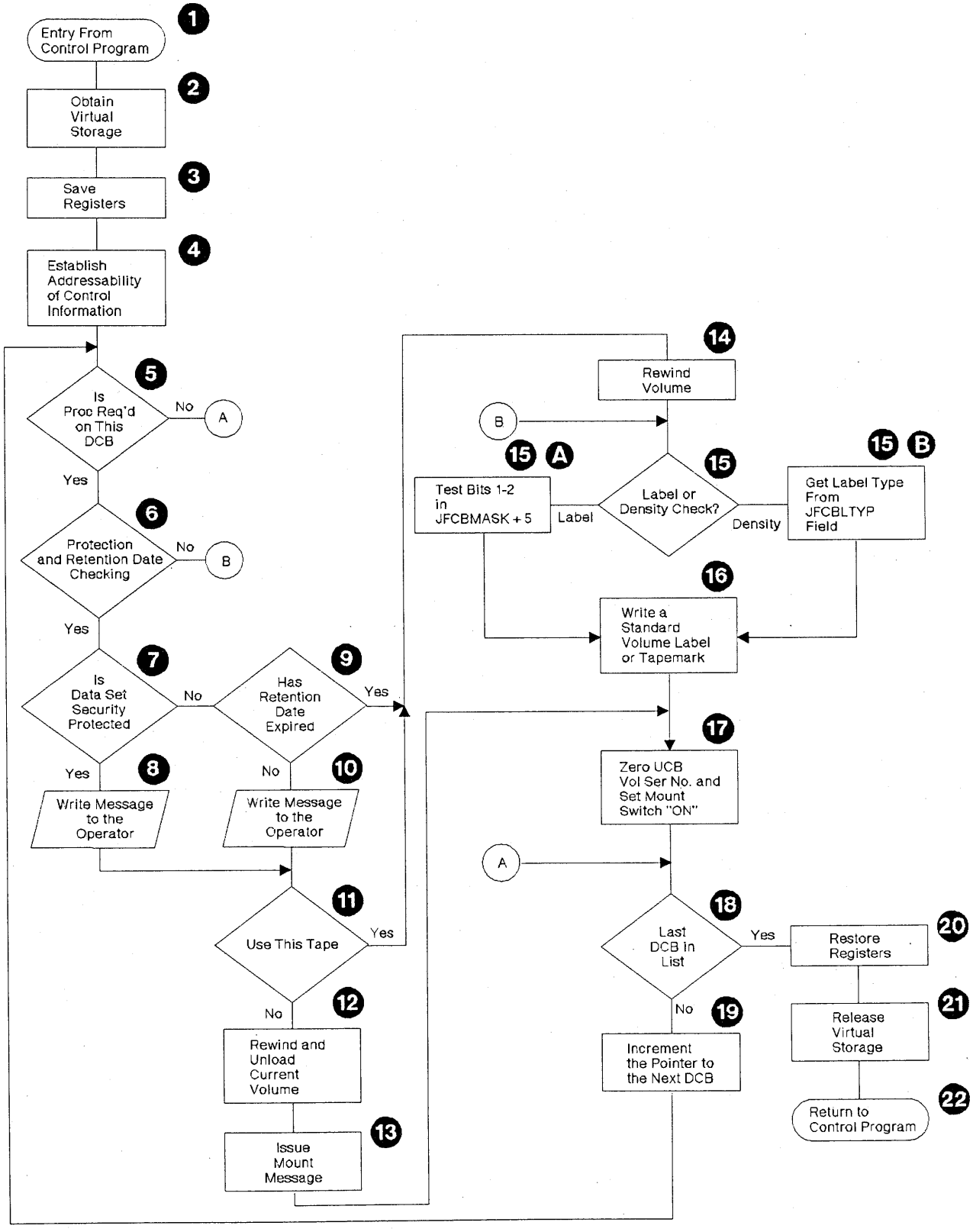


Figure 99. General Flow of an Editor Routine after Receiving Control from the Open Routine

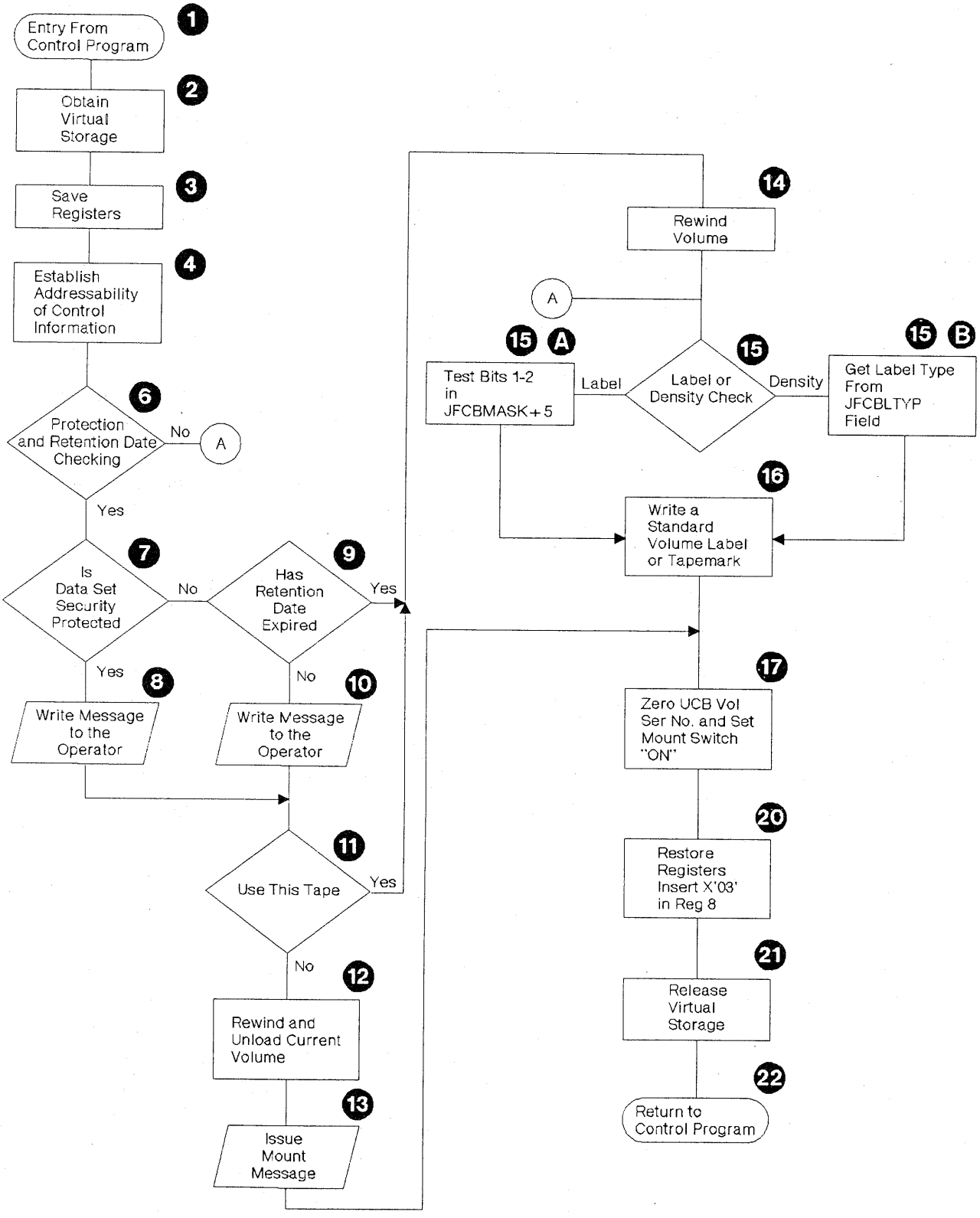


Figure 100. General Flow of an Editor Routine after Receiving Control from the EOV Routine

Explanation of Logic Blocks—Figures 99 and 100

1 Your exception routine receives control from the open or EOV routines of the control program.

2 Use the GETMAIN macro instruction. The virtual storage you obtain must contain all your work areas, including those used to read in a label or write a label.

3 Use the store multiple (STM) instruction.

4 Figure 88 on page 182 provides the information you need to establish addressability of the DCB address list and work and control block area for each DCB defined by the OPEN macro instruction.

When you receive control from the EOV routine, general register 2 contains the address of the DCB for the data set, and general register 4 contains the address of the work and control block area associated with the DCB.

The IECDSECT macro instruction (described in "Mapping the Common O/C/EOV Workarea" on page 184) symbolically defines the fields of the work and control block area (see Figure 89 on page 183).

You will also need to address the UCB for the device on which the tape volume is mounted. The address of the UCB may be obtained from the DXDEBUCB field of the DEB defined by the IECDSECT macro instruction. The IEFUCBOB macro instruction (described in *System—Data Administration*) defines the fields of the unit control block.

5 Bit configurations in the byte addressed by JFCBMASK+5 indicate whether label checks or density checks have occurred, and, in the case of a label check, the condition that caused the check. At this point, you test bits 0 and 3. If either bit is set to 1, processing is required. However, if bits 6 and 7 of DCBOFLGS are set to 0, you should discontinue processing. When bit 6 (lock bit) is 0, the control program cannot open the DCB. When bit 7 (busy bit) is 0, the DCB is already being processed or is already open.

The field JFCBMASK is defined by the IECDSECT macro instruction. Bit settings in the byte at JFCBMASK+5 are defined as:

Bits	Setting	Meaning
0	1	Label check has occurred.
1	1	Standard label (SL or AL) specified; no label/nonstandard label on mounted volume. **Note:** If JFCBAL (AL label requested) is set and UCCBBSTR is set in the UCB (ISCII/ASCII tape is mounted), an ISO/ANSI/FIPS version conflict has occurred, and a valid Version 3 volume label must be created.
2	1	No label (NL) or nonstandard label (NSL) specified; standard label (AL or SL) on mounted volume.
3	1	Density check has occurred.
4-7	—	Reserved for possible future use.

6 If your installation supports a protection and retention date scheme involving nonstandard labels, and/or you want to maintain retention date and protection checking on standard labels, you must incorporate code in your editor routines to check for protection and retention date expiration.

If checking is desired, you must, at this point, read the first record and determine the label type.

To perform the I/O operation, move your CCWs into the channel program field of the work and control block area. The symbolic name for the first entry in this field is DXCCW. Then issue an EXCP macro instruction specifying the address of the control program's IOB. The symbolic name for the IOB is DXIOB. These fields (DXCCW, DXIOB) are defined by the IECDSECT macro instruction.

Note: There are 12 CCW locations in the DXCCW field. You can only use the first six locations.

7 To check the retention date and/or protection fields in a standard label, you must read the data set header 1 record into a work area. The format of the nonstandard label defined by your installation determines how you access the protection and retention date fields in the nonstandard label. Step 6 provides directions for handling the I/O operation.

8 Write a message to the operator that the volume is protected and to determine if it is to be used.

9 Repeat step 7 above.

10 Write a message to the operator that the expiration date for the mounted volume has not elapsed and to determine if it is to be used.

11 If the volume is to be used, continue processing to resolve label or density conditions.

12 Rewind and unload the currently mounted volume. Step 6 provides directions for handling the I/O operation. When you issue the rewind and unload command, you must turn on the UCB not-ready bit (UCBFL2) after the ECB has been posted. If you want the open/EOV mount verification routines to handle the mounting/demounting on volume verification, set bit 4 (X'08') of JFCBMASK+5 in the open/EOV work area and go to block 22 to return to open/EOV. Subsequent volume level errors will cause the label editor routines to be reentered.

13 Write a message to the operator requesting demounting of the current volume and mounting of a new volume. The device name (in EBCDIC) may be obtained from the UCBNAME field of the UCB. Step 6 provides directions for handling the I/O operation.

14 If a new volume is to be mounted, repeat step 6.

15 Test bit 3 of the byte at JFCBMASK+5. If set to 1, control was received as a result of a density check.

Test bit 0 of the byte at JFCBMASK+5. If set to 1, control was received as the result of a label check.

a If control was received as the result of a label check, test bits 1 and 2 of the byte at JFCBMASK+5. See step 5.

b If control is received as the result of a density check, use the JFCBLTYP field in the JFCB to ascertain the type of label specified in the program. A hexadecimal 04 indicates a nonstandard label (NSL) has been specified; a hexadecimal 02 indicates that a standard label has been specified.

16 When correcting a density check or label check condition, and an NSL or NL is specified by the program, you must write some kind of record on the tape that will be interpreted by the open or EOV routines as a nonstandard label or no label; for example, it does not contain VOL1 in the first four bytes of the record. The easiest way to do this is to write a tapemark. Upon return to open or EOV and reverification of the label, the specification for label type and density will have been met. Open or EOV will transfer control to your nonstandard label routines if NSL is specified, or position the tape for writing if NL has been specified.

You must supply information for the label identifier, the label number, and the volume serial number fields, and record the balance of the label as blanks.

You enter VOL in the label identifier field, a 1 in the label number field, and a 6-character serial number in the volume serial number field.

Note: To ensure that two or more tape volumes carrying the same serial number are not produced, write to the operator at this point for assignment of a serial number.

Data set header labels 1 and 2 are constructed by the open or EOV routine after control is returned to them.

Note: At this point, you can change the control block settings to conform to the characteristics of the tape volume mounted (that is, reset the label type field in the JFCB to conform with the type of label on the volume mounted and change the density field in the DCB to the density of the tape mounted).

17 The symbolic name for the volume serial number field in the UCB is UCBVOLI. The mount switch is the high-order bit of the field named UCBDMCT in the UCB. These fields are defined by the IEFUCBOB macro instruction. Perform an exclusive OR (XC) operation on the UCBVOLI field with itself and perform an OR (OI) operation on the UCBDMCT field with X'80'. This will cause the mount verification routines to bypass further label processing and reverify the tape without an intervening demount.

18 When receiving control from the open routine, you must process the entire DCB list. The last entry in the list can be recognized by a 1 in bit 0 of the first byte in the entry.

19 You increase the pointer to the DCB address list by 4 bytes. You must also increase the pointer to the work and control block area for each DCB. You increase this pointer by 8 bytes.

20 Use the load multiple (LM) instruction.

21 Use the FREEMAIN macro instruction.

22 Return control to the open or EOV routine by means of an IECRES macro instruction, specifying the module to be given control as follows:

Return From	To Module
OMODVOL1	IGG0190A (Open)
EMODVOL1	IGG0550P (EOV)

Note: Open and EOV will rewind the volume upon receiving control from OMODVOL1 or EMODVOL1.

Return is via the XCTL macro instruction (E-form). See Figure 94 on page 191 and its accompanying "Explanation of Logic Blocks—Figure 94 ."

Inserting Your Label Editor Routines into the Control Program

You will be replacing the IBM-supplied modules OMODVOL1 and/or EMODVOL1 with your routines. Use the replace function of SMP or link-edit your editor routines into SYS1.LPALIB after system generation.

Note: When using SMP, specify DLIB member name IFG0193C or IFG0553C in the ++MOD statement.

The setup for making the linkage editor run is shown below.

```
                .
                .
                .
//jobname       JOB     [parameters]
//stepname      EXEC    PGM=HEWLH096[,PARM='LET,RENT,...']
//SYSPRINT      DD      SYSOUT=A
//SYSUT1        DD      UNIT=SYSDA,SPACE=(parameters)
//SYSLMOD       DD      DSNAME=SYS1.LPALIB,DISP=OLD
//SYSLIN        DD      *
                .
                .
                .
     Object Deck for Open
                .
                .
                .
ENTRY           OMODVOL1
ALIAS           IFG0193C
NAME            OMODVOL1(R)
                .
                .
                .
     Object Deck for EOV
                .
                .
                .
ENTRY           EMODVOL1
ALIAS           IFG0553C
NAME            EMODVOL1(R)
/*
```

Note: Your modules will be placed into the LPA the next time an IPL with a CLPA is done. You must have requested space for the LPALIB directory entries for the additional modules when the library was allocated.

ISO/ANSI/FIPS Version 3 Installation Exits

Four installation exits are provided, as defaults, for ISO/ANSI/FIPS Version 3 volumes:

- Volume access,
- File access,
- Label validation, and
- Label validation suppression.

A fifth installation exit, WTOR, can be written (or modified, if one has already been written) by your installation to convert ISO/ANSI/FIPS non-Version 3 to Version 3 labels (see "WTOR Installation Exit").

All the default installation exit routines are supplied in a module containing a single CSECT (IFG0193G, alias IFG0553G), in SYS1.LPALIB. A copy of the source code for the module is contained in member ANSIEXIT of SYS1.SAMPLIB.

The default routines, except the validation suppression exit, reject the volume. They execute in a privileged (supervisor) state and can be modified or replaced to perform I/O (such as overwriting a label), change system control blocks, and mount or demount volumes. The return code from the exits may be modified to request continued processing. However, results are unpredictable in cases in which the label validation exit is entered and it has not been modified to also correct certain errors. The prolog of the source code for the exits, in SYS1.SAMPLIB, gives additional details on modifying the exits.

A parameter list, mapped by the macro IECIEPRM, is passed to the exit routines. The same parameter list is passed to the RACF installation exits if a volume is RACF protected and the VOL1 access code is A through Z. However, whereas return codes from the Version 3 exits are returned in the IECIEXRC field of the parameter list, return codes from the RACF exits are returned in register 15 (return codes from the Version 3 and RACF exits are not the same). Neither the Version 3 nor RACF installation exits should alter any of the parameter list fields, except IECIEXRC or IECIEUSR. For information about RACF installation exits, see *SPL: RACF*.

An important extension to the parameter list is the UCB tape class extension. It contains such items as the volume access code (UCBCXACC), owner identification (UCBCXOWN), and ISO/ANSI/FIPS version (UCBCXVER). The address of the appropriate UCB tape class extension is maintained in the parameter list.

WTOR Installation Exit

For ISO/ANSI/FIPS tape volumes, MVS/DFP supports output only to ISO/ANSI/FIPS Version 3 and input from either ISO/ANSI/FIPS Version 1 or Version 3. If a label version conflict is detected during an output request to the first data set on a volume, the WTOR message "IEC704A C" is issued to the installation operator to obtain information for rewriting the volume label as a Version 3 label. If your installation does not want the operator to have to provide the necessary label information (volume serial number, owner identification, and volume access code), it may use the WTOR installation exit to intercept message "IEC704A C" and provide this information.

The name of the WTOR installation exit routine is IEECVXIT. For information on how to use the linkage editor to include module IEECVXIT in the control program, see *System Modifications*. WTOR message "IEC704A C" described in *System Messages Volume 1* and *System Messages Volume 2*.

Label Validation Exit

The label validation exit is entered during open/EOV if an invalid label condition is detected, and label validation has not been suppressed. Invalid conditions include unsupported characters, incorrect field alignment, unsupported values (for example, RECFM=U, block size greater than 2048, or a zero generation number), invalid label sequence, asymmetrical labels, invalid expiration-date sequence, and duplicate data set names.

Input to the exit will be the address of the exit parameter list containing the type of exit being executed, the type of error detected, location of the error, and an address for the label in error.

Except for duplicate data set name checking, label validation occurs only at tape load point (beginning-of-volume label group) and at the requested data set position (beginning-of-data-set label group); only duplicate name checking occurs *during* positioning to the requested data set.

Trailer labels produced by the system are not validated during close or EOV for the old volume. Thus, an input data set read in a forward direction is processed during close/EOV even if it is followed by an invalid trailer label; however, later, if the same data set is read backward, the invalid label will be detected during open or EOV for the new volume and cause the label validation exit to be entered.

Because modifications to an existing data set can result in nonsymmetrical trailer labels, the following open options will cause the label validation exit to be entered:

- Open for MOD (OLD OUTPUT/OUTIN), INOUT, EXTEND, or OUTINX.
- Open for an EXCP DCB (OUTPUT/OUTIN) that does not contain at least a 4-word device dependent area for maintaining a block count.

Note: If you have generalized library subroutine programs that specify the INOUT option, but you are using a tape for input only, you can avoid entering the exit by coding LABEL=(,AL,,IN) on the JCL DD statement.

The label validation exit can either continue processing a volume or reject it, issuing one of the following return codes:

Return Code	Meaning
X'00'	Continue processing volume
X'04'	Reject volume (set by the IBM-supplied exit)

To identify the invalid condition, an IEC512I LBL STD message is issued to the operator. For a rejected volume, an abend code message is also issued.

Entry to the label validation exit is tracked in the UCB. This serves as an audit trail if the exit forces continuation for an invalid condition but the condition causes an abend in subsequent processing.

Note: The system does not rewrite labels after return from the label validation exit. Therefore, if a label is to be corrected, it must be done by an installation-written label validation exit. If certain errors are not corrected, they will cause unpredictable results when the volume is processed by a return code of zero from the label validation exit. These errors include:

- Incorrect sequencing
- Unsupported characters
- Incorrect field alignment
- Certain unsupported values (RECFM=U, block size greater than 2048, and a zero generation number *will* be processed as expected by the system)

If an error is corrected by a return code of zero from the label validation exit, the resulting volume may not meet the specifications of Version 3 standards, and will therefore require agreements between interchange parties.

Label Validation Suppression Exit

The validation suppression exit allows the option of suppressing label validation. It is entered during open/EOV if volume security checking has been suppressed (JSCBPASS is on), if the volume label accessibility field contains an ASCII space character, or if RACF accepts a volume and the accessibility field does not contain an uppercase letter from A through Z.

Label validation can also be suppressed by the volume access exit.

Note that if you suppress label validation, the resulting volume may not meet the specifications of Version 3 standards, and therefore would require agreements between interchange parties.

Volume Access Exit

The volume access exit is entered during open/EOV if a volume is not RACF protected and the accessibility field in the volume label contains an ASCII uppercase letter from A through Z. The exit is bypassed if volume security checking has been suppressed (as indicated in the Program Properties Table).

The exit can accept or reject the volume and can suppress label validation, issuing one of the following return codes:

Return Code	Meaning
X'00'	Use volume
X'04'	Reject volume (set by IBM-supplied exit)

Label validation is suppressed by setting the high-order bit of the return code in the field named CONTROL in the source module ANSIEXIT (for example, a return code of 80 would indicate to use the volume and suppress validation). This bit is acted on every time the exit returns to the system.

Note that the volume access exit is complementary to the label validation suppression exit, in that it is entered only when the latter exit is not.

File Access Exit

The file access exit is entered after positioning to a requested data set if the accessibility field in the HDR1 label contains an ASCII uppercase letter from A through Z and the volume is not RACF protected. Likewise, the exit is entered when a data set will be written to an output volume if the first character of the JCL ACCODE keyword is A through Z.

The exit can either accept the data set or reject the volume, issuing one of the following return codes:

Return Code	Meaning
X'00'	Use data set
X'04'	Reject volume (set by IBM-supplied exit)

The file access exit can reject a volume that was accepted earlier by the volume access exit.

Installation-Written Exit Routines

If you replace any of the IBM-supplied exit routines with your own routines, observe the programming conventions described under "Programming Conventions" on page 180, except that return must be via a BR 14 instruction.

Your routines should not alter any fields in the exit parameter list, except the return code (IECIEXRC) and the field reserved for user data (IECIEUSR).

In addition, your routines cannot use the DCB parameter list to process any DCB other than the current entry, because the DCBs are not synchronized during Version 3 exit processing.

It is necessary to MODESET to key 0 in order to alter protected control blocks (such as the UCB). The original key at entry should always be restored immediately after any alterations to key 0 storage are complete; this will minimize risk of inadvertent data destruction.

Exit Parameter List—IECIEPRM

The parameters passed to a Version 3 installation exit during label processing will vary slightly between different types of exits. These differences, when they exist, are noted in the "Exit Type" column in Figure 101 on page 215. The parameter list is passed to the exit as an address in general purpose register 1; it is 32 bytes in length and is mapped by macro IECIEPRM beginning at DSECT IECIEPRM. This macro is available only in assembler language. Parameter fields not available to a particular exit will be set to zero(s). The only fields allowed to be altered by an exit are the return code (IECIEXRC) and the user area (IECIEUSR); changing any other field will have an unpredictable effect on system processing. A flag in the parameter list indicates which type of exit was entered.

Offset	Field Name	Length	Contents	Exit Type	Comments
+0	IECIEID	4	C'APRM'	all	Parameter list identifier
+4	IECIESIZ	4	X'20'	all	Length of IECIEPRM
	IECIESZ	=	32	all	Comparand for size
+8		4	X'00'	all	Reserved
+12	IECIEFL1	1	flags	all	Exit flags
	IECIEVAL	=	X'80'	VAL	Entry is Validity Check
	IECIEVAE	=	X'40'	VAE	Entry is Volume Access
	IECIEFAE	=	X'20'	FAE	Entry is File Access
	IECIEVSP	=	X'10'	VSP	Entry is Validation Suppression
	IECIEWRT	=	X'08'	all	Label will be written ("WRITE MODE")
	IECIEEOV	=	X'04'	all	EOV in process
+13	IECIEERR	1	flags	VAL	Validation error type
	IECIEVRS	=	X'80'	N/A	Version compatibility conflict (Note 1)
	IECIEUNK	=	X'40'	VAL	Unsupported/unknown value
	IECIEADJ	=	X'20'	VAL	Invalid field alignment
	IECIESEQ	=	X'10'	VAL	Label sequence error
	IECIEDUP	=	X'08'	VAL	Duplicate file name
	IECIECHR	=	X'04'	VAL	Invalid character type
	IECIEXPR	=	X'02'	VAL	Invalid expiration date
	IECIESYM	=	X'01'	VAL	Symmetry conflict (Note 5)
+14	IECIEPOS	1	X'offset'	VAL	Starting character position in in label examined (Note 2)
+15	IECIEXRC	1	X'04'	all	Return code from exit processing (Note 3)
	IECIESUP	=	X'80'	VAE, VSP	Suppress label validation (Note 8)
	IECIERC0	=	X'00'	all	Accept volume
	IECIERC4	=	X'04'	all	Reject volume (ignored for VSP Exit)
+16	IECIEJAC	1	C'access code'	FAE	User-requested file accessibility code (Note 7)
+17	rsvd	2	0	all	Reserved for future use
+19	IECIEDCB	1	flags	all	Copy of open parmlist options (4 low-order bits)
	IECIEOUT	=	X'02'	all	Bit on for OUTPUT,OUTIN
	IECIEIN	=	X'0E'	all	Bits off for INPUT,RDBK
+20	IECIELBL	4	A(address)	all	Address of label being processed (Note 4)
+24	IECIEUCB	4	A(address)	all	Address of UCB for volume from VOL1 label (Note 6)
+28	IECIEUSR	4	0	all	User area (maintained across exits)
+32	IECIEND	0	0		End of exit parameter list

FAE = File Access Exit
VAE = Volume Access Exit
VAL = Label Validation Exit
VSP = Validation Suppression Exit

Figure 101. ISO/ANSI/FIPS Version 3 Exit Parameter List

Notes to Figure 101:

1. "Version" error is set for the O/C/E message routine for internal use, and the volume is unconditionally rejected.
2. The first character position is offset 0, the second position is offset 1, and so forth.
3. A return code of 4 is set by the IBM-supplied exits. This will cause a volume to be rejected. The exception is the validation suppression exit, which always sets a return code of zero in the IBM-supplied exits (although the system will *always unconditionally accept* a volume after execution of the validation suppression exit). IECIEXRC is ignored by open/EOV when control is returning from RACF.
4. For volume access exit and file access exit, the label area contains the accessibility code from tape. When the label area is not available to the exit, IECIELBL will be zero. Data in the label that is not available to an exit will be indicated by binary zeros. The volume accessibility code is *always* available in the UCB tape class extension at UCBCXACC (for ISO/ANSI/FIPS) when an ISO/ANSI/FIPS volume has been opened and not demounted.
5. A symmetry conflict results from a condition that will produce nonmatching or asymmetrical labels framing a file, and/or inconsistent file structure.
6. The UCB tape class extension for ISO/ANSI/FIPS volumes will contain the VOL1 label standard version number, the VOL1 owner identification, and the VOL1 accessibility code. The extension can be addressed by the following sequence:

   ```
   L     Rx,UCBEXTPT(,UCBREG)   COMMON EXTENSION
   L     Rx,UCBCLEXT(,Rx)       CLASS EXTENSION
   USING IECUCBCX,Rx            IECUCBCX MAPPING
   ```

   The base UCB may be useful to access the serial number for the mounted volume (in UCBVOLI).
7. The file accessibility code in IECIEJAC is only valid when "Write Mode" (IECIEWRT) is set during the file access exit. This code comes from ACCODE (A-Z) or LABEL (password, "1" or "3") parameters from the user job step (blank, if none). The value in IECIEJAC, when IECIEWRT is set, will be written (if valid) as the accessibility code in the file label when the exit returns.
8. IECIESUP will be recognized any time the volume access exit returns to the system, when RACF returns to the system after it was passed the parameter list, or when the validation suppression exit returns to the system.

UCB Tape Class Extension—IECUCBCX

The tape class extension area generated for a UCB is addressed by UCBCLEXT in the UCB common extension (created at system generation time). The pointer will be zero when no class extension exists. The tape class extension will contain zeros at IPL, and will be set to zeros whenever the volume label is about to be verified and processed for accessibility (as in open, or as in "next volume" for EOV). The main purpose of the class extension is to hold volume label data across opens when there is no intervening volume label reverification (as would be the case after CLOSE LEAVE and another OPEN in the same job step). The tape class extension area is mapped by the UCBCX DSECT in the IECUCBCX macro.

UCB Tape Class Extension Data Area

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
0	(0) STRUCTURE	56	UCBCX	UCB TAPE CLASS EXTENSION
0	(0) CHARACTER	8	UCBCXPRE	TAPE CLASS EXT PFIX
0	(0) CHARACTER	4	UCBCXID	ID = UCBT
4	(4) UNSIGNED	2	UCBCXTLN	TOTAL LENGTH OF EXTENSION
6	(6) UNSIGNED	2	UCBCXCLR	LENGTH TO CLEAR
ANSI SECTION				
8	(8) CHARACTER	20	UCBCXANS	ANSI PORTION OF EXTENSION
8	(8) CHARACTER	1	UCBCXACC	VOL1 ACCESS CODE FROM LABEL
THE UCB EXIT FLAGS (UCBCXFL1) ARE SET WITH AN AUDIT TRAIL FOR ANSI EXIT ACTIVITY DURING VOLUME VERIFICATION.				
9	(9) BITSTRING	1	UCBCXFL1	FLAG BYTE
	1...		UCBCXVAL	VALIDATION EXIT ENTERED
	.1..		UCBCXSUP	SUPPRESS LBL VALIDATION CHECK
10	(A) CHARACTER	1	UCBCXVER	VOL1 LABEL-STANDARD VERSION
11	(B) CHARACTER	1	UCBCXV3	VERSION 3
12	(C) UNSIGNED	1	UCBCXRS1	RESERVED FOR FUTURE USE
13	(D) CHARACTER	14	UCBCXOWN	VOL1 OWNER IDENTIFICATION
27	(1B) CHARACTER	2	UCBCXRS2	RESERVED FOR FUTURE USE
3480 SECTION				
29	(1D) CHARACTER	28	UCBCX1	3480 PORTION OF EXTENSION
29	(1D) CHARACTER	16	UCBCXENV	3480 ENVIRONMENTAL DATA
29	(1D) UNSIGNED	2	UCBCXERG	NO. OF ERASE GAPS
31	(1F) UNSIGNED	2	UCBCXCLN	NO. OF CLEANER ACTIONS
33	(21) UNSIGNED	2	UCBCXRD	READ FWD DATA CHECKS
35	(23) UNSIGNED	2	UCBCXRDB	READ BKWD DATA CHECKS
37	(25) UNSIGNED	2	UCBCXWR	WRITE DATA CHECKS
39	(27) UNSIGNED	3	UCBCXMBR	# OF BYTES READ/4K
42	(2A) UNSIGNED	3	UCBCXMBW	# OF BYTES WRITTEN/4K
45	(2D) UNSIGNED	2	UCBCXSEQ	TRAILER FILE SEQUENCE NUMBER
47	(2F) UNSIGNED	1	UCBCXFL2	3480 FLAG BYTE
48	(30) UNSIGNED	1	UCBCXCKP	CHKPNT ('C') IN EOF2
49	(31) UNSIGNED	1	UCBCXSC1	PASSWD REQ'D R/W (EOF1)
50	(32) UNSIGNED	1	UCBCXSC3	PASSWD REQ'D WR ONLY (EOF1)
51	(33) UNSIGNED	1	UCBCXWRT	EOF LABEL WAS WRITTEN
52	(34) UNSIGNED	1	UCBCXRS3	RESERVED
53	(35) UNSIGNED	4	UCBCXTUS	SERIAL NO. OF TAPE DRIVE
56	(3A) CHARACTER	4	UCBCXRS4	RESERVED
60	(3E) CHARACTER	16	UCBCX2	
60	(3E) CHARACTER	6	UCBCXVID	VOL ID SAVE AREA
66	(44) UNSIGNED	10	UCBCXRS5	RESERVED
76	(4E) *		UCBCXEND	END OF TAPE CLASS EXTENSION

Note: The UCB exit flags (UCBCXFL1) are set with an audit trail for Version 3 exit activity during volume verification.

Chapter 11. Automatic Class Selection (ACS) Installation Exits

General Guidance

There are four Storage Management Subsystem (SMS) constructs—data class, storage class, management class, and storage group. One ACS routine exists for each of these constructs. Each ACS routine, except the one for storage group, has an ACS installation exit. The ACS installation exits allow you to code exit routines that provide capabilities beyond the scope of the ACS routines. The ACS installation exit routines you code are processed when the corresponding ACS installation exit is invoked. The ACS installation exits are Assembler H programs. The exit routines you code must also be Assembler H programs.

This chapter is intended for the storage administrator. It explains how to write ACS installation exit routines and provides a sample for purposes of illustration. For more information on defining SMS classes and storage groups, and writing ACS routines, see the *Storage Administration Reference*.

Choosing Between ACS Routines and ACS Installation Exits

ACS routines and ACS installation exit routines can perform many of the same functions. Wherever possible, ACS routines should be used, rather than exit routines. Compared with ACS installation exit routines, ACS routines are relatively easy to write, maintain, and modify. However, only the ACS installation exit routines can be used to

- Call other programs
- Call other subsystems
- Write SMF records
- Write GTF trace records
- Invoke SVC dumps
- Maintain large, easily searched tables of information in storage.

Note that you must re-IPL the system after modifying or creating any ACS installation exit routines, because they reside in SYS1.LPALIB.

The Exit Environment

Execution of ACS installation exits is the second of four steps during automatic class selection. The system processes which cause automatic class selection are:

- Allocation of new data sets that are eligible to be SMS managed
- Conversion of SMS volumes and data sets
- DFHSM RECALL and RECOVER
- DFDSS COPY/RESTORE and CONVERTV
- Access method services ALLOCATE, DEFINE, ALTER and IMPORT

Figure 102 on page 220 summarizes the process that occurs for each of the SMS constructs whenever automatic class selection takes place. This process occurs first for the data class ACS routine, then for the storage class ACS routine. If no storage class was assigned, automatic class selection ends here.

If a storage class was assigned, the process occurs again for the management class ACS routine and then for the storage group ACS routine.

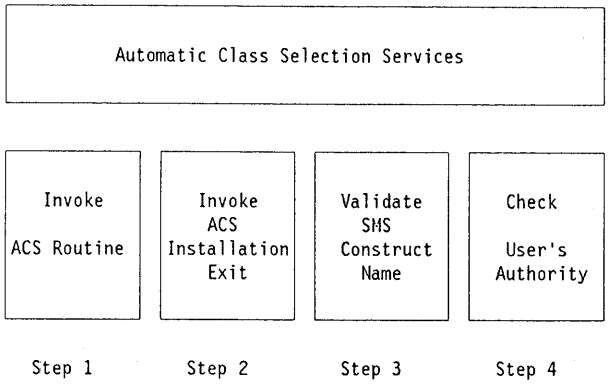


Figure 102. Automatic Class Selection Process for Each Construct

During automatic class selection, ACS Services performs the following four steps. In step 1, ACS Services invokes the ACS routine for an SMS construct, which assigns a value for that construct to a data set, if that ACS routine exists. This assigned value can be overridden and replaced by a value determined by the ACS installation exit routine. In step 2, ACS Services invokes the corresponding ACS installation exit. If an ACS installation exit routine exists, then it executes. ACS installation exit routine assignments override ACS routine assignments if the two differ. Also, ACS installation exit routines can alter the input to ACS routines and re-invoke them one time. In step 3, ACS Services verifies that a valid value for the construct name has been assigned. Finally, in step 4, ACS Services checks the user's authority to determine if an end user is allowed to use a given storage class or management class.

Note: For storage groups, only step 1 is performed, because there is no storage group ACS installation exit.

Finding the ACS Installation Exits

The ACS installation exits reside in SYS1.LPALIB and have the following names:

IGDACSDC Data class exit
IGDACSSC Storage class exit
IGDACSMC Management class exit

You can code routines for any, all, or none of the ACS installation exits. You will receive a CSV003I message for each ACS installation exit routine that does not exist in SYS1.LPALIB. The message does not represent an error unless you have coded a routine for that ACS installation exit and ensured that it resides in SYS1.LPALIB.

Programming Considerations

In general, the routines you code for the ACS installation exits must:

- Handle multiple requests (reentrant)
- Reside in SYS1.LPALIB
- Have AMODE 31
- Have RMODE ANY

The exit routines are given control in task mode and protect key zero with no locks held and 31-bit addressing mode. They must not be operating in cross-memory mode.

Linkage is via standard MVS linkage conventions. Figure 103 illustrates the parameter structure for the ACS installation exits. A 4K byte work area exists on a doubleword boundary for each ACS installation exit. You can use this work area to satisfy the reentrant requirement. The following macros map the parameters that are passed to each ACS installation exit:

IGDACERO Maps the read-only variables that the ACS installation exit can reference when selecting an SMS class.

IGDACERW Maps the read-write variables that the ACS installation exit can set when selecting an SMS class.

IGDACSPM Describes the parameter list for an ACS installation exit.

See Appendix C, "Read-Only Variables Referenced by ACS Installation Exits" on page 243, Appendix D, "Read-Write Variables Set by ACS Installation Exits" on page 249, and Appendix E, "Parameter List for ACS Installation Exits" on page 251 for the individual macro mappings.

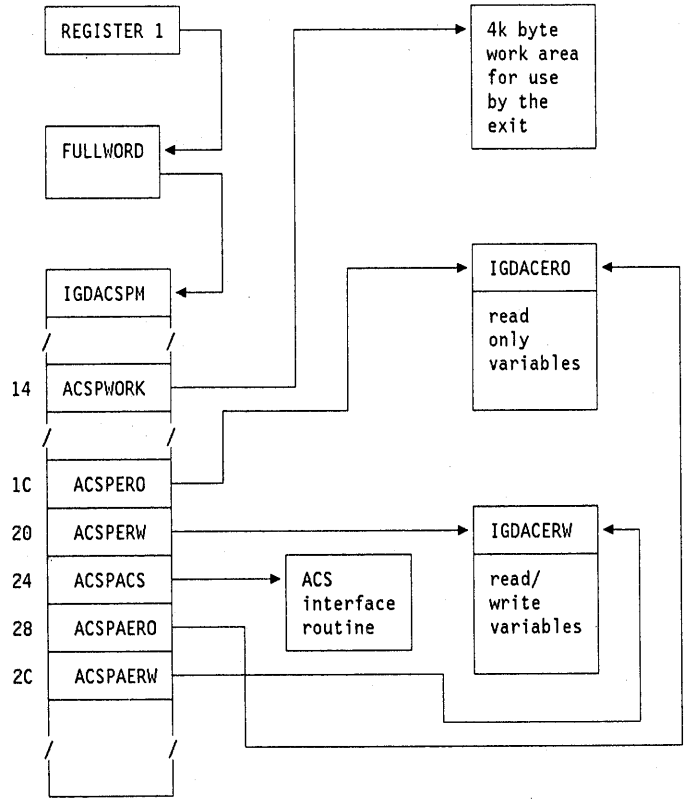


Figure 103. Parameter Structure for the ACS Installation Exits. This figure shows the control block structure upon entry into the exit. All offsets are in hexadecimal.

The ACS Installation Exit Routines

ACS Services allows an installation-written exit routine to take control after the ACS routine has processed. An ACS installation exit routine can override any values assigned by the ACS routine and can return messages to the batch job, started task, or TSO/E user.

An ACS installation exit routine can invoke an ACS routine one time, to determine a new value for a construct. This new value can be assigned to a data set or used for comparison with the original value assigned in step 1, Figure 102 on page 220. Your exit routine uses the ACS interface routine to invoke an ACS routine. The parameter list that is passed to your exit routine, as shown in Figure 103 on page 221, contains a field (ACSPACS) which points to the ACS interface routine. For more information on how to invoke the ACS interface routine, see "Invoking ACS Interface Routines" on page 223.

Assigning SMS Classes

Upon entry to an ACS installation exit, ACSPERW points to a list of read-write variables, which are mapped by IGDACERW. To assign an SMS class, you must code an installation exit routine for the corresponding ACS installation exit that sets the following fields in IGDACERW:

- Set the ACERWNCS field to one. This field specifies the number of SMS classes to be assigned. The valid values for this field are zero and one. Initially, it contains zero, indicating no SMS classes are to be assigned. If more than one SMS class is returned, only the first is accepted.
- Set the ACERWVLN field to the length of the SMS class name. This field has an initial value of zero.
- Set the ACERWVAL field to the name of the SMS class being assigned. This field has an initial value of zero.

To assign a null value (zero) to an SMS class, you must set the ACERWNCS field to one and the ACERWVLN field to zero. The ACERWVAL is then ignored. If you do not want to assign a class, leave the ACERWNCS field as zero.

Returning Messages

An ACS installation exit routine can return a series of messages to the batch job, started task, or TSO/E user. To return messages, set the value in ACERWNMG equal to the number of messages you want returned. Place the text of the messages in ACERWMSG. ACERWMSG can hold up to six messages. Messages must be 110 bytes long. Pad messages with blanks if needed.

Registers at Return from an ACS Installation Exit Routine

When you return to ACS Services from your ACS installation exit routine, register contents must be as follows:

Register	Contents
0	Contains a reason code, if any
1-14	Same as on entry to your exit routine
15	A return code

Return Codes

The ACS installation exit routine must terminate with a return code in register 15 that indicates what action is to be taken upon return from the exit. The return codes and their meanings are as follows:

Code	Meaning
00(X'00')	Indicates processing completed normally, and that you want the SMS class that the ACS installation exit returns to be used.
04(X'04')	Indicates that you want the job or the dynamic allocation request to be failed, and that register 0 contains the relevant reason code.
16(X'10')	Indicates that the ACS installation exit contains at least one error. You want it to be placed in disabled wait until the SMS address space has restarted.

Any other return code represents an error.

Reason Codes

Your ACS installation exit routine can put a reason code in register 0. You determine the reason codes and their meanings. When your exit routine passes back a return code of X'04' in register 15, the reason code your exit routine placed in register 0 appears in the text of message IGD1001I.

Invoking ACS Interface Routines

The ACS interface routine is used to invoke an ACS routine from your ACS installation exit routine. The ACSPACS field shown in Figure 103 on page 221 contains the address of an ACS interface routine that invokes the corresponding ACS routine for the SMS construct being selected. If the ACSPACS field contains a zero, then no ACS routine exists for the SMS class. The IGDACERC macro defines the ACS interface routine return codes, which appear in register 15. The IGDACERC macro also defines the reason codes which appear in register 0 if the return code is nonzero.

Linkage is via standard MVS linkage conventions. Figure 104 on page 225 illustrates the parameter structure for the ACS interface routine. The parameter list for the ACS interface routine, ACSPACSP, is imbedded within the parameter list that is passed to the exit. ACSPACSP contains the following fields:

ACSPAERO Points to a list of variables mapped by IGDACERO that are read-only variables used by the ACS interface routine. Initially, ACSPAERO points to the same list of read-only variables as ACSPERO. You can modify the passed variables pointed to by ACSPAERO and invoke the ACS routine. Alternatively, you can code the ACS installation exit routine to create an entirely new list of read-only variables for the ACS interface routine and point to them with ACSPAERO before invoking the ACS routine.

ACSPAERW Points to a list of read-write variables mapped by IGDACERW and used by the ACS interface routine. Initially, ACSPAERW contains the same value as ACSPERW, which is a pointer to a list of read-write variables that contain the original value for the SMS class that was derived in step 1 of Figure 102 on page 220. You can invoke the ACS routine without changing the value in ACSPAERW. When the ACS routine executes, it replaces the values in the list pointed to by ACSPAERW with new values for the SMS class derived by the ACS routine and any messages generated by the ACS routine. When your ACS installation exit routine returns control to ACS Services, the SMS class contained in the list pointed to by ACSPERW is assigned to the data set. Because ACSPERW and ACSPAERW are pointing to the same list, the class that is assigned to the data set will be the new class that was created when your exit routine invoked the ACS routine.

Alternatively, you can code your ACS installation exit routine to create an entirely new list of read-write variables and point to them with ACSPAERW before invoking the ACS routine. After the ACS routine executes, the read-write variable list pointed to by ACSPAERW contains the new values derived by the ACS routine. The read-write variable list pointed to by ACSPERW contains the original values that were created in step 1 of Figure 102 on page 220. By creating a new read-write variable list in your exit routine, and then invoking the ACS routine, you can compare the original values, pointed to by ACSPERW, with the new values, pointed to by ACSPAERW.

Note: If you have created a new list of read/write variables and invoked the ACS routine, and you wish to have the *new* values that are pointed to by ACSPAERW used to assign a class, *you must copy the new values into the original list of variables,* pointed to by ACSPERW. If you omit this copying step, the new values will *not* be used to assign a class.

A value of zero in the ACERWNCS field upon return from the ACS routine indicates that a null value was derived for the SMS class. To assign a null value to the SMS class, you must set the ACERWNCS field to one and leave the ACERWVLN and ACERWVAL fields zero.

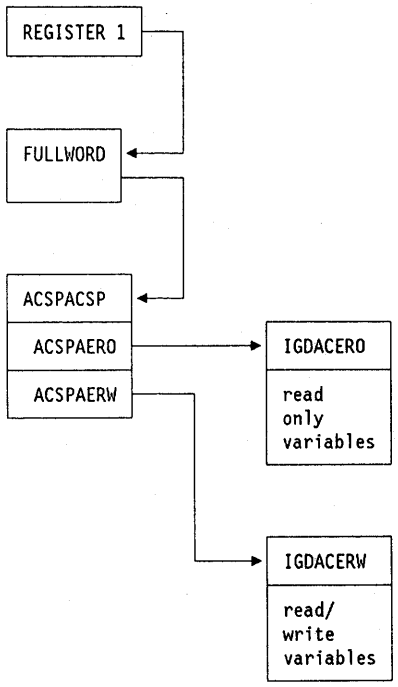


Figure 104. Parameter Structure for the ACS Interface Routine. This figure shows the control block structure for invoking the ACS interface routine from the installation exit.

Reference Restrictions

When coding an ACS installation exit routine, you should only reference data that is explicitly passed, because too many different environments can invoke an ACS installation exit. For example, an ACS installation exit routine should not reference the name of the current system, the ASCB for the current address space, or the CPUID of the current processor. Also, an ACS installation exit routine should not issue a Dynamic Allocation Request, because a Dynamic Allocation Request can invoke an ACS installation exit.

ACS Installation Exit Errors

SMS establishes a recovery environment for the ACS installation exits by issuing an ESTAE before invoking them. The following represent ACS installation exit errors:

- The SMS ESTAE covering the exit is entered.
- The ACS installation exit returns an invalid return code.
- The ACS installation exit returns return code 16.

If any of the above errors occur, the following results:

1. An output message describes the error.
2. An SVC dump is taken, SYS1.LOGREC error recording is performed, and the failing ACS installation exit is marked invalid.
3. The failed ACS installation exit is not reinvoked until the SMS address space is restarted.

Sample ACS Installation Exit Routine

The ACS installation exit routine in Figure 105 re-invokes the storage class ACS routine and writes two messages.

```
        TITLE ' SAMPLE STORAGE CLASS INSTALLATION EXIT'
IGDACSSC CSECT ,
IGDACSSC AMODE 31                       MUST RUN IN 31-BIT MODE
IGDACSSC RMODE ANY                      SHOULD HAVE AN RMODE OF ANY
         DS    0H
         USING *,15
         B     PROLOG
         DC    AL1(16)
         DC    C'IGDACSSC  85.078'
         DROP  15
PROLOG   STM   14,12,12(13)             STANDARD ENTRY LINKAGE
         USING ACSPMD,PARMLIST          ESTABLISH ADDRESSABILITY TO
         L     PARMLIST,0(1)              EXIT PARAMETER LIST
         USING WORKAREA,WORKBASE        ESTABLISH ADDRESSABILITY
         L     WORKBASE,ACSPWORK          TO WORK AREA
         LA    11,SAVEAREA              STANDARD SAVE AREA
         ST    13,4(11)
         ST    11,8(13)
         LR    13,11
         LR    12,15                    LOAD BASE REGISTER
PSTART   EQU   IGDACSSC
         USING PSTART,12
         LA    TOADDR,STARTWK           SET UP TO CLEAR THE WORK AREA
         L     TOLEN,ACSPWLEN             FOLLOWING THE SAVE AREA
         SH    TOLEN,=AL2(L'SAVEAREA)   DON'T OVERLAY THE SAVE AREA
         LA    FROMLEN,0                INITIALIZE TO ZEROES
         MVCL  TOADDR,FROMADDR          CLEAR WORK AREA
*
*  SETUP ACCESS TO READ-ONLY VARIABLES
*
         USING ACERO,ERO                ANCHOR READ-ONLY VARIABLES
         L     ERO,ACSPAERO
*
*  SETUP ACCESS TO READ-WRITE VARIABLES
*
         USING ACERW,ERW                ANCHOR READ-WRITE VARIABLES
         L     ERW,ACSPERW
         MVC   ACERWVAL(8),BLANK8
*
*  INVOKE ACS ROUTINE
*
         LA    SERVICEP,ACSPACSP        PARAMETER LIST FOR ACS RTN
         ST    SERVICEP,ACSPARM         USE STANDARD MVS LINKAGE
         LA    1,ACSPARM                CONVENTIONS
         L     15,ACSPACS               LOAD ADDR OF ACS INTERFACE RTN
         BALR  14,15                    CALL STORCLAS SELECTION ROUTINE
*
*   RETURN MESSAGE INDICATING EXIT WAS ENTERED
*
         LA    WORKREG,1                ONE MESSAGE
         STH   WORKREG,ACERWNMG
         MVC   ACERWMSG(110),MESSAGE1
```

Figure 105 (Part 1 of 2). Sample ACS Installation Exit Routine

```
*
* IF THE ACS ROUTINE RETURNED A NULL VALUE, WE NEED TO DO SOME RESETS
*
         MVC   SCNAME,BLANK8
         MVC   SCNAME(8),ACERWVAL       PRIME AREA FOR MESSAGE 2
         CLC   ACERWNCS,=X'00000000'    IS NUMBER OF CONSTRUCTS = 0?
         BNE   CONTINUE                 IF ACS RTN SET IT, NO CHANGE
         MVC   ACERWNCS,=X'00000001'    SET NUMBER OF CONSTRUCTS = 1
         MVC   ACERWVLN,=X'0000'
         MVC   SCNAME,NULLSC            UPDATE AREA FOR MESSAGE 2
*
* BUILD A MESSAGE INDICATING A STORAGE CLASS WAS ASSIGNED
*
CONTINUE EQU   *
         LA    WORKREG,2                WE NOW HAVE 2 MESSAGES
         STH   WORKREG,ACERWNMG
         MVC   MSG2AOUT,MSG2A
         MVC   MSG2BOUT,MSG2B
         MVC   PAD,BLANK75
         MVC   ACERWMSG+110(110),MESSAGE2    MOVE IN SECOND MESSAGE
*
RETURN   EQU   *
         L     13,4(13)
         LM    14,12,12(13)
         LA    15,0                      SET RETURN CODE
         BR    14
*
DATA     DS    0F                        START OF DATA AREA
MESSAGE1 DC    CL110'STORAGE CLASS INSTALLATION EXIT ENTERED'
MSG2A    DC    CL14'STORAGE CLASS '
MSG2B    DC    CL13' WAS ASSIGNED'
NULLSC   DC    CL8' *NULL* '           INDICATE NO STORAGE CLASS
BLANK8   DC    CL8' '                  FOR CLEARING MESSAGE INSERT
BLANK75  DC    CL75' '
*
WORKAREA DSECT ,
SAVEAREA DS    CL72                     STANDARD SAVE AREA
STARTWK  EQU   *                        START OF NON-SAVE AREA STUFF
ACSPARM  DS    F                        POINTER TO ACS PARAMETERS
*
MESSAGE2 DS    0CL110                  MESSAGE TO ISSUE SO THAT
MSG2AOUT DS    CL14                    USER KNOWS WE ASSIGNED A
SCNAME   DS    CL8                     STORAGE CLASS
MSG2BOUT DS    CL13
PAD      DS    CL75
*
WORKREG  EQU   2                        WORK REGISTER
TOADDR   EQU   2                        REGISTER FOR MVCL DESTINATION
TOLEN    EQU   3                        REGISTER FOR MVCL DEST LENGTH
ERO      EQU   3                        ADDRESS OF IGDACERO
FROMADDR EQU   4                        REGISTER FOR MVCL SOURCE
ERW      EQU   4                        ADDRESS OF IGDACERW
FROMLEN  EQU   5                        REGISTER FOR MVCL SOURCE LENGTH
SERVICEP EQU   6                        ADDRESS OF PARMLIST FOR SERVICE
WORKBASE EQU   7                        ADDRESS OF WORKAREA
PARMLIST EQU   8                        ADDRESS OF EXIT PARAMETER LIST
*
IGDACSPM IGDACSPM                       MAP EXIT PARAMETER LIST(ACSPMD)
IGDACERO IGDACERO                       MAP READ ONLY VARIABLES(ACERO)
IGDACERW IGDACERW                       MAP READ-WRITE VARIABLES(ACERW)
         END IGDACSSC
```

Figure 105 (Part 2 of 2). Sample ACS Installation Exit Routine

Appendix A. Example of an OPEN Installation Exit Module

Processing in IFG0EX0B

The following program listing is a sample of IFG0EX0B. The four subroutines (BUFNO, SCREEN, RLSE, and SQTY) show examples of the kind of processing that can be done in your installation's version of IFG0EX0B.

The BUFNO subroutine defaults the number of buffers for QSAM DCBs (DCBBUFNO) if the value is zero when the exit is given control. The block size in the DCB (DCBBLKSI) is used, together with a fixed amount of storage (64K bytes in the example) to determine a buffer number. A buffer number is limited to a fixed value (32 in the example). Storage quantity and maximum buffer number are contained in two tables, DAMAX and TPMAX, that are used for DASD devices and tape devices, respectively. Storage quantity is expressed in units of 1024 (1K) bytes. The values in the DAMAX and TPMAX tables can be altered by your installation.

The SCREEN subroutine determines those cases in which the succeeding subroutines, RLSE and SQTY, should be executed. DASD sequential and partitioned data sets being processed by BSAM or QSAM and opened for OUTPUT or OUTIN are selected. The VTOC data set and data sets starting with 'SYS1.' (system data sets) are excluded. An installation may want to make further selection tests.

Requesting Partial Release

The RLSE subroutine sets on the partial release indicators in the JFCB if the number of extents in the data set is less than a fixed value (8 in the example). It sets off the partial release indicators in the JFCB if the number of extents in the data set is equal or greater than a fixed value (8 in the example). Partitioned data sets are not processed, because they may be opened many times to write one new member for each OPEN/CLOSE.

Updating the Secondary Space Data

The SQTY subroutine provides a default secondary space quantity if none is specified. The default is one half of the primary space quantity if it is greater than one. If the primary quantity is zero, secondary is set to a fixed default number of tracks (5 in the example). If the primary quantity is one, secondary is set to the same fixed default (5); note that, in this case, the secondary quantity is in units of tracks, cylinders, or average blocks, depending on the unit of the primary quantity.

If the secondary space quantity is not zero, the SQTY subroutine tests the number of extents in the data set. If the number of extents is equal to or greater than a fixed value (10 in the example), then the secondary quantity is increased by 50% if it is greater than 1. It is set to a default quantity (5 in the example) if the secondary quantity is one; note that, in this case, the secondary

quantity is in units of tracks, cylinders, or average blocks, depending on that of the primary quantity.

```
IFG0EX0B CSECT
*************************************************************************
*                                                                       *
* FUNCTION =                                                            *
*        FOUR SAMPLE ROUTINES ARE SUPPLIED.                             *
*                                                                       *
*    BUFNO - DEFAULT DCBBUFNO                                           *
*        DCBBUFNO (NUMBER OF BUFFERS) IS DEFAULTED FOR                  *
*        OPENS TO PHYSICAL SEQUENTIAL AND PARTITIONED DATA SETS         *
*        ON DASD AND TAPE USING QSAM, FOR WHICH DCBBUFNO IS ZERO.       *
*        DCBBUFNO FOR SYSIN, SYSOUT, TERMINAL, AND DUMMY DATA SETS      *
*        IS SET TO THE EQUATE, INOUTBNO, OR THE VALUE IN THE            *
*        FULLWORD, INOUTBN.                                             *
*                                                                       *
*        DCBBUFNO IS SET TO THE NUMBER OF DCBBLKSZ BUFFERS WHICH        *
*        FIT IN A GIVEN AMOUNT OF STORAGE. THE AMOUNT OF STORAGE IS     *
*        DEFINED BY THE EQUATES, DAMXK AND TPMXK (OR THE FULLWORDS      *
*        AT LABELS, DAMAXK AND TPMAXK), FOR DASD AND                    *
*        TAPE, RESPECTIVELY. THE EQUATES DEFINE THE AMOUNT OF           *
*        STORAGE FOR BUFFERS IN UNITS OF 1024 (IF DAMXK IS 32, THEN     *
*        THE AMOUNT OF STORAGE IS 32K, OR 32768).                       *
*        DAMXK OR TPMXK TIMES 1024 IS DIVIDED BY DCBBLKSI TO            *
*        DETERMINE THE NUMBER OF BUFFERS TO DEFAULT.                    *
*                                                                       *
*        THE EQUATES, DAMXBNO AND TPMXBNO, OR THE FULLWORDS             *
*        AT LABELS, DAMAXBNO AND TPMAXBNO,                              *
*        DEFINE THE MAXIMUM NUMBER OF BUFFERS TO BE                     *
*        DEFAULTED FOR DASD AND TAPE IF THE CALCULATION, ABOVE,         *
*        RESULTS IN A LARGER NUMBER.                                    *
*                                                                       *
*    SCREEN - SCREEN OUT CASES FOR RLSE, SQTY                           *
*                                                                       *
*    RLSE - SET OR ZERO PARTIAL RELEASE                                 *
*        THIS ROUTINE SETS PARTIAL RELEASE FOR DASD PS (NOT PO) DATA    *
*        SETS BEING OPENED FOR OUTPUT OR OUTIN.                         *
*                                                                       *
*        PARTIAL RELEASE IS SET ON IF THE NUMBER OF EXTENTS IS LESS     *
*        THAN A QUANTITY DEFINED BY THE EQUATE, RLSE1, OR THE BYTE,     *
*        EXTRLSE1.                                                      *
*                                                                       *
*        PARTIAL RELEASE IS SET OFF IF THE NUMBER OF EXTENTS IS NOT     *
*        LESS THAN A QUANTITY DEFINED BY THE EQUATE, RLSE0, OR THE      *
*        BYTE, EXTRLSE0.                                                *
*                                                                       *
*    SQTY - SET OR UPDATE SECONDARY SPACE QUANTITY                      *
*        THIS ROUTINE UPDATES THE SECONDARY SPACE                       *
*        QUANTITY FOR DASD PS OR PO DATA SETS BEING                     *
*        OPENED FOR OUTPUT OR OUTIN.                                    *
```

```
*       IF THE SECONDARY QUANTITY IS NOT ZERO,                        *
*       AND IF THE NUMBER OF EXTENTS IN THE DATA SET IS               *
*       AT LEAST EQUAL TO THE QUANTITY IN THE EQUATE, EXTSQT (OR      *
*       THE BYTE AT LABEL, EXTSQTY), THEN:                            *
*       1. IF THE SECONDARY QUANTITY IS GREATER THAN ONE,             *
*       SECONDARY QUANTITY IS INCREASED BY ONE HALF                   *
*       (50%).                                                        *
*                                                                     *
*       2. IF THE SECONDARY QUANTITY IS ONE,                          *
*       SECONDARY QUANTITY IS SET TO THE VALUE IN THE FULLWORD        *
*       AT LABEL, SQTYDFLT (EQUAL TO THE EQUATE, SQTYDFL).            *
*                                                                     *
*       IF THE SECONDARY QUANTITY IS NOT ZERO,                        *
*       AND IF THE NUMBER OF EXTENTS IN THE DATA SET IS               *
*       LESS THAN THE QUANTITY IN THE EQUATE, EXTSQT (OR              *
*       THE BYTE AT LABEL, EXTSQTY), SECONDARY QUANTITY               *
*       IS LEFT UNCHANGED.                                            *
*                                                                     *
*       IF SECONDARY QUANTITY IS ZERO, IT IS SET TO ONE HALF          *
*       OF PRIMARY QUANTITY IF PRIMARY IS NOT ZERO OR ONE.            *
*       IF PRIMARY QUANTITY IS ZERO, THE SPACE TYPE IS SET TO TRACKS,*
*       AND SECONDARY QUANTITY IS SET TO THE VALUE IN THE FULLWORD    *
*       AT LABEL SQTYDFLT (EQUAL TO THE EQUATE, SQTYDFL).             *
*       IF PRIMARY QUANTITY IS ONE, SECONDARY QUANTITY IS SET TO      *
*       VALUE IN THE FULLWORD AT LABEL SQTYDFLT (EQUAL TO THE         *
*       EQUATE, SQTYDFL).                                             *
*                                                                     *
* NOTES = SEE BELOW                                                   *
*                                                                     *
*    DEPENDENCIES =                                                   *
*            CLASS ONE CHARACTER CODE.  THE EBCDIC CHARACTER CODE     *
*            WAS USED FOR ASSEMBLY.  THE MODULE MUST BE REASSEMBLED   *
*            IF A DIFFERENT CHARACTER SET IS USED FOR EXECUTION.      *
*                                                                     *
*    RESTRICTIONS = NONE                                              *
*                                                                     *
*    REGISTER CONVENTIONS =                                           *
*            R1  OIEXL ADDRESS                                        *
*            R2  DCB ADDRESS                                          *
*            R3  UCB ADDRESS                                          *
*            R4  DCB BLOCK SIZE                                       *
*            R5  ADDRESS OF TPMAX OR DAMAX TABLES                     *
*            R6  EVEN REGISTER OF EVEN/ODD PAIR                       *
*            R7  ODD REGISTER OF EVEN/ODD PAIR                        *
*            R8  TIOT ENTRY ADDRESS                                   *
*            R8  JFCB ADDRESS                                         *
*            R10 FORMAT 1 DSCB ADDRESS                                *
*            R11 SAVE RETURN CODE                                     *
*            R13 SAVE AREA ADDRESS                                    *
*            R14 RETURN ADDRESS                                       *
*            R15 BASE REGISTER                                        *
```

```
*                                                                     *
*    PATCH LABEL = PATCH                                              *
*                                                                     *
* MODULE TYPE = CONTROL (OPEN, CLOSE, EOV DATA MANAGEMENT)            *
*                                                                     *
*    PROCESSOR = ASSEMBLER XF                                         *
*                                                                     *
*    MODULE SIZE = SEE EXTERNAL SYMBOL DICTIONARY                     *
*                                                                     *
*    ATTRIBUTES = REENTRANT, REFRESHABLE,READ-ONLY, ENABLED,          *
*                 PRIVILEGED, SUPERVISOR STATE, KEY ZERO,             *
*                 LINK PACK AREA RESIDENT/PAGEABLE                    *
*                                                                     *
* ENTRY POINT = IFG0EX0B                                              *
*                                                                     *
*    PURPOSE = SEE FUNCTION                                           *
*                                                                     *
*    LINKAGE =                                                        *
*        FROM IFG0196L:                                               *
*           BALR 14,15                                                *
*                                                                     *
* INPUT = STANDARD LINKAGE CONVENTIONS                                *
*                                                                     *
* OUTPUT =   DCBBUFNO DEFAULTED                                       *
*            PARTIAL RELEASE SET OR RESET                             *
*            CONTIGUOUS FLAG SET TO ZERO                              *
*            SECONDARY SPACE REQUEST MODIFIED                         *
*          RETURN CODE IN REGISTER 15                                 *
*            0 IF JFCB NOT MODIFIED                                   *
*            4 IF JFCB MODIFIED                                       *
*                                                                     *
* EXIT-NORMAL =                                                       *
*          BR 14                                                      *
*                                                                     *
* EXIT-ERROR =                                                        *
*        NONE                                                         *
*                                                                     *
* EXTERNAL REFERENCES = SEE BELOW                                     *
*                                                                     *
*    ROUTINES = NONE                                                  *
*                                                                     *
*    DATA AREAS = NONE                                                *
*                                                                     *
*    CONTROL BLOCK = NONE                                             *
*                                                                     *
* TABLES = NONE                                                       *
*                                                                     *
* MACROS = MODESET, IECOIEXL, DCBD, IEFUCBOB, IEFTIOT1, IEFJFCBN,     *
*          IECSDSL1                                                   *
*                                                                     *
***********************************************************************
```

```
*********************************************************************
*
*        REGISTER EQUATES
*
*********************************************************************
R1       EQU   1                  OIEXL PARAMETER LIST ADDRESS
RDCB     EQU   2                  DCB ADDRESS
RUCB     EQU   3                  UCB ADDRESS
RBKSIZ   EQU   4                  DCB BLOCK SIZE
RMAX     EQU   5                  ADDRESS OF TPMAX OR DAMAX
REVEN    EQU   6                  EVEN REGISTER OF EVEN/ODD PAIR
RODD     EQU   7                  ODD REGISTER OF EVEN/ODD PAIR. HAS   *
                                  DCBBUFNO DEFAULT
RTIOT    EQU   8                  TIOT ENTRY ADDRESS
RJFCB    EQU   9                  JFCB ADDRESS
RDSCB    EQU   10                 FORMAT 1 DSCB ADDRESS
RINCODE  EQU   11                 INTERNAL RETURN CODE
R12      EQU   12
RSAVE    EQU   13                 SAVE AREA ADDRESS
RET      EQU   14                 RETURN ADDRESS
RCODE    EQU   15                 BASE REGISTER/RETURN CODE ON EXIT
*********************************************************************
*
*        RETURN CODE
*
*********************************************************************
MODJFCB  EQU   4                  RETURN CODE IF JFCB MODIFIED
         USING IFG0EX0B,RCODE
*********************************************************************
*
*        START OF SAMPLE PROGRAM
*
*********************************************************************
         B     AFTRID1
         DC    C'IFG0EX0B JDM1137 &SYSDATE'
+        DC    C'IFG0EX0B JDM1137 05/01/81'
 AFTRID1 SAVE  (14,12)            SAVE REGISTERS
+AFTRID1 DS    0H
+        STM   14,12,12(13)                SAVE REGISTERS
         XR    RINCODE,RINCODE    ZERO RETURN CODE
         USING OIEXL,R1           PARAMETER LIST
         BAL   RET,BUFNO          DEFAULT BUFNO
         BAL   RET,SCREEN         SCREEN OUT CASES WHERE RLSE, *
                                  AND SQTY SHOULD NOT BE CALLED
         BAL   RET,RLSE           SET PARTIAL RELEASE
         BAL   RET,SQTY           SET SECONDARY QUANTITY
 EXIT    EQU   *                  RETURN TO CALLER
*********************************************************************
*        RETURN TO CALLER
*********************************************************************
         LR    RCODE,RINCODE
         RETURN (14,12),RC=(15)   RESTORE REGISTER
+        L     14,12(13,0)                 RESTORE REGISTER 14
+        LM    0,12,20(13)                 RESTORE THE REGISTERS
+        BR    14                          RETURN
 BUFNO   EQU   *                  DEFAULT DCB BUFNO
```

```
*********************************************************************
*
*        DEFINE DEFAULT VALUES
*          DAMXK   = NUMBER OF K (1024) OF BUFFERS FOR DASD
*          TPMXK   = NUMBER OF K (1024) OF BUFFERS FOR TAPE
*          DAMXBNO = MAXIMUM NUMBER OF BUFFERS FOR DASD
*          TPMXBNO = MAXIMUM NUMBER OF BUFFERS FOR TAPE
*        NOTE THAT DAMXBNO AND TPMXBNO MUST NOT BE GREATER THAN 255
*
*********************************************************************
DAMXK    EQU   64                64K BUFFERS FOR DASD
TPMXK    EQU   64                64K BUFFERS FOR TAPE
DAMXBNO  EQU   32                32 BUFFERS MAXIMUM FOR DASD
TPMXBNO  EQU   32                32 BUFFERS MAXIMUM FOR TAPE
INOUTBNO EQU   1                 DCBBUFNO DEFAULT FOR SYSIN, SYSOUT,  *
                                 AND DD DUMMY
ONEK     EQU   10                SHIFT ARGUMENT TO MULTIPLY BY 1024
         B     AFTRID2
         DC    CL8'BUFNO'        BUFNO ROUTINE ID
AFTRID2  BCR   0,RET             NOP RETURN
         L     RDCB,OIEXPDCB     PROTECTED COPY OF DCB
         USING DCBD,RDCB
*********************************************************************
*        DO NOT PROCESS EXCP, BSAM, DSORG NOT PS OR PO,
*                        DCBBUFNO SPECIFIED
*********************************************************************
         TM    DCBMACF1,DCBMRECP  EXCP DCB?
         BO    RETBUFNO           RETURN IF EXCP
         TM    DCBMACF1,DCBMRRD   READ MACRO
         BO    RETBUFNO           RETURN IF READ-NOT QSAM
         TM    DCBMACF2,DCBMRWRT  WRITE MACRO
         BO    RETBUFNO           RETURN IF WRITE-NOT QSAM
         TM    DCBDSRG1,DCBDSGPS+DCBDSGPO PS OR PO
         BZ    RETBUFNO           EXIT IF NOT PS OR PO
         CLI   DCBBUFNO,0         IS DCBBUFNO SPECIFIED
         BNE   RETBUFNO           RETURN IF DCBBUFNO SPECIFIED
*********************************************************************
*        DEFAULT DCBBUFNO TO 1 FOR SYSIN, SYSOUT, TERMINAL, DUMMY
*********************************************************************
         L     RTIOT,OIEXTIOT     TIOT ENTRY ADDRESS
         USING TIOENTRY,RTIOT
         L    RODD,INOUTBN        BUFNO DEFAULT FOR SYSIN/SYSOUT/      *
                                  DD DUMMY
         TM    TIOELINK,TIOESSDS+TIOTTERM SYSIN/SYSOUT OR TERMINAL
         BNZ   STORE              BRANCH IF SYSIN OR SYSOUT OR TERMINAL
         L     RJFCB,OIEXJFCB     JFCB ADDRESS
         USING INFMJFCB,RJFCB
         CLC   JFCBDSNM(L'NULLFILE),NULLFILE DUMMY DATA SET
         BE    STORE              BRANCH IF DUMMY
*********************************************************************
*        EXIT IF NO UCB ADDRESS OR BLOCK SIZE NOT POSITIVE
*********************************************************************
         L     RUCB,OIEXUCB       UCB ADDRESS
         LTR   RUCB,RUCB          ANY UCB?
         BZ    RETBUFNO           EXIT IF NO UCB
         LH    RBKSIZ,DCBBLKSI    DCB BLOCK SIZE
         LTR   RBKSIZ,RBKSIZ      ANY BLOCK SIZE?
         BNP   RETBUFNO           RETURN IF NO BLOCK SIZE
```

```
*********************************************************************
*        GET TAPE OR DASD MAX TABLE
*********************************************************************
         USING UCBOB,RUCB
         TM    UCBTBYT3,UCB3DACC  DASD UCB?
         LA    RMAX,DAMAX         MAX TABLE FOR DASD
         BO    CALC               BRANCH IF DASD
         TM    UCBTBYT3,UCB3TAPE  TAPE UCB?
         LA    RMAX,TPMAX         MAX TABLE FOR TAPE
         BZ    RETBUFNO           RETURN IF NOT DASD OR TAPE
CALC     EQU   *                  DEFAULT DCBBUFNO
*********************************************************************
*        CALCULATE DEFAULT BUFFER NUMBER
*********************************************************************
         USING MAX,RMAX
         XR    REVEN,REVEN        ZERO EVEN REG
         L     RODD,MAXBUF        MAXIMUM STORAGE FOR BUFFERS
         SLL   RODD,ONEK          SHIFT TO MULTIPLY BY 1024
         DR    REVEN,RBKSIZ       DIVIDE MAS BUFFER SPACE BY BKSI
         C     RODD,MAXBNO        ARE THERE TOO MANY BUFFERS?
         BNH   STORE              USE CALCULATION IF NOT TOO LARGE
         L     RODD,MAXBNO        USE MAXIMUM NUMBER OF BUFFERS
STORE    EQU   *                  DEFAULT DCBBUFNO FOR USER/COPY DCB
         STC   RODD,DCBBUFNO      PUT IN PROTECTED COPY OF DCB
         L     RDCB,OIEXUDCB      USER DCB
         XR    REVEN,REVEN        MODESET USES REG 6 = REVEN
         MODESET KEYADDR=OIEXUKEY,WORKREG=6 GET IN USER KEY
+* /* MACDATE Y-3 77277                                    @ZA26071*/
+* /*
+        IC    6,OIEXUKEY          GET KEY FROM SAVE LOCATION
+        SPKA  0(6)                SET PSW KEY
         STC   RODD,DCBBUFNO      PUT IN USER DCB
         MODESET EXTKEY=ZERO      BACK TO KEY ZERO
+* /* MACDATE Y-3 77277                                    @ZA26071*/
+* /*
+        SPKA  0(0)                SET PSW KEY
RETBUFNO EQU   *                  RETURN FROM BUFNO
         BR    RET                RETURN
INOUTBN  DC    A(INOUTBNO)        SYSIN/SYSOUT/DUMMY BUFNO DEFAULT
*********************************************************************
*
*        MAX TABLE FOR TAPE
*
*********************************************************************
         DS    0F
         DC    CL8'TPMAX'         TPMAX ID
TPMAX    DS    0F
TPMAXK   DC    A(TPMXK)           MAXIMUM SIZE FOR BUFFERS IN UNITS    *
                                  OF 1024
TPMAXBNO DC    A(TPMXBNO)         MAXIMUM NUMBER OF BUFFERS
```

```
*********************************************************************
*
*        MAX TABLE FOR DASD
*
*********************************************************************
         DS    0F
         DC    CL8'DAMAX'          DAMAX ID
DAMAX    DS    0F
DAMAXK   DC    A(DAMXK)            MAXIMUM SIZE FOR BUFFERS IN UNITS    *
                                   OF 1024
DAMAXBNO DC    A(DAMXBNO)          MAXIMUM NUMBER OF BUFFERS
SCREEN   EQU   *                   SCREEN OUT CASES WHERE RLSE,         *
                                   AND SQTY SHOULD NOT EXECUTE
*********************************************************************
*        DO NOT PROCESS IF
*           SYSIN/SYSOUT/TERMINAL
*           DD DUMMY
*           USER ASKS JFCB NOT BE RE-WRITTEN
*           SYSTEM DATA SET ('SYS1.XXX')
*           NON-DASD UCB
*           NOT A FORMAT 1 DSCB
*           EXCP DCB
*           DSORG IN DCB IS NEITHER PS NOR PO
*           DSORG IN DSCB IS NEITHER PS NOR PO
*           NEITHER PUT NOR WRITE MACRO CODED IN DCB
*           OPEN FOR OTHER THAN OUTPUT OR OUTIN
*********************************************************************
         B     AFTRID3
         DC    CL8'SCREEN'         SCREEN ROUTINE ID
AFTRID3  L     RTIOT,OIEXTIOT      TIOT ENTRY ADDRESS
         TM    TIOELINK,TIOESSDS+TIOTTERM SYSIN/SYSOUT OR TERMINAL
         BNZ   EXIT                EXIT IF SYSIN OR SYSOUT OR   TERMINAL
         L     RJFCB,OIEXJFCB      JFCB ADDRESS
         CLC   JFCBDSNM(L'NULLFILE),NULLFILE DUMMY DATA SET
         BE    EXIT                EXIT IF DUMMY
         CLC   SYS1,JFCBDSNM       SYS1.XXX DATA SET
         BE    EXIT                EXIT IF SYSTEM DATA SET
         TM    JFCBTSDM,JFCNWRIT    DON'T MODIFY JFCB
         BO    EXIT                EXIT IF YES
         L     RUCB,OIEXUCB        UCB ADDRESS
         LTR   RUCB,RUCB           ANY UCB?
         BZ    EXIT                EXIT IF NO UCB
         TM    UCBTBYT3,UCB3DACC   DASD UCB?
         BNO   EXIT                EXIT IF NOT DASD
         L     RDSCB,OIEXDSCB      FORMAT 1 DSCB ADDRESS
         USING DS1FMTID,RDSCB
         CLI   DS1FMTID,C'1'       IS THIS A FORMAT 1 DSCB
         BNE   EXIT                EXIT IF NOT
         L     RDCB,OIEXPDCB       PROTECTED DCB ADDRESS
         TM    DCBMACF1,DCBMRECP   EXCP DCB?
         BO    EXIT                EXIT IF EXCP
         TM    DCBDSRG1,DCBDSGPS+DCBDSGPO PS OR PO DCB
         BZ    EXIT                EXIT IF NOT PS OR PO
         NC    DS1DSORG,DS1DSORG   IS DSORG SPECIFIED
         BZ    TSTMACRF            TRUST DCB IF NOT SPECIFIED
         TM    DS1DSORG,DS1DSGPS+DS1DSGPO IS DATA SET PS OR PO
         BZ    EXIT                EXIT IF NOT PS OR PO
```

```
TSTMACRF EQU   *                 TEST MACRF IN DCB
         TM    DCBMACF2,DCBMRPUT  PUT MACRO
         BO    TSTOOPT            TEST OPEN OPTION
         TM    DCBMACF2,DCBMRWRT  WRITE MACRO
         BZ    EXIT               EXIT IF NOT WRITE
TSTOOPT  EQU   *                 TEST OPEN OPTION
         TM    OIEXOOPT,OIEXOOUT  OPEN FOR OUTPUT
         BO    SCREENOK           BRANCH IF YES
         TM    OIEXOOPT,OIEXOOIN  OPEN FOR OUTIN
         BNO   EXIT               EXIT IF NO
SCREENOK EQU   *
         BR    RET                RETURN TO CALL RLSE, SQTY
RLSE     EQU   *                  SET PARTIAL RELEASE
*********************************************************************
*
*        DEFINE DEFAULT VALUES
*          RLSE0   = NUMBER OF EXTENTS. IF THE DATA SET HAS THIS
*                    NUMBER OF EXTENTS OR MORE, THEN PARTIAL RELEASE
*                    WILL NOT BE ALLOWED.
*          RLSE1   = NUMBER OF EXTENTS. IF THE DATA SET HAS LESS THAN
*                    THIS NUMBER OF EXTENTS, PARTIAL RELEASE IS
*                    REQUIRED.
*
*        NOTE THAT RLSE0 MUST NOT BE GREATER THAN RLSE1
*
*        SETTING RLSE0 TO 17 OR GREATER WILL CAUSE THIS ROUTINE TO
*          NEVER PREVENT A REQUEST FOR PARTIAL RELEASE
*
*        SETTING RLSE1 TO 0 WILL CAUSE THIS ROUTINE TO
*          NEVER FORCE A REQUEST FOR PARTIAL RELEASE
*
*********************************************************************
RLSE0    EQU   8                  SET RELEASE BIT TO ZERO IF NUMBER OF *
                                  EXTENTS EQUAL OR GREATER THAN THIS
RLSE1    EQU   8                  SET RELEASE BIT TO ONE IF NUMBER OF  *
                                  EXTENTS LESS THAN THIS
         B     AFTRID4
         DC    CL8'RLSE'          RLSE ROUTINE ID
AFTRID4  BCR   0,RET              NOP RETURN
         L     RDSCB,OIEXDSCB     FORMAT 1 DSCB ADDRESS
         TM    DS1DSORG,DS1DSGPO  IS DATA SET PARTITIONED
         BO    TSTRLSE            DO NOT SET RELEASE FOR PARTITIONED
         CLC   DS1NOEPV,EXTRLSE1  FEW ENOUGH TO SET RELEASE
         BNL   TSTRLSE            BRANCH IF NOT
         L     RJFCB,OIEXJFCB
         OI    JFCBIND1,JFCRLSE   SET RELEASE
         LA    RINCODE,MODJFCB    JFCB MODIFIED
         B     RETRLSE            RETURN
TSTRLSE  CLC   DS1NOEPV,EXTRLSE0  ENOUGH TO ZERO RELEASE
         BL    RETRLSE            BRANCH IF NO
         NI    JFCBIND1,255-JFCRLSE ZERO RELEASE
         LA    RINCODE,MODJFCB    JFCB MODIFIED
RETRLSE  EQU   *                  RETURN FROM RLSE
         BR    RET                RETURN
         DC    CL8'RLSECONS'      RLSE CONSTANTS ID
         DS    0H
```

```
EXTRLSE1 DC    AL1(RLSE1)          IF FEWER THAN THIS NUMBER OF EXTENTS,*
                                   PARTIAL RELEASE WILL BE SET
EXTRLSE0 DC    AL1(RLSE0)          IF THIS NUMBER OR MORE EXTENTS,      *
                                   PARTIAL RELEASE WILL BE ZEROED
SQTY     EQU   *                   SET SECONDARY QUANTITY
*********************************************************************
*
*        DEFINE DEFAULT VALUES
*          SQTYDFL = DEFAULT SECONDARY QUANTITY. THIS QUANTITY IS
*                    SET IF THE SECONDARY QUANTITY IS ZERO AND THE
*                    PRIMARY QUANTITY IS ZERO OR ONE. IT IS USED
*                    IF SECONDARY QUANTITY IS ONE, AND THE NUMBER OF
*                    EXTENTS IS EQUAL OR GREATER TO EXTSQT.
*          EXTSQT  = NUMBER OF EXTENTS. IF THE DATA SET HAS THIS MANY
*                    EXTENTS OR MORE, THEN INCREASE SECONDARY QUANTITY.
*
*********************************************************************
SQTYDFL  EQU   5                   DEFAULT SECONDARY QUANTITY
EXTSQT   EQU   10                  IF DATA SET HAS THIS MANY EXTENTS,   *
                                   THEN INCREASE SECONDARY QUANTITY
         B     AFTRID6
         DC    CL8'SQTY'           SQTY ROUTINE ID
AFTRID6  BCR   0,RET               NOP RETURN
         L     RJFCB,OIEXJFCB      JFCB ADDRESS
         NC    JFCBSQTY,JFCBSQTY   ANY SECONDARY QUANTITY
         BZ    TSTPRIM             TEST PRIMARY IF NOT
         L     RDSCB,OIEXDSCB      FORMAT 1 DSCB ADDRESS
         CLC   DS1NOEPV,EXTSQTY    ENOUGH TO ADD TO SECONDARY QTY
         BL    RETSQTY             BRANCH IF NOT
         XR    RODD,RODD
         ICM   RODD,7,JFCBSQTY     GET SECONDARY QUANTITY
         LR    REVEN,RODD          SAVE IN REVEN
         SRL   REVEN,1             HALVE SECONDARY QUANTITY
         LTR   REVEN,REVEN         IS SECONDARY ONE
         BZ    SETDFLT             DEFAULT SECONDARY IF ONE
         AR    RODD,REVEN          150% OF SECONDARY
         B     STSQTY
TSTPRIM  EQU   *                   SECONDARY QUANTITY IS ZERO
         NC    JFCBPQTY,JFCBPQTY   IS PRIMARY QUANTITY ZERO
         BZ    DFLTSQTY            DEFAULT SECONDARY
         XR    RODD,RODD
         ICM   RODD,7,JFCBPQTY
         SRL   RODD,1              HALVE PRIMARY
         LTR   RODD,RODD           IS PRIMARY ONE
         BNZ   STSQTY              BRANCH IF NOT
SETDFLT  EQU   *                   USE QUANTITY IN SQTYDFLT
         L     RODD,SQTYDFLT       DEFAULT SECONDARY
         B     STSQTY              STORE SECONDARY
```

```
DFLTSQTY EQU   *                 PRIMARY AND SECONDARY ZERO
         L     RODD,SQTYDFLT     GET DEFAULT SECONDARY
         TM    JFCBCTRI,JFCBSPAC
         BNZ   STSQTY
         CLI   DS1EXT1,X'01'     TRACK EXTENT
         BE    DFLTTRK           YES -- SET TRACKS
         CLI   DS1EXT1,X'81'     CYL EXTENT
         BNE   RETSQTY           NO -- RETURN
         OI    JFCBCTRI,JFCBCYL  SET CYLINDER UNITS
         B     STSQTY
DFLTTRK  EQU   *                 SET TRACK UNITS
         OI    JFCBCTRI,JFCBTRK  MAKE TRACK REQUEST
STSQTY   EQU   *                 STORE SECONDARY QTY
         STCM  RODD,7,JFCBSQTY
         LA    RINCODE,MODJFCB   JFCB MODIFIED
RETSQTY  EQU   *                 RETURN FROM SQTY
         BR    RET               RETURN
         DS    0F
         DC    CL8'SQTYCONS'     SQTY ROUTINE CONSTANTS ID
SQTYDFLT DC    A(SQTYDFL)        DEFAULT SECONDARY QUANTITY
         DC    AL1(0)            NOTE ONE BYTE OF ZERO BEFORE EXTSQTY
EXTSQTY  DC    AL1(EXTSQT)       IF DATA SET HAS THIS MANY EXTENTS,   *
                                 THEN ADD TO SECONDARY QUANTITY
*********************************************************************
*
*        CONSTANTS / PATCH AREA
*
*********************************************************************
NULLFILE DC    C'NULLFILE '      DD DUMMY DATA SET NAME
SYS1     DC    C'SYS1.'          START OF SYSTEM DATA SET NAMES
         DS    0F
PATCH    DC    C'IFG0EX0B PATCH AREA '
         DC    XL50'00'
*********************************************************************
*
*        MAX TABLE MAPPING DSECT (MAPS TPMAX OR DAMAX)
*
*********************************************************************
MAX      DSECT
MAXBUF   DS    A                 MAXIMUM SIZE FOR BUFFERS
MAXBNO   DS    A                 MAXIMUM NUMBER OF BUFFERS
*********************************************************************
*
*        DCB OPEN INSTALLATION EXIT PARAMETER LIST
*        --THE IECOIEXL MACRO
*
*********************************************************************
         IECOIEXL
******** THE MACRO EXPANSION IS NOT SHOWN
```

```
*********************************************************************
*
*        DCB - THE DCBD MACRO
*
*********************************************************************
         DCBD DSORG=PS,DEVD=DA
******** THE MACRO EXPANSION IS NOT SHOWN
*********************************************************************
*
*        UCB - THE IEFUCBOB MACRO
*
*********************************************************************
UCB      DSECT
         IEFUCBOB LIST=YES
******** THE MACRO EXPANSION IS NOT SHOWN
*********************************************************************
*
*        TIOT - THE IEFTIOT1 MACRO
*
*********************************************************************
TIOT     DSECT
         IEFTIOT1
******** THE MACRO EXPANSION IS NOT SHOWN
*********************************************************************
*
*        JFCB - THE IEFJFCBN MACRO
*
*********************************************************************
JFCB     DSECT
         IEFJFCBN LIST=YES
******** THE MACRO EXPANSION IS NOT SHOWN
*********************************************************************
*
*        FORMAT 1 DSCB - THE IECSDSL1 MACRO
*
*********************************************************************
F1DSCB   DSECT
         IECSDSL1 (1)
******** THE MACRO EXPANSION IS NOT SHOWN
         END
```

Appendix B. SMS Indicators for DADSM Installation Exits

The format-1 and format-4 DSCBs have indicators that show the status of SMS-managed data sets.

In the format-1 DSCB, a 3-bit pattern at offset X'4E' from the beginning of the DSCB is used to show data set status:

Code Bit	Meaning
1...	SMS-managed data set
.1..	Uncataloged SMS-managed data set (The VTOC index is an uncataloged SMS-managed data set, as are all temporary data sets on SMS-managed volumes.)
..1.	Data set might be reblocked (That means the data set might be reblocked to a system-determined block size.)

A 3-byte field (secondary space extension) has been added at offset X'4F' from the beginning of the format-1 DSCB to reflect the AVGREC specification on the original allocation. The existence of this field is indicated by a flag at offset X'5E' from the beginning of the DSCB.

In the format-4 DSCB, a 2-bit pattern at offset X'3C' from the beginning of the format-4 DSCB is used to show the SMS-managed volume state:

Code Bit	Meaning
00..	Non-SMS-managed volume
01..	Initial SMS-managed volume
11..	Converted SMS-managed volume

This field was previously the last byte, unused, of a 5-byte pointer to a format-6 DSCB.

Appendix C. Read-Only Variables Referenced by ACS Installation Exits

IGDACERO

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
0 (0)	STRUCTURE	1472	ACERO	
	READ/ONLY VARIABLES PARAMETER LIST. CALLER SUPPLIES THESE VALUES. NOTE!!!!!!!! FIELDS NOT USED SHOULD BE SET TO BINARY ZEROES (X'00)			
	CONTROL BLOCK HEADER			
0 (0)	CHARACTER	8	ACEROID	CONTROL BLOCK ID=ACERO
8 (8)	SIGNED	2	ACEROLEN	LENGTH OF CONTROL BLOCK
10 (A)	SIGNED	2	ACEROVER	CONTROL BLOCK VERSION
	READ/ONLY VARIABLES FOLLOW			
12 (C)	SIGNED	4	ACEROSIZ	PRIMARY/ACTUAL SIZE OF DATA SET IN KB
16 (10)	SIGNED	4	ACEROMSZ	MAXIMUM SIZE OF DATA SET IN KB
20 (14)	CHARACTER	8	ACEROUNT	UNIT NAME
28 (1C)	CHARACTER	8	ACEROMVG	MSS VOLUME GROUP NAME
36 (24)	CHARACTER	8	ACEROAPP	APPLICATION ID (RACF)
44 (2C)	CHARACTER	8	ACERODSO	DATA SET OWNER (RACF)
52 (34)	CHARACTER	8	ACEROUSR	USERID
60 (3C)	CHARACTER	8	ACEROGRP	GROUPID
68 (44)	SIGNED	4	ACERODSG	DATA SET ORGANIZATION (MAY BE ACEROPS, ACEROPO, ACEROVS, ACERODA, ACERONUL)
72 (48)	SIGNED	4	ACERORCG	DATA SET RECORD ORGANIZATION (MAY BE ACEROKS, ACEROES, ACERORR, ACEROLS, ACERONUL)
76 (4C)	SIGNED	4	ACERODST	DATA SET TYPE (MAY BE ACEROGDS, ACEROPRM, ACEROTMP, ACERONUL)

80	(50)	SIGNED	4	ACEROXMD	EXECUTION MODE (MAY BE ACEROBCH, ACEROTSO, ACEROTSK, ACERONUL)
84	(54)	CHARACTER	8	ACEROJOB	JOB NAME
92	(5C)	CHARACTER	8	ACERODD	DD NAME
100	(64)	CHARACTER	8	ACEROPGM	PROGRAM NAME
					THE ACEROEXP AND ACERORTP FIELDS MAY BE SET IN THE FOLLOWING WAYS: 1) THE ACEROEXP AND ACERORTP FIELDS MAY BOTH BE NULL, 2) EITHER THE ACEROEXP OR ACERORTP FIELD MAY BE FILLED IN THE EXECUTOR WILL CALCULATE THE VALUE OF THE FIELD THAT WAS NOT PROVIDED, 3) THE ACEROEXP AND ACERORTP FIELDS MAY BOTH BE FILLED IN THE EXECUTOR WILL NOT CHECK THAT THE VALUES EXPIRATION DATE AND RETENTION PERIOD ARE CORRECTLY CALCULATED.
108	(6C)	CHARACTER	4	ACEROEXP	EXPIRATION DATE (YYYYDDD) IN PACKED DECIMAL
112	(70)	SIGNED	4	ACERORTP	RETENTION PERIOD DAYS
116	(74)	CHARACTER	128	*	RESERVED
					ENVIRONMENT WHERE INVOKED - MUST BE ACERORCL FOR RECALL, ACERORCV FOR RECOVER, ACEROCNV FOR CONVERT, ACEROALC FOR NEW ALLOCATIONS OR INSTALLATION EXIT MAY SET OWN VALUE.
244	(F4)	CHARACTER	8	ACEROENV	
					DEFAULT VALUES OF READ/WRITE VARIABLES
252	(FC)	CHARACTER	32	ACERODDC	DEFAULT DATA CLASS (FROM RACF)
252	(FC)	SIGNED	2	ACERODDL	LENGTH OF DEFAULT DATACLAS
254	(FE)	CHARACTER	30	ACERODDV	VALUE OF DEFAULT DATACLAS
284	(11C)	CHARACTER	32	ACERODSC	DEFAULT STORAGE CLASS (FROM RACF)
284	(11C)	SIGNED	2	ACERODSL	LENGTH OF DEFAULT STORCLAS
286	(11E)	CHARACTER	30	ACERODSV	VALUE OF DEFAULT STORCLAS
316	(13C)	CHARACTER	32	ACERODMC	DEFAULT MANAGEMENT CLASS (FROM RACF)
316	(13C)	SIGNED	2	ACERODML	LENGTH OF DEFAULT MGMTCLAS
318	(13E)	CHARACTER	30	ACERODMV	VALUE OF DEFAULT MGMTCLAS
348	(15C)	CHARACTER	80	*	RESERVED

	INPUT VALUES OF READ/WRITE VARIABLES				
428	(1AC)	CHARACTER	32	ACERODC	DATA CLASS INPUT ONLY. OUTPUT RETURNED IN IGDACERW
428	(1AC)	SIGNED	2	ACERODCL	LENGTH OF DATACLAS
430	(1AE)	CHARACTER	30	ACERODCV	VALUE OF DATACLAS
460	(1CC)	CHARACTER	32	ACEROSC	STORAGE CLASS INPUT ONLY. OUTPUT RETURNED IN IGDACERW
460	(1CC)	SIGNED	2	ACEROSCL	LENGTH OF STORCLAS
462	(1CE)	CHARACTER	30	ACEROSCV	VALUE OF STORCLAS
492	(1EC)	CHARACTER	32	ACEROMC	MANAGMENT CLASS INPUT ONLY. OUTPUT RETURNED IN IGDACERW
492	(1EC)	SIGNED	2	ACEROMCL	LENGTH OF MGMTCLAS
494	(1EE)	CHARACTER	30	ACEROMCV	VALUE OF MGMTCLAS
	DATA SET NAME				
524	(20C)	CHARACTER	44	ACERODSN	DATA SET NAME
568	(238)	CHARACTER	8	*	RESERVED
576	(240)	CHARACTER	4	*	RESERVED
	ACCT_JOB FIELD - JOB ACCOUNT INFORMATION IS IN THE FOLLOWING FORMAT: FIRST BYTE CONTAINS THE NUMBER OF FIELDS. ALL SUBSEQUENT ENTRIES CONTAIN THE LENGTH OF THE FIELD FOLLOWED BY FIELD, ETC.				
580	(244)	CHARACTER	257	ACEROJAC	
580	(244)	UNSIGNED	1	ACEROJNM	
581	(245)	CHARACTER	256	ACEROJFL	
837	(345)	CHARACTER	7	*	RESERVED
	ACCT_STEP FIELD - STEP ACCOUNT INFORMATION IS IN THE FOLLOWING FORMAT: FIRST BYTE CONTAINS THE NUMBER OF FIELDS ALL SUBSEQUENT ENTRIES CONTAIN THE LENGTH OF THE FIELD FOLLOWED BY FIELD, ETC.				
844	(34C)	CHARACTER	257	ACEROSAC	
844	(34C)	UNSIGNED	1	ACEROSNM	
845	(34D)	CHARACTER	256	ACEROSFL	
1101	(44D)	CHARACTER	7	*	RESERVED
	VOLSER INFORMATION				
1108	(454)	SIGNED	2	ACERONVL	NUMBER OF VOLSERS
1110	(456)	CHARACTER	6	ACEROVOL (59)	ARRAY OF 6 BYTE VOLSERS
1464	(5B8)	CHARACTER	8	*	RESERVED FOR FUTURE EXTENSIONS

CONSTANTS				
LENGTH	TYPE	VALUE	NAME	DESCRIPTION
8	CHARACTER	'ACERO'	ACEROI	CONTROL BLOCK ID
2	DECIMAL	0	ACEROV	VERSION NUMBER
	CONSTANTS FOR THE FOLLOWING FIELDS IN ACERO: ACERODSG, ACERORCG, ACERODST, ACEROXMD, AND ACEROENV. NULL			
1	HEX	00	ACERONUL	NULL
	ACERODSG - DATA SET ORGANIZATION			
4	DECIMAL	1	ACEROPS	PS PHYSICAL SEQUENTIAL
4	DECIMAL	2	ACEROPO	PO PARTITIONED
4	DECIMAL	3	ACEROVS	VS VSAM ORGANIZATION

CONSTANTS				
LENGTH	TYPE	VALUE	NAME	DESCRIPTION
4	DECIMAL	4	ACERODA	DA BDAM ORGANIZATION
	ACERORCG - DATA SET RECORD ORGANIZATION			
4	DECIMAL	1	ACEROKS	KSDS VSAM CLUSTER
4	DECIMAL	2	ACEROES	ESDS VSAM ENTRY SEQUENCED
4	DECIMAL	3	ACERORR	RRDS VSAM RELATIVE RECORD
4	DECIMAL	4	ACEROLS	LINEAR SPACE
	ACERODST - DATA SET TYPE			
4	DECIMAL	1	ACEROGDS	ONE GENERATION DATA SET OF A GENERATION DATA GROUP
4	DECIMAL	2	ACEROPRM	STANDARD PERMANENT DATA SETS
4	DECIMAL	3	ACEROTMP	TEMPORARY DATA SETS
	ACEROXMD - EXECUTION MODE			
4	DECIMAL	1	ACEROBCH	BATCH EXECUTION MODE
4	DECIMAL	2	ACEROTSO	TSO EXECUTION MODE
4	DECIMAL	3	ACEROTSK	STARTED TASK

ACEROENV - ENVIRONMENT WHERE INVOKED				
8	CHARACTER	'RECALL'	ACERORCL	DATA SET RECALL OPERATIONS
8	CHARACTER	'RECOVER'	ACERORCV	DATA SET RECOVER OPERATIONS
8	CHARACTER	'CONVERT'	ACEROCNV	DATA SET CONVERT IN PLACE OPERATIONS
8	CHARACTER	'ALLOC'	ACEROALC	NEW DATA SET ALLOCATIONS (DEFAULT)

Appendix D. Read-Write Variables Set by ACS Installation Exits

IGDACERW

OFFSETS	TYPE	LENGTH	NAME	DESCRIPTION
0 (0)	STRUCTURE	1172	ACERW	
HEADER				
0 (0)	CHARACTER	8	ACERWID	CONTROL BLOCK ID=ACERW
8 (8)	SIGNED	2	ACERWLEN	LENGTH OF CONTROL BLOCK
10 (A)	SIGNED	2	ACERWVER	CONTROL BLOCK VERSION
EXECUTOR FILLS IN THE FOLLOWING FIELDS				
RETURN READ/WRITE VARIABLE(S)				
12 (C)	SIGNED	4	ACERWNCS	NUMBER OF CLASS SELECTION RETURN VARIABLES
16 (10)	CHARACTER	32	ACERWCSV (15)	CLASS SELECTION VARIABLE RETURNED
16 (10)	SIGNED	2	ACERWVLN	LENGTH OF VALUE
18 (12)	CHARACTER	30	ACERWVAL	VALUE RETURN VARIABLES
496 (1F0)	CHARACTER	2	*	RESERVED
MESSAGE AREA				
498 (1F2)	SIGNED	2	ACERWNMG	NUMBER OF MESSAGES
500 (1F4)	CHARACTER	110	ACERWMSG (6)	MESSAGES GENERATED BY EXECUTION OF ACS ROUTINE WRITE STATEMENTS
500 (1F4)	CHARACTER	110	ACERWTXT	TEXT OF MESSAGES
1160 (488)	CHARACTER	12	*	RESERVED

CONSTANTS

LENGTH	TYPE	VALUE	NAME	DESCRIPTION
8	CHARACTER	'ACERW'	ACERWI	CONTROL BLOCK ID
2	DECIMAL	0	ACERWV	VERSION NUMBER

Appendix E. Parameter List for ACS Installation Exits

IGDACSPM

OFFSETS		TYPE	LENGTH	NAME	DESCRIPTION
0	(0)	STRUCTURE	52	ACSPM	ACSPB IS THE BASING EXPR
0	(0)	CHARACTER	8	ACSPID	CONTROL BLOCK ID='IGDACSPM'
8	(8)	SIGNED	2	ACSPLEN	ACSP CONTROL BLOCK LENGTH
10	(A)	SIGNED	2	ACSPVER	CONTROL BLOCK VERSION
12	(C)	CHARACTER	8	*	RESERVED
20	(14)	ADDRESS	4	ACSPWORK	POINTER TO A WORK AREA FOR THE EXIT
24	(18)	SIGNED	4	ACSPWLEN	LENGTH OF WORK AREA
28	(1C)	ADDRESS	4	ACSPERO	POINTER TO READ ONLY VARIABLES MAPPED BY IGDACERO
32	(20)	ADDRESS	4	ACSPERW	POINTER TO READ/WRITE VARIABLES MAPPED BY IGDACERW
36	(24)	ADDRESS	4	ACSPACS	POINTER TO INTERFACE ROUTINE FOR INVOKING THE ACS ROUTINES LINKAGE IS VIA STANDARD MVS LINKAGE CONVENTIONS IF THIS FIELD IS ZERO, THEN NO ACS ROUTINE EXISTS FOR THE CURRENT CONSTRUCT.
40	(28)	CHARACTER	12	ACSPACSP	PARAMETERS FOR ACS ROUTINES
40	(28)	ADDRESS	4	ACSPAERO	POINTER TO READ ONLY VARIABLES INITIALLY SET TO ACSPERO.
44	(2C)	ADDRESS	4	ACSPAERW	POINTER TO READ/WRITE VARS, INITIALLY SET TO ACSPERW.
48	(30)	CHARACTER	4	ACSPATOK	TOKEN FOR USE BY ACS INTERFACE ROUTINE (DO NOT MODIFY)

CONSTANTS				
LENGTH	TYPE	VALUE	NAME	DESCRIPTION
2	DECIMAL	0	ACSVER	VERSION NUMBER
8	CHARACTER	'IGDACSPH'	ACSID	CONTROL BLOCK ID
RETURN CODES FROM SMS INSTALLATION EXIT				
4	DECIMAL	0	ACSPCOMP	EXIT COMPLETED SUCCESSFULLY
4	DECIMAL	4	ACSJERR	ERROR, FAIL THE JOB
4	DECIMAL	16	ACSEXERR	AN ERROR OCCURRED IN THE EXIT, DO NOT REINVOKE IT

Abbreviations

The following acronyms and abbreviations are defined as they are used in the MVS/DFP library. If you do not find the term or abbreviation you are looking for, see *Dictionary of Computing*, SC20-1699 (formerly published as *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems*, GC20-1699).

This glossary includes acronyms and abbreviations developed by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO). This material is reproduced from the *American National Dictionary for Information Processing*, copyright 1977 by the Computer and Business Equipment Manufacturers American National Standards Institute, 1430 Broadway, New York, New York 10018.

A

A. ANSI control code, in RECFM.

ABE. Abnormal end, in EROPT.

ABE. Abnormal-end appendage, an appendage of EXCP.

ABEND. Abnormal end, macro instruction.

ACB. Access method control block.

ACC. Accept erroneous block, in EROPT.

ACS. Automatic class selection.

adcon. Address constant.

ADDR. Addressed processing or addressed.

ADR. Same as ADDR.

AIX. Alternate index.

AL. American National Standard Labels.

ANSI. American National Standards Institute.

APAR. Authorized program analysis report. A report of a problem caused by a suspected defect in a current unaltered release of a program.

ASCB. Address space control block.

ASCII. American National Standard Code for Information Interchange.

ASI. Asynchronous interrupt.

AUL. American National Standard user labels, in LABEL.

B

B. Blocked records, in RECFM.

BDAM. Basic direct access method.

BDW. Block descriptor word.

BISAM. Basic indexed sequential access method.

BLKSIZE. Block size, operand of DCB.

BPAM. Basic partitioned access method.

BPI. Bytes per inch.

BSAM. Basic sequential access method.

BSP. Backspace one block, macro instruction.

BUFC. Buffer control block.

BUFCB. Buffer pool control block, operand of DCB.

BUFL. Buffer length, operand of DCB.

BUFNO. Buffer number, operand of DCB.

BUFOFF. Buffer offset, operand of DCB.

C

C. Close.

CA. Control area.

CCW. Channel command word.

CDS. Control data set.

CI. Control interval. Also compatibility interface.

CNTRL. Control, macro instruction.

CONTIG. Contiguous space allocation, in SPACE.

CSECT. Control section.

CSW. Channel status word.

CTAP. ISMF Command Table—Application Table.

CVAF. Common VTOC access facility.

CVOL. Control volume.

CVT. Communication vector table.

D

D. Format-D, (ISCII/ASCII variable-length) records, in RECFM.

DA. Direct access, in DEVD or DSORG.

DADSM. Direct access device space management.

DASD. Direct access storage device.

DB. ISCII/ASCII variable-length, blocked records, in RECFM.

DBS. ISCII/ASCII variable-length, blocked spanned records, in RECFM.

DCB. Data control block.

DCBD. Data control block dummy section, macro instruction.

DCS. DASD calculation services.

DD. Data definition (statement).

DDNAME. Data definition name.

DEB. Data extent block.

DECB. Data event control block.

DFDSS. Data Facility Data Set Services.

DFHSM. Data Facility Hierarchical Storage Manager.

DFSORT. Data Facility Sort.

DIR. Direct processing.

DISP. Data set disposition, parameter of DD statement.

DS. ISCII/ASCII variable-length, spanned records, in RECFM.

DSAB. Data set association block.

DSCB. Data set control block.

DSDR. Data set descriptor record.

DSECT. Dummy control section.

DSL. DEB save list.

DSNAME. Data set name.

DSORG. Data set organization, operand of DCB.

E

EBCDIC. Extended binary-coded decimal interchange code.

ECB. Event control block.

EOB. End-of-block.

EOD. End-of-data.

EODAD. End of data set exit routine address, operand of DCB.

EOF. End-of-file.

EOM. End-of-module.

EOV. End-of-volume.

EP. External procedure entry point.

EROPT. Error options, operand of DCB.

ERP. Error recovery procedure.

ESETL. End sequential retrieval, QISAM macro instruction.

ESTAE. Extended specify task abnormal exit.

EXCD. Exceptional conditions.

EXCP. Execute channel program, macro instruction.

EXLST. Exit list, operand of DCB.

Ext Proc. External procedure.

F

F. Fixed-length records, in RECFM.

FB. Fixed-length, blocked records, in RECFM.

FBS. Fixed-length, blocked, standard records, in RECFM.

FBT. Fixed-length, blocked records with track overflow option, in RECFM.

FCB. Forms control buffer.

FEOV. Force end-of-volume, macro instruction.

FIPS. Federal Information Processing Standard.

FS. Fixed-length, standard records, in RECFM.

G

GEN. Generic key search.

GL. GET macro, locate mode, in MACRF.

GM. GET macro, move mode, in MACRF.

GSR. Global shared resources.

GTF. Generalized trace facility.

H

H. DOS tapes with embedded checkpoint records, parameter of OPTCD.

HA. Home address.

I

ID. Identifier or identification.

II. ISAM Interface.

IICB. ISAM interface control block.

INOUT. Input then output, operand of OPEN.

I/O. Input/output.

IOB. Input/output block.

IOS. I/O supervisor.

IPL. Initial program load.

IS. Indexed sequential, in DSORG.

ISAM. Indexed sequential access method.

ISCII. International Standard Code for Information Interchange.

ISMF. Interactive Storage Management Facility.

ISO. International Organization for Standardization.

ISU. Indexed sequential unmovable, in DSORG.

J

JCL. Job control language.

JFCB. Job file control block.

JFCBE. Job file control block extension for 3800 printer.

JSCB. Job step control block.

K

K. Kilobyte (equals two to the tenth power bytes, or 1024).

KEYLEN. Key length, operand of DCB.

L

LPA. Link pack area.

LPALIB. Link pack area library.

LPAP. ISMF Line Operator Table—Application Table.

LRECL. Logical record length, operand of DCB.

LRI. Logical record interface.

LSR. Local shared resources.

M

M. Machine control code, in RECFM.

MACR. Macro reference.

MACRF. Macro instruction form, operand of DCB.

MBBCCHHR. Absolute disk address. (Module#, bin#, cylinder#, head#, record#).

MOD. Modify data set, in DISP.

MOD. Module.

MSS. Mass Storage System.

N

n. Integer number.

NCP. Number of channel programs, operand of DCB.

NIP. Nucleus initialization program.

NL. No label, in LABEL.

NSI. Next sequential instruction.

NSL. Nonstandard label, in LABEL.

NUP. No update.

O

O. Open.

O/C/EOV. Open/close/end-of-volume.

OFLG. Open flags.

OPTCD. Optional services code, operand of DCB.

OS CVOL. Operating system control volume.

OS/VS. Operating system/virtual storage.

OUTIN. Output then input, operand of OPEN.

P

PCI. Program-controlled interruption.

PDAB. Parallel data access block.

PDS. Partitioned data set.

PDSCB. Partial data set control block.

PL. PUT macro, locate mode, in MACRF.

PLH. Placeholder list.

PM. PUT macro, move mode, in MACRF.

PO. Partitioned organization, in DSORG.

POU. Partitioned organization unmovable, in DSORG.

PR. Pseudo register.

PROC. Procedure.

PS. Physical sequential, in DSORG.

PSL. Page save list.

PSU. Physical sequential unmovable, in DSORG.

PSW. Program status word.

PTF. Program temporary fix.

Q

QISAM. Queued indexed sequential access method.

QSAM. Queued sequential access method.

R

RACF. Resource Access Control Facility.

RB. Request block.

RBA. Relative byte address.

RDBACK. Read backward, operand of OPEN.

RDW. Record descriptor word.

RECFM. Record format, operand of DCB.

RLSE. Release unused space, DD statement.

Rn. General purpose register n.

RPL. Request parameter list.

RPLE. Request parameter list extension.

RPS. Rotational position sensing.

RRDS. Relative record data set.

RTN. Routine.

R0. Record zero.

S

S. Standard format records, in RECFM.

SAM. Sequential access method.

SAMB. Sequential access method block.

SCRA. Catalog recovery area in system storage.

SEQ. Sequential or sequential processing.

SER. Volume serial number, in VOLUME.

SETL. Set lower limit of sequential retrieval

SF. Sequential forward, operand of READ or WRITE.

SIO. Start I/O.

SIO appendage. Start I/O appendage.

SK. Skip to a printer channel, operand of CNTRL.

SKP. Skip erroneous block, in EROPT.

SKP. Skip sequential or skip sequential processing.

SL. IBM standard labels, in LABEL.

SMF. System management facilities.

SMP. System Modification Program.

SMP/E. System Modification Program Extended.

SMS. Storage Management Subsystem.

SP. Space lines on a printer, operand of CNTRL.

SRA. Sphere record area.

SRB. Service request block.

SS. Select stacker on card reader, operand of CNTRL.

STRNO. Number of RPL strings.

SUL. IBM standard and user labels, in LABEL.

SVC. Supervisor call instruction.

SVCLIB. Supervisor call library.

SVRB. Supervisor request block.

SVT. Supervisor vector table.

SWA. Scheduler work area.

SYNAD. Synchronous error routine address, operand of DCB.

SYSCTLG. The data set name of the CVOL catalog.

SYSDUMP. System dump.

SYSIN. System input stream.

SYSOUT. System output stream.

T

T. Track overflow option, in RECFM; user-totaling, in OPTCD.

TCB. Task control block.

TIOT. Task I/O table.

TSO. Time sharing option.

TTR. Relative track record address.

U

U. Undefined length records, in RECFM.

UCB. Unit control block.

UHL. User header label.

UPD. Update mode, or data modify.

USVR. User security-verification routine.

UTL. User trailer label.

V

V. Format-V, variable-length) records, in RECFM.

VB. Variable-length, blocked records, in RECFM.

VBS. Variable-length, blocked, spanned records, in RECFM.

VICE. Volume index control entry.

VIO. Virtual input/output.

VIR. VTOC index record.

VIXM. VTOC index map.

VMDS. VTOC map of DSCBs.

VRP. VSAM resource pool.

VS. Variable-length, spanned records.

VSAM. Virtual storage access method.

VSI. VSAM shared information.

VSL. Virtual subarea list, same as PFL or PFPL.

VSM. Virtual storage manager.

VSRT. VSAM shared resource table.

VTAM. Virtual telecommunications access method.

VTOC. Volume table of contents.

VVDS. VSAM volume data set.

VVR. VSAM volume record.

W

WTO. Write to operator.

WTOR. Write to operator with reply.

X

XCTL. Transfer control.

Glossary

The following terms and phrases are defined as they are used in the MVS/DFP library. If you do not find the term or abbreviation you are looking for, see *Dictionary of Computing,* SC20-1699 (formerly published as *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems,* GC20-1699).

This glossary includes acronyms and abbreviations developed by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO). This material is reproduced from the *American National Dictionary for Information Processing,* copyright 1977 by the Computer and Business Equipment Manufacturers American National Standards Institute, 1430 Broadway, New York, New York 10018.

A

absolute address. An address that, without further modification, identifies a unique DASD storage location. The format is MBBCCHHR

access method control block (ACB). A control block that links an application program to VSAM or ACF/VTAM.

access method services. A multifunction service program that is used to manage both VSAM and non-VSAM data sets and integrated catalog facility or VSAM catalogs. It is used to define data sets and allocate space for them, convert indexed-sequential data sets to key-sequenced data sets, modify data set attributes in the catalog, reorganize data sets, facilitate data portability between operating systems, create backup copies of data sets, help make inaccessible data sets accessible, list the records of data sets and catalogs, define and build alternate indexes, and convert OS CVOLs and VSAM catalogs to integrated catalog facility catalogs.

acquire. In Mass Storage System, to allocate space on a staging drive, and to stage the volume table of contents (VTOC) from a cartridge to the staging drive.

ACS installation exit. User-written code, executed after an ACS routine, that provides capabilities beyond the scope of the ACS routine.

ACS interface routine. The routine that is used to invoke an ACS routine from an ACS installation exit routine.

ACS routine. A procedural set of ACS language statements. Based on a set of input variables, the ACS language statements generate the name of a predefined SMS class, or a list of names of predefined storage groups, for a data set.

actual extent. An area in the DEB containing data that describes the space occupied by an extent of a data set. BDAM module IGG0193A builds one actual extent for each extent in the data set.

addressed-direct access. In systems with VSAM, the retrieval or storage of a data record identified by its relative byte address, independent of the record's location relative to the previously retrieved or stored record. See also *keyed-direct access, addressed-sequential access,* and *keyed-sequential access.*

addressed-sequential address. In systems with VSAM, the retrieval or storage of a data record in its entry sequence relative to the previously retrieved or stored record. See also *keyed-sequential access, addressed-direct access,* and *keyed-direct access.*

alias. An alternative name an entry or for a member of a partitioned data set (PDS). In a CVOL catalog, only the high-level name of a fully qualified data set name may have an alias.

allocated space. All space allocated (on a device) to a data set.

allocated used space. The amount of allocated space that is in use.

alternate index (AIX). In systems with VSAM, a key-sequenced data set containing index entries organized by the alternate keys of its associated base data records. It provides an alternate means of locating records in the data component of a cluster on which the alternate index is based.

alternate index cluster. The data and index components of an alternate index.

alternate key. One or more characters within a data record used to identify the data record and control its use. Unlike the prime key, the alternate key can identify more than one data record. It is used to build an alternate index or to locate one or more base data records via an alternate index. See also *generic key, key,* and *key field.*

application. The use to which an access method is put or the end result that it serves; contrasted to the internal operation of the access method.

authorized program facility (APF). A facility that permits the identification of programs that are authorized to use restricted functions.

automatic class selection (ACS). A mechanism for assigning SMS classes and storage groups.

B

base cluster. A key-sequenced or entry-sequenced data set over which one or more alternate indexes are built.

base RBA. In VSAM, the relative byte address (RBA) stored in the header of an index record that is used to calculate the RBAs of data or index control intervals governed by the index record.

blkref field. A field the user specifies in a program and that contains either the relative or the actual address of the record the user wants access to. If it is the relative address, the BDAM address conversion routines convert it to an actual address (MBBCCHHR). The actual address is then placed in the IOBSEEK field of the IOB so that the channel program can use the address to find a block. The address of the blkref field is in the block address operand of the READ or WRITE macro.

block position feedback. A user-specified option that causes the system to put the actual or relative address of the block just read or written into the area specified in the block address operand of the READ or WRITE macro. The format of the address will be MBBCCHHR if feedback was not specified in the DCB macro; otherwise, the format will be the same as the addressing scheme in the DCB macro.

block unused. For non-VSAM data sets, block unused represents the amount of space (returned in kilobytes) that would be saved if the optimal block size were used instead of the the current block size. For VSAM data set, block unused represents the amount of space (returned in kilobytes) that would be saved if the optimal CI (control interval) size were used instead of the current CI size.

buffer pool. A continuous area of storage divided into buffers.

C

candidate volume. A direct-access storage volume that has been defined in a VSAM catalog as a VSAM volume; VSAM can automatically allocate space on this volume, as needed.

capacity record. The first block (block 0) on each track of a data set. It contains the ID of the last block on the track and the number of usable bytes remaining on the track.

catalog. A data set that contains extensive information required to locate other data sets, to allocate and deallocate storage space, to verify the access authority of a program or operator, and to accumulate data set usage statistics. See *master catalog* and *user catalog.*

catalog recovery area (CRA). An entry-sequenced data set that exists on each volume owned by a recoverable catalog, including the volume on which the catalog resides. The CRA contains copies of the catalog's records, and can be used to recover a damaged catalog.

cataloged data set. In a CVOL Catalog, a data set that is represented in an index or hierarchy of indexes that provides the means for locating the data set.

chained RPL. See *RPL string.*

class. See *SMS class.*

cluster. In VSAM, a named structure consisting of a group of related components. For example, when the data is key-sequenced, the cluster contains both the data and the index components; for data that is entry-sequenced, the cluster contains only a data component. See also *base cluster* and *alternate index cluster.*

collating sequence. An ordering assigned to a set of items, such that any two sets in that assigned order can be collated. As used in this publication, the order defined by the System/370 8-bit code for alphabetic, numeric, and special characters.

compendium. A compendium gathers together and presents in concise form all the essential facts and details about a VSAM functional unit.

component. A named, cataloged collection of stored records. A component, the lowest member of the hierarchy of data structures that can be cataloged, contains no named subsets.

control area (CA). A group of control intervals used as a unit for formatting a data set before adding records to it. Also, in a key-sequenced data set, the set of control intervals pointed to by a sequence-set index record; used by VSAM for distributing free space and for placing a sequence-set index record adjacent to its data.

control area split. The movement of the contents of some of the control intervals in a control area to a newly created control area, to facilitate the insertion or lengthening of a data record when there are no remaining free control intervals in the original control area.

control data set (CDS). With respect to SMS, a VSAM linear data set containing configurational, operational, or communication information. SMS introduces three types of control data sets: source control data set, active control data set, and communications data set.

control interval (CI). A fixed-length area of auxiliary storage space in which VSAM stores records. It is the unit of information transmitted to or from auxiliary storage by VSAM.

control interval access. The retrieval or storage of the contents of a control interval.

control interval definition field (CIDF). In VSAM, the four bytes at the end of a control interval that contains the displacement from the beginning of the control interval to the start of the free space and the length of the free space. If the length is 0, the displacement is to the beginning of the control information.

control interval split. The movement of some of the stored records in a control interval to a free control interval, to facilitate the insertion or lengthening of a record that won't fit in the original control interval.

control volume (CVOL). A volume that contains one or more indexes of the catalog.

cross memory. A synchronous method of communication between address spaces.

CVOL catalog. The collection of all data set indexes maintained by CVOL catalog management.

D

data class. A list of data set allocation parameters and their values, used when allocating a new SMS-managed data set.

data record. A collection of items of information from the standpoint of its use in an application, as a user supplies it to VSAM for storage.

data set. The major unit of data storage and retrieval in the operating system, consisting of data in a prescribed arrangement and described by control information to which the system has access. As used in this publication, a collection of fixed- or variable-length records in auxiliary storage, arranged by VSAM in key sequence or in entry sequence. See also *key-sequenced data set* and *entry-sequenced data set.*

data set application. ISMF is used to construct a list of data sets. Using this list, you can perform tasks against an individual data set or a group of data sets. These tasks include editing, browsing, recovering unused space, copying, migrating, deleting, backing up, and restoring data sets.

data set name. An identifier that clearly names a data set.

DASD calculation services (DCS). A subcomponent of MVS/DFP common services. DCS retrieves and calculates data set information for both VSAM and non-VSAM data sets based on the user's input request.

DEQ. An Assembler language macro instruction used to remove control of one or more serially reusable resources from the active task. It can also be used to determine whether control of the resource is currently assigned to or requested for the active task.

dequeue. To remove a request for a resource from a list of requests.

direct access. The retrieval or storage of data by a reference to its location in a data set rather than relative to the previously retrieved or stored data. See also *addressed-direct access* and *keyed-direct access.*

direct access storage device (DASD). A device in which the access time is effectively independent of the location of the data.

distributed free space. Space reserved within the control intervals of a key-sequenced data set for inserting new records into the data set in key sequence; also, whole control intervals reserved in a control area for the same purpose.

dummy record. A record, created when BSAM builds a BDAM data set containing format F records, whose purpose is to provide space in which new records can be added to the data set after it is created. The first byte in the key field of the dummy record contains X'FF', and the first byte in the data field has a value indicating the position of the dummy record on the track (the R in MBBCCHHR).

dynamic buffering. A user-specified option that requests that the system handle acquisition, assignment, and release of buffers.

E

ENQ. An Assembler language macro instruction that requests the control program to assign control of one or more serially reusable resources to the active task. It is also used to determine the status of a resource; that is, whether it is immediately available or in use, and whether control has been previously requested for the active task in another ENQ macro instruction.

enqueue. To build a list of requests for a named resource.

entry. A collection of information about a cataloged object in a master or user catalog. Each entry resides in one or more 512-byte records.

entry sequence. The order in which data records are physically arranged (according to ascending RBA) in

auxiliary storage, without respect to their contents. (Contrast with key sequence.)

entry-sequenced data set (ESDS). In VSAM, a data set whose records are loaded without respect to their contents, and whose RBAs cannot change. Records are retrieved and stored by addressed access, and new records are added at the end of the data set.

exclusive control. Preventing multiple WRITE-add requests from updating the same dummy record or writing over the same available space on a track. When specified by the user, exclusive control requests that the system prevent the data block about to be read from being modified by other requests; it is specified in a READ macro and released in a WRITE or RELEX macro. When a WRITE-add request is about to be processed, the system automatically gets exclusive control of either the data set or the track.

exclusive control list. An area of storage containing the UCB address and actual address of resources under exclusive control, and the addresses of the first and last IOBs for requests waiting to get exclusive control of that resource.

extended search. A user-specified option that requests that the system search for the specified block or a place in which to add a new block, starting with the first block on the track containing the block address operand specified in the request macro, and continuing either for as many tracks or blocks (rounded up to a complete track) as are specified in the request macro, or until the search ends successfully.

Extended search is only applicable if relative addressing is being used.

extended specify task abnormal exit (ESTAE). A task recovery routine that provides recovery for those programs that run enabled, unlocked, and in task mode.

extent. A continuous space on a DASD volume occupied by a data set or portion of a data set. An extent of a data set contains a whole number of control areas.

external procedure. A procedure that can be called by any other VSAM procedure; a procedure whose name is in the module's (assembler listing) "external symbol dictionary."

F

field. In a record or a control block, a specified area used for a particular category of data or control information.

free control interval pointer list. In a sequence-set index record, a vertical pointer that gives the location of a free control interval in the control area governed by the record.

FREEMAIN. An Assembler language macro instruction that releases one area of main storage that had previously been allocated to the job step as a result of a GETMAIN macro instruction.

free space. Space reserved within the control intervals of a key-sequenced data set for inserting new records into the data set in key sequence; also, whole control intervals reserved in a control area for the same purpose.

G

generation. One member of a generation data group.

generation data group (GDG). A collection of historically related non-VSAM data sets that are arranged in chronological order; each data set is known as a generation data set.

generation index. An index of the CVOL catalog that identifies the generations of a generation data group.

generic key. A high-order portion of a key, containing characters that identify those records that are significant for a certain application. For example, it might be desirable to retrieve all records whose keys begin with the generic key AB, regardless of the full key values.

GETMAIN. An Assembler language macro instruction that is used to allocate an area of main storage for use by the job step task.

global shared resources (GSR). An option for sharing I/O buffers, I/O-related control blocks, and channel programs among VSAM data sets in a resource pool that serves all address spaces in the system.

global storage. Virtual storage that is not part of a user's private address space.

H

header, index record. In an index record, the 24-byte field at the beginning of the record that contains control information about the record.

header entry. In a parameter list of GENCB, MODCB, SHOWCB, or TESTCB, the entry that identifies the type of request and control block and gives other general information about the request.

high-level name. The first component of a qualified data set name. This name is found in a volume index of the CVOL catalog.

horizontal extension. An extension record pointed to by a catalog record's extension field. See also *vertical extension*.

horizontal pointer. A pointer in an index record that contains the RBA of another index record in the same level that contains the next key in collating sequence; used for keyed-sequential access.

I

index. As used in this publication, an ordered collection of pairs, each consisting of a key and a pointer, used by VSAM to sequence and locate the records of a key-sequenced data set; organized in levels of index records. See also *index level*, *index set*, and *sequence set*.

index entry. A catalog entry that describes the index component of a key-sequenced cluster, alternate index, or catalog. An index entry contains the index component's attributes, passwords and protection attributes, allocation and extent information, and statistics.

index level. A set of index records that order and give the location of all the control intervals in the next lower level or in the data set that it controls.

index record. A collection of index entries that are retrieved and stored as a group. Contrast to data record.

index record header. In an index record, the 24-byte field at the beginning of the record that contains control information about the record.

index replication. The use of an entire track of direct access storage to contain as many copies of a single index record as possible; this reduces rotational delay.

index set. The set of index levels above the sequence set. The index set and the sequence set together comprise the index.

index upgrade. The process of reflecting changes made to a base cluster in its associated alternate indexes.

integrated catalog facility. The name of the catalog associated with the Data Facility Product program product.

Interactive Storage Management Facility (ISMF). An interactive DFP facility for defining and viewing the policy of how the Storage Management Subsystem manages storage.

Interactive System Productivity Facility (ISPF). An IBM licensed program used to develop, test, and run application programs interactively. ISPF is the interactive interface for all storage management functions.

internal procedure. A procedure that can be called only by other procedures within the module. See also *external procedure*.

ISAM interface. A set of routines that allow a processing program coded to use ISAM (indexed sequential access method) to gain access to a VSAM key-sequenced data set.

J

job catalog. A catalog made available for a job by means of the JOBCAT DD statement.

job control language (JCL). A problem-oriented language used to identify the job or describe its requirements to an operating system.

job step catalog. A catalog made available for a job by means of the STEPCAT DD statement.

K

key. One or more characters within an item of data that are used to identify it or control its use. As used in this publication, one or more consecutive characters taken from a data record, used to identify the record and establish its order with respect to other records.

key field. A field located in the same position in each record of a data set, whose contents are used for the key of a record.

key sequence. The collating sequence of data records, determined by the value of the key field in each of the data records. May be the same as, or different from, the entry sequence of the records.

key-sequenced data set (KSDS). A VSAM data set whose records are loaded in ascending key sequence and controlled by an index. Records are retrieved and stored by keyed access or by addressed access, and new records can be inserted in key sequence because of free space allocated in the data set. Relative byte addresses can change, because of control interval or control area splits.

keyed-direct access. The retrieval or storage of a data record by use of either an index that relates the record's key to its relative location in the data set or a relative record number, independent of the record's location relative to the previously retrieved or stored record. See also *addressed-direct access*, *keyed-sequential access*, and *addressed-sequential access*.

keyed-sequential access. The retrieval or storage of a data record in its key or relative record sequence relative to the previously retrieved or stored record, as defined by the sequence set of an index. See also *addressed-sequential access*, *keyed-direct access*, and *addressed-direct access*.

L

level. A conceptual relationship between indexes of the CVOL catalog. The index corresponding to the simple name of a data set is said to be the lowest level; the first component of a qualifier name is said to correspond to the highest-level index.

level number. For the index of a key-sequenced data set, a binary number in the header of an index record that indicates the index level to which the record belongs.

linear data set (LDS). A VSAM data set that contains data but no control information. A linear data set can be accessed as a byte-addressable string in virtual storage.

LINK. An Assembler language macro instruction that causes control to be passed to a specified entry point. The linkage relationship established is the same as that created by a BAL instruction.

local shared resources (LSR). An option for sharing I/O buffers, I/O-related control blocks, and channel programs among VSAM data sets in a resource pool that serves one partition or address space.

local storage. Virtual storage in a user's private address space.

locate. Pertains to functions that do not change the status of a catalog; that is, read-only operations are performed.

M

management class. A list of the migration, backup, and retention parameters and their values, for an SMS-managed data set.

mass sequential insertion. A technique VSAM uses for keyed sequential insertion of two or more records in sequence into a collating position in a data set: more efficient than inserting each record directly.

mass storage volume. Two data cartridges in the IBM 3850 Mass Storage System that contain information equivalent to what could be stored on a direct-access storage volume.

master catalog. A catalog that contains extensive data set and volume information that VSAM requires to locate data sets, to allocate and deallocate storage space, to verify the authorization of a program or operator to gain access to a data set, and to accumulate usage statistics for data sets.

memory. As used in this book, a synonym for the private address space in virtual storage.

module. The unit of code that is link-edited. A program module has at least one procedure, and may have many.

must-complete. An indication to the operating system that the event must be performed without interruption or waiting.

MVS/DFP. An IBM licensed program which is the base for the Storage Management Subsystem.

MVS/ESA. An MVS operating system environment which supports ESA/370.

N

next address feedback. A user-specified option that causes the system to put the relative address (TTR) of the next data or capacity record into the area specified in the next address operand of the READ or WRITE macro. .(If the *type* operand in the READ or WRITE macro terminated with an R, the address of the next data record is returned; if it terminated with an RU, the address of the next data or capacity record is returned, whichever occurs first.)

Next address feedback is only applicable for operations involving format VS records.

nonlocate. Pertains to functions that change the status of a catalog; that is, write operations are performed.

O

operating system. Software that controls the execution of programs; an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

optimal block size. For non-VSAM data sets, optimal block size represents the block size that would result in the greatest space utilization on a device, taking into consideration record length and device characteristics.

optimal CI size. For VSAM data sets, optimal CI size represents the control interval size that would result in the greatest space utilization on a device.

P

password. A unique string of characters stored in a catalog that a program, a computer operator, or a terminal user must supply to meet security requirements before a program gains access to a data set.

path. A named, logical entity composed of one or more clusters (an alternate index and its base cluster, for example).

PDS directory. The portion of a partitioned data set that provides a means of locating any of the members of the data set.

period. A group of tracks in which the first track does not begin with an overflow block, and the last track does not contain a block that overflows to another track.

physical record. A record whose characteristics depend on the manner or form in which it is stored, retrieved, or moved. A physical record may contain all or part of one or more logical records.

pointer. An address or other indication of location. For example, an RBA is a pointer that gives the relative location of a data record or a control interval in the data set to which it belongs.

preformat channel program. A channel program that writes a new format F record to an already existing data set.

prime index. The index component of a key-sequenced data set.

prime key. One or more characters within a data record used to identify the data record or control its use. A prime key must be unique.

procedure. A functional unit of VSAM code that is entered only at one entry point and exits at the end of the procedure (the last line of the procedure's code). The procedure can call (transfer control, with a return to the procedure expected) other procedures within the module (internal calls) and can call other procedures in other VSAM modules (external calls). See also *internal procedure* and *external procedure*.

processing program. Any program that is not a control program; synonymous with problem program.

program status word (PSW). An area in storage used to indicate the order in which instructions are executed, and to hold and indicate the status of the system.

program temporary fix (PTF). A temporary solution or bypass of a problem diagnosed by IBM Support Center as the result of a defect in a current unaltered release of the program.

Q

qualified name. A data set name consisting of a string of names separated by periods; for example, "TREE.FRUIT.APPLE" is a qualified name.

qualifier. Each component name in a qualified name other than the rightmost name. For example, "TREE" and "FRUIT" are qualifiers in "TREE.FRUIT.APPLE.."

queued sequential access method (QSAM). An extended version of the basic sequential access method (BSAM). Input data blocks awaiting processing or output data blocks awaiting transfer to auxiliary storage are queued on the system to minimize delays in I/O operations.

R

random access. See *direct access*.

read-only variable. An ACS language variable that contains data set or system-derived information. It can be referenced but not altered in an ACS routine.

read-write variable. An ACS language variable that is assigned a value within an ACS routine. It can be referenced, and each ACS routine assigns a value to its own unique read-write variable.

record. A set of data treated as a unit.

record definition field (RDF). A field stored as part of a stored record segment; it contains the control information required to manage stored record segments within a control interval.

record zero (R0). Track capacity record on a DASD device.

relative address. The position of a block in a data set relative to the first block of a data set. The relative address can be a relative track number or relative block number. See "relative track address" and "relative block address."

relative block address. A 3-byte binary number that indicates the position of a block in relation to the first block of a data set. The first block of a data set always has a relative block address of 0.

relative byte address (RBA). The displacement (expressed as a fullword binary integer) of a data record or a control interval from the beginning of the data set to which it belongs; independent of the manner in which the data set is stored.

relative extent. An area in the DEB containing the number of blocks in each extent and the number of blocks in each track (if track overflow is not in effect) of a data set. Module IGG0193A builds the relative extent area when relative block addressing is specified in the processing program.

relative record data set (RRDS). A data set whose records are loaded into fixed-length slots.

relative record number (RRN). A number that identifies not only the slot, or data space, in a relative record data set but also the record occupying the slot. Used as the key for keyed access to a relative record data set.

relative track and record address (TTR). Relative track and record address on a direct-access device, where TT represents two hexadecimal digits specifying the track relative to the beginning of the data set, and R is one hexadecimal digit specifying the record on that track.

replication. See *index replication*.

request parameter list (RPL). In VSAM, a control block that contains the information needed to process an I/O request.

resource. Any facility of the computing system or operating system required by a job or task, including main storage, input/output devices, the central processing unit, data sets, and control processing systems.

resource pool, VSAM. See *VSAM resource pool*.

RETURN. An Assembler language macro instruction that is used to return control to the calling CSECT, and to signal normal termination of the returning CSECT.

reusable data set. A VSAM data set that can be reused as a work file, regardless of its old contents. It must not be a base cluster of an alternate index.

ripple. Moving data from one block of a chain to the next, due to modification of data in a preceding block.

RPL string. A set of chained RPLs (the set may contain one or more RPLs) used to gain access to a VSAM data set by action macros (GET, PUT, etc). Two or more RPL strings may be used for concurrent direct or sequential requests made from a processing program or its subtasks.

S

SAVE. An Assembler language macro instruction that causes the contents of the specified registers to be stored in the save area at the address contained in register 13.

SCRATCH. An Assembler language macro instruction that points to the CAMLST macro instruction. SCRATCH, the first operand of CAMLST, specifies that a data set be deleted.

search argument. The field of a data block that contains information identifying the block as unique from any other block in the data set. Can be either the key field or the block ID in the count field. This term is also used to describe the string of keywords containing software failure symptom keywords.

search limit. The track following the last track that should actually be searched in a data set. The search limit is calculated and put in the IOBUPLIM field of the IOB when the DCB specifies the extended search option.

security. See *data security*.

segment. The portion of a spanned record contained within a control interval. See also *spanned record*.

sequence checking. The process of verifying the order of a set of records relative to some field's collating sequence.

sequence set. The lowest level of the index of a key-sequenced data set; it gives the locations of the control intervals in the data set and orders them by the key sequence of the data records they contain. The sequence set and the index set together comprise the index.

sequential access. The retrieval or storage of a data record in either its entry sequence, its key sequence, or its relative record number sequence, relative to the previously retrieved or stored record. See also *addressed-sequential access* and *keyed-sequential access*.

sequential access method (SAM). An access method for storing or retrieving data blocks in a continuous sequence, using either a sequential access or a direct access device.

shared resources. A set of functions that permit the sharing of a pool of I/O-related control blocks, channel programs, and buffers among several VSAM data sets open at the same time.

simple name. The rightmost component of a qualified name. For example, "APPLE" is the simple name in "TREE.FRUIT.APPLE." The simple name corresponds

to the lowest index level in the CVOL catalog for the data set name.

skip-sequential access. Keyed-sequential retrieval or storage of records here and there throughout a data set, skipping automatically to the desired record or collating position for insertion: VSAM scans the sequence set to find a record or a collating position. Valid for processing in ascending sequences only.

slot. For a relative record data set, the data area addressed by a relative record number which may contain a record or be empty.

SMS class. A list of attributes that SMS applies to data sets having similar allocation (data class), performance (storage class), or backup and retention (management class) needs.

SMS configuration. A configuration base, SMS class and storage group definitions, and ACS routines that SMS uses to manage storage.

SMS-managed data set. A data set that has been assigned a storage class.

spanned record. A logical record whose length exceeds control interval length, and as a result, crosses, or spans, one or more control interval boundaries within a single control area.

sphere. The collection of base cluster, alternate indexes, and upgrade alternate indexes opened to process one or more paths related to the same Base Information Block (BIB).

step catalog. A catalog made available for a step by means of the STEPCAT DD statement. See also *job step catalog*.

storage administrator. A person in the data processing installation who is responsible for defining, implementing, and maintaining storage management policies.

storage class. A list of DASD storage performance, security, and availability service level requirements for an SMS-managed data set.

storage group. A list of traits and characteristics that SMS applies to groups of storage volumes having similar migration, backup, and dump needs. Only the storage administrator can access storage group definitions.

Storage Management Subsystem (SMS). An operating environment that helps automate and centralize the management of storage. To manage storage, SMS provides the storage administrator with control over data class, storage class, management class, storage group, and ACS routine definitions.

stored record. A data record, together with its control information, as stored in auxiliary storage.

string. The part of a control block structure built around a placeholder (PLH) that enables VSAM to keep track of one position in the data set that the control block structure describes.

supervisor request block (SRB). A system control block containing program status information and general register contents.

system-managed storage. An approach to storage management in which the system determines data placement and an automatic data manager handles data backup, movement, space, and security.

T

terminal monitor program (TMP). In TSO, a program that accepts and interprets commands from the terminal, and causes the appropriate command processors to be scheduled and executed.

time sharing option (TSO). An optional configuration of the operating system that provides conversational time sharing from remote stations.

track overflow. A user-specified option that will allow a format F record whose space requirements exceed the space remaining on the track to be partially written on that track and completed on the next track.

tracks unused. For data sets specifying cylinder allocation, tracks unused represents the number of unused tracks (returned in kilobytes) over all cylinders allocated.

transaction ID. A number associated with each of several request parameter lists that define requests belonging to the same data transaction.

transfer control (XCTL). An Assembler language macro that causes control to be passed to a specified entry point.

true name. In a CVOL catalog, the high-level qualifier to which an alias is related.

U

uncatalog. To remove the catalog entry of a data set from a catalog.

unit control block (UCB). A data area used by MVS/ESA for device allocation and for controlling input/output, I/O) operations.

update channel program. A channel program that reads or writes data for purposes other than adding a new block to an existing data set.

update number. For a spanned record, a binary number in the second RDF of a record segment that indicates how many times the segments of a spanned record should be equal. An inequality indicates a possible error.

upgrade set. All the alternate indexes that VSAM has been instructed to update whenever there is a change to the data component of the base cluster.

user buffering. The use of a work area in the processing program's address space for an I/O buffer; VSAM transmits the contents of a control interval between the work area and direct access storage without intermediary buffering.

user catalog. An optional catalog used in the same way as the master catalog and pointed to by the master catalog. It also lessens the contention for the master catalog and facilitates volume portability.

V

vertical extension. An extension record pointed to by a set-of-fields pointer in the object's base catalog record or its horizontal extension. See also *base catalog record* and *horizontal extension.*

vertical pointer. A pointer in an index record of a given level that gives the location of an index record in the next lower level or the location of a control interval in the data set controlled by the index.

virtual storage access method (VSAM). An access method for direct or sequential processing of fixed- and variable-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry sequence), or by relative record number.

virtual telecommunications access method (VTAM). A set of programs that control communication between terminals and application programs running under VSE, OS/VS1, and OS/VS2.

volume application. Using a list of volumes constructed by ISMF, you can perform tasks against an individual volume. These tasks include consolidating or recovering unused space, copying, backing up, and restoring volumes.

volume index. The highest level of index in the CVOL catalog structure. Entries in the volume index point to all lower indexes and simple names.

volume index control entry (VICE). The first entry in the volume index. The VICE describes the volume index and controls space allocation in SYSCTLG.

VSAM catalog. A key-sequenced data set or file with an index containing extensive data set and volume information that VSAM requires to locate data sets or files, allocate and deallocate storage space, verify the authorization of a program or operator to gain access to a file, and accumulate usage statistics for data sets or files. VSAM catalogs have been functionally replaced by integrated catalog facility catalogs. VSAM catalogs are not supported by the Storage Management Subsystem.

VSAM resource pool. A virtual storage area that is used to share I/O buffers, I/O-related control blocks, and channel programs among VSAM data sets. A resource pool is local or global; it serves tasks in one partition or address space or tasks in all address spaces in the system.

VSAM shared information (VSI). Blocks that are used for cross-system sharing.

W

WAIT. An Assembler language macro instruction that informs the control program that the issuing program cannot continue until a specific event, represented by an event control block, has occurred.

WRITE-add request. A request to write a new block to the data set.

WRITE-release request. A WRITE-update request that specifies exclusive control should be released for the record about to be written.

WRITE-update request. A request to write an already existing block to the data set.

WRITE-validity check. A user-specified option that causes the system to verify the accuracy of any information written by the channel program.

Index

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Numerics

MVS/ESA
Data Facility Product Version 3:
Customization

SC26-4504-1

Reader's Comment Form

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

If you have applied any technical newsletters (TNLs) to this book, please list them here: _____

Comments (please include specific chapter and page references) :

Note: Staples can cause problems with automatic mail-sorting equipment.
Please use pressure-sensitive or other gummed tape to seal this form.

If you want a reply, please complete the following information:

Name _____ Date _____

Company _____ Phone No. (_____) _____

Address _____

_____

Thank you for your cooperation. No postage is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail them directly to the address in the Edition Notice on the back of the title page.)

SC26-4504-1

Reader's Comment Form

Fold and tape | Please do not staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department J57
P.O. Box 49023
San Jose, CA 95161-9945

Fold and tape | Please do not staple | Fold and tape

IBM®

IBM®

Program Number
5665-XA3

File Number
S370-30

SC26-4504-1

Printed in U.S.A.