z/OS
2.5

DFSMS Installation Exits

SC23-6850-50

Note

Before using this information and the product it supports, read the information in “Notices” on page 251.

This edition applies to Version 2 Release 5 of z/OS® (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2022-08-22

Contents

Figures

Tables

About This Document

This document helps you customize DFSMS with installation exit routines and modules that extend or replace IBM-supplied functions at your installation. Storage administrators and system programmers can use this information to centralize customization at their installations. All routines described reside in system libraries; installing them requires authority to update the system library.

This document does not cover all installation exits available in DFSMS. The following table indicates the documents that cover installation exits for the indicated components.

Component	Document
Network File System	*z/OS Network File System Guide and Reference*
DFSMSrmm	*z/OS DFSMSrmm Implementation and Customization Guide*
Object Access Method	*z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support*, *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Tape Libraries*, and *z/OS DFSMS OAM Application Programmer's Reference*.

When you install the z/OS system, initialization parameters and SMS options let you tailor the system to your requirements. How you tune z/OS can affect how you customize MVS™ and DFSMS.

Related reading:

- For additional information about initialization parameters, see the *z/OS MVS Initialization and Tuning Guide*.
- For information about the installation exits that MVS provides, see *z/OS MVS Installation Exits*.
- For information about customization at a system level, see *z/OS MVS Using the Subsystem Interface*.
- For information about SMS options, see *z/OS DFSMSdfp Storage Administration*.

For information about the accessibility features of z/OS, for users who have a physical disability, see Appendix A, “Accessibility,” on page 249.

Required product knowledge

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

- Assembler language
- Job control language
- Standard program linkage conventions
- Data management access methods and macro instructions.

Referenced documents

The following publications are referenced in this document. You can order these documents by order number.

Publication Title	Order Number
z/OS SMP/E Reference	SA23-2276
z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN	SA23-1372

Publication Title	Order Number
z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG	SA23-1373
z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU	SA23-1374
z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO	SA23-1375
z/OS MVS Diagnosis: Reference	GA32-0904
z/OS DFSMShsm Implementation and Customization Guide	SC23-6869
z/OS DFSMShsm Diagnosis	GC52-1387
z/OS Problem Management	G325-2564
z/OS MVS Initialization and Tuning Guide	SA23-1379
z/OS DFSMS Access Method Services Commands	SC23-6846
z/OS DFSMSdfp Diagnosis	SC23-6863
z/OS DFSMS Using Data Sets	SC23-6855
z/OS DFSMSdfp Storage Administration	SC23-6860
z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support	SC23-6866
z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Tape Libraries	SC23-6867
z/OS DFSMS OAM Application Programmer's Reference	SC23-6865
z/OS DFSMSdfp Advanced Services	SC23-6861
z/OS DFSMS Using Magnetic Tapes	SC23-6858
z/OS DFSMSdfp Utilities	SC23-6864
z/OS Security Server RACF System Programmer's Guide	SA23-2287
z/OS MVS Using the Subsystem Interface	SA38-0679
z/OS MVS System Messages, Vol 1 (ABA-AOM)	SA38-0668
z/OS MVS System Messages, Vol 2 (ARC-ASA)	SA38-0669
z/OS MVS System Messages, Vol 3 (ASB-BPX)	SA38-0670
z/OS MVS System Messages, Vol 4 (CBD-DMO)	SA38-0671
z/OS MVS System Messages, Vol 5 (EDG-GLZ)	SA38-0672
z/OS MVS System Messages, Vol 6 (GOS-IEA)	SA38-0673
z/OS MVS System Messages, Vol 7 (IEB-IEE)	SA38-0674
z/OS MVS System Messages, Vol 8 (IEF-IGD)	SA38-0675
z/OS MVS System Messages, Vol 9 (IGF-IWM)	SA38-0676
z/OS MVS System Messages, Vol 10 (IXC-IZP)	SA38-0677
z/OS MVS Installation Exits	SA23-1381
z/OS MVS Diagnosis: Tools and Service Aids	GA32-0905

z/OS information

This information explains how z/OS references information in other documents and on the web.

When possible, this information uses cross document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see *z/OS Information Roadmap*.

To find the complete z/OS library, go to IBM Documentation (www.ibm.com/docs/en/zos).

How to send your comments to IBM

We invite you to submit comments about the z/OS product documentation. Your valuable feedback helps to ensure accurate and high-quality information.

Important: If your comment regards a technical question or problem, see instead "If you have a technical problem" on page xxi.

Submit your feedback by using the appropriate method for your type of comment or question:

Feedback on z/OS function
If your comment or question is about z/OS itself, submit a request through the IBM RFE Community (www.ibm.com/developerworks/rfe/).

Feedback on IBM® Documentation function
If your comment or question is about the IBM Documentation functionality, for example search capabilities or how to arrange the browser view, send a detailed email to IBM Documentation Support at ibmdocs@us.ibm.com.

Feedback on the z/OS product documentation and content
If your comment is about the information that is provided in the z/OS product documentation library, send a detailed email to mhvrcfs@us.ibm.com. We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

To help us better process your submission, include the following information:

- Your name, company/university/institution name, and email address
- The following deliverable title and order number: z/OS DFSMS Installation Exits, SC23-6850-50
- The section title of the specific information to which your comment relates
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive authority to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

If you have a technical problem or question, do not use the feedback methods that are provided for sending documentation comments. Instead, take one or more of the following actions:

- Go to the IBM Support Portal (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Note: IBM z/OS policy for the integration of service information into the z/OS product documentation library is documented on the z/OS Internet Library under IBM z/OS Product Documentation Update Policy (www-01.ibm.com/servers/resourcelink/svc00100.nsf/pages/ibm-zos-doc-update-policy?OpenDocument).

Summary of changes for z/OS Version 2 Release 5 (V2R5)

New

The following content is new.

August 2022 Refresh

In support of APAR OA62261, new information is added to "IGBDCSX1 and IGBDCSX2 Parameter List" on page 22 and Table 7 on page 22. (APAR OA62261, which also applies to z/OS V2R4)

Prior to August 2022 Refresh

- For APAR OA61850, DFSMS OPEN, STOW, and CLOSE SVC installation exits are added. For more information see:
  - Chapter 2, "Data Management Installation/Dynamic Exits," on page 7
  - "OPEN/CLOSE/EOV and access method SVC exits" on page 61

Changed

The following content is changed.

August 2022 Refresh

- "Adding a New Exit" on page 3 is updated.
- Chapter 4, "Pre-ACS Installation Exit IGDACSDX," on page 137 is updated.

Summary of changes for z/OS Version 2 Release 4 (V2R4)

Changed

The following content is changed.

- The description for the 24 (X'18') offset in "IFG0EX0B Parameter List" on page 42 was updated.
- A typographical error in "File End on Volume Exit Function-Specific Parameter List" on page 100 was corrected. In the table, UCBFSC should have been UCBFSCT.

Summary of changes for z/OS V2R3

The following changes are made for z/OS Version 2 Release 3 (V2R3). The most recent updates are listed at the top of each section.

Changed

- New UFOFCTXRC field in ADRUFO parameter list. See "ADRUFO Parameter List" on page 229.

- Added information about encrypted-format data sets in “Installation-Supplied ADRREBLK Exit Routine” on page 237.
- Added information about identifying new parameter list fields in “Open, Close, and EOV Main Parameter List” on page 75, and new fields at offset 36 (X'24).
- Details about the handling of return codes for multiple exit routines are updated in “Controlling the label anomaly exit routine through the dynamic exits facility” on page 88, “Controlling the volume mount exit routine through the dynamic exits facility” on page 92, “Controlling the file validate exit routine through the dynamic exits facility” on page 96, “Controlling the file start exit routine through the dynamic exits facility” on page 98, and “Controlling the file end exit routine through the dynamic exits facility” on page 101
- Added recording formats EFMT4 and EEFMT4 and media types MEDIA11, MEDIA12, and MEDIA13 to “Open, Close, and EOV Main Parameter List” on page 75.
- New UFFUSREL field and related updates for the SPACEREL operation. See “ADRUFO Parameter List” on page 229.
- Added fields to support cloud storage in “ADRUFO Parameter List” on page 229 and “ARCMDEXT—Input Data Structure” on page 175.

Deleted

No content was removed from this information.

Chapter 1. Introduction

DFSMS installation exits allow you to extend the capabilities of DFSMS to customize your installation's z/OS operation. You can also create standard user exits that your application programmers can use from their programs.

Choosing between Installation and User Exits

You might decide to customize DFSMS to:

- Enforce your standards.
- Intercept errors for analysis and additional processing.
- Add specialized tape label processing.
- Tailor I/O processing.
- Extend security controls.
- Change or bypass processing.

There are two ways to customize with exits in DFSMS. Choose an installation exit if the areas you need to customize affect every user of a function. These exits affect the entire installation's processing. Or, choose a user exit if the application programmers should have the opportunity to bypass the modification or to use their own. Whichever exit you choose, consider the impact your modifications have on users especially because some changes can only be made with installation exits.

DFSMS is a licensed program and can be modified for your own use only. IBM supports and maintains only unmodified IBM-supplied modules.

Using Installation Exits

DFSMS provides replaceable modules or exit points for installation exit routines that modify DFSMS system functions. A replaceable module is an IBM-supplied module that you can modify or replace.

Following are the types of replaceable modules or exit locations that are defined as installation exits:

- Dummy modules. The IBM-supplied module does not perform a useful function, it just supplies a return code to its caller. For example, the DADSM exit routines are dummy modules.
- Functional modules. The IBM-supplied module already performs a useful function. Although you can replace these modules, carefully consider whether you should preserve the IBM-supplied functions. For example, the data management abend installation exit (IFG0199I) is a functional module.
- No module. IBM does not supply a module, but does supply an exit point so that you can create an exit routine if you desire. If you do not supply a routine, the exit is not taken. For example, the nonstandard tape label processing exit points do not come with a replaceable module.

Any modules you create for installation exits must be reentrant and refreshable. You can install them during system installation by using SMP/E or you can link-edit the module into the appropriate library.

Installation exits are available at a variety of points in DFSMS processing. The sections for each exit documented in this information specify where in the system's processes the exits are called.

Using User Exits

DFSMS provides user exit locations as part of macros and commands where you can specify the name or address of your user-written exit routine. The DCB macro, VSAM macros, and some access method 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.

You can create a library of proven user exits to provide standard functions for frequently used exits; however, each user program must request the exits. Application programmers can create their own exit routines instead of the ones you provide.

User exit routines do not need to be reentrant.

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
- Open, close, and end-of-volume processing
- DESERV macro processing

For more information on the user exits available in DFSMS, see the following publications.

Document	Order Number
z/OS DFSMS Access Method Services Commands	SC23-6846
z/OS MVS Program Management: Advanced Facilities	SA23-1392
z/OS DFSMS Using Data Sets	SC23-6855
z/OS DFSMSdfp Utilities	SC23-6864
z/OS DFSMSrmm Implementation and Customization Guide	SC23-6874

The DESERV user exit is described in *z/OS DFSMSdfp Advanced Services*.

Programming Considerations

Most requirements for coding vary depending on the part of DFSMS you customize. Be aware of these:

- The examples in this information are written in assembler language and you can also create exits in assembler.
- At entry to your exit routine, save all registers and restore them before you return to your calling routine. The section for each exit specifies any registers that you must return with special contents. For example, frequently you must supply a return code in register 15 upon returning to DFSMS processing.
- If you replace a module, make sure you thoroughly test it before you make it available.
- Your routine should be reentrant so it can handle concurrent requests.
- Keep an unmodified copy of any replaceable modules you modify.

Installing Exits

There are two types of installation exits:

- An IBM-supplied routine which the system programmer replaces, usually a dummy routine
- A new routine that the system programmer adds to the system

In this information, the term *installation* refers to one or more computing systems, the work the systems perform, and the people who manage, operate, and service the systems, apply the systems to problems, and use the output they produce.

The term *installation exit* refers to a routine which the system programmer adds or replaces in the operating system. It does not refer to the process of installing the operating system.

If you install an installation exit, you should use SMP/E. You can do that before or after the product is installed. After following these steps, do the APPLY or ACCEPT or both steps. For additional information see *z/OS SMP/E Reference*.

Replacing an Existing Exit

The following shows SMP/E system modification statements. Follow these steps to replace a module which was supplied by IBM or by the installation.

```
++USERMOD(changename)   /* Replacement of existing module */
  .
++VER(Z038) FMID(fmid) PRE(xxxxxxx) SUP(yyyyyyy)
  .
++SRC(modname) SYSLIB(SUSERLIB) DISTLIB(ASRCLIB)
  .
```

Figure 1. Replacing an Existing Exit

Explanation of Figure 1 on page 3:

1. Replace *changename* with a name to identify the system change. For example, you might select the name UPDATE5.

   ```
   ++USERMOD(changename)   /* Replacement of existing module */
   ```

2. In place of fmid, use the actual FMID (function modification identifier). You can find this in the report generated by the SMP/E list command LIST SYSMODS. When you first install this exit, you probably will omit the PRE(*xxxxxxx*) SUP(*yyyyyyy*) because there probably is no prerequisite or superseding. If you later replace your own exit, you will code a different *changename* on the ++USERMOD statement and use the old *changename* with the SUP keyword.

   ```
   ++VER(Z038) FMID(fmid) PRE(xxxxxxx) SUP(yyyyyyy)
   ```

3. Replace *exitname* with the name of the exit. For some exits, the entry point name must match the exit name. The control section (CSECT) name should be the same name unless you choose a different name and code that name with the CSECT keyword. SYSLIB points to the target source library which contains the source code. You can omit the DISTLIB keyword and value unless stated otherwise in this information. It is only needed when SMP/E has copied the existing routine into the system while copying an entire library.

   ```
   ++SRC(exitname) SYSLIB(SUSERLIB) DISTLIB(ASRCLIB)
   ```

4. Place the source code after the ++SRC statement.

   ```
   4.   (source code)
   ```

   If you prefer, you can code the equivalent of a sequential concatenated DD statement as shown following:

   ```
      //         DD DSN=SYSPROG.EXITS.OBJ(modname),DISP=SHR
   ```

Adding a New Exit

This publication describes some dynamic exits. IBM suggests that you use SMP/E to install them. They require definition as described in Chapter 9, “IEHINITT Dynamic Exits,” on page 241.

The following example shows SMP/E system modification statements to add a new module which does not replace an exit.

```
++USERMOD(changename)  /* USERMOD for new module.  Has JCLIN */
  .
++VER(Z038) FMID(fmid) PRE(xxxxxxx) SUP(yyyyyyy)
  .
++JCLIN .
// EXEC LINKS,
// PARM='NCAL,LIST,XREF,RENT,LET',NAME=targlib
//SYSEXITS DD DISP=SHR,VOLUME=(,RETAIN),DSNAME=exitlibrary
//SYSLIN DD *
 INCLUDE SYSEXITS(exitname)
 NAME exitname(R)
/*
++ MOD(exitname) LEPARM(LET,LIST,NCAL,RENT,XREF) .
      (object module)
```

Figure 2. Installing a New Exit

Explanation of Figure 2 on page 4:

1. Replace *changename* with a name to identify the system change. For example, you might choose a name such as UPDATE6.

   ```
   ++USERMOD(changename)  /* USERMOD for new module.  Has JCLIN */
   ```

2. In place of the fmid, use the actual FMID (function modification identifier). When you first install this exit, omit the PRE(*xxxxxxx*) SUP(*yyyyyyy*) because there probably is no prerequisite or superseding. If you later replace your own exit, you will code a different *changename* on the ++USERMOD statement and use the old *changename* with the SUP keyword.

   ```
   ++VER(Z038) FMID(fmid) PRE(xxxxxxx) SUP(yyyyyyy)
   ```

3. Replace the link edit or binder options in quotation marks as appropriate. Do not specify AC(1) directly with JCL. That would be a possible violation of system integrity and would serve no purpose. System integrity rules are described in *z/OS MVS Programming: Authorized Assembler Services Guide*. That would give APF authorization and allow someone to call the exit. Replace *targlib* with the DD name for the target library. The typical DD name is LINKLIB or LPALIB.

   ```
   // PARM='NCAL,LIST,XREF,RENT,LET',NAME=targlib
   ```

4. Replace *exitlibrary* with the DD name of a PDS for load modules or a PDSE for program objects. When you later submit an SMP/E ACCEPT statement, SMP/E will call the linkage editor or binder to store the exit in this library. You can choose a different name to replace SYSEXITS on this line and in the INCLUDE line.

   ```
   //SYSEXITS DD DISP=SHR,VOLUME=(,RETAIN),DSNAME=exitlibrary
   ```

5. Replace the three occurrences of *exitname* with the name of the exit stated in this publication. For some exits, the entry point name must match this name. The CSECT name should be the same name unless you choose a different name and code that name with the CSECT keyword. You can omit the DISTLIB keyword and parameter because that information is in the JCLIN statements. Some exit descriptions show control statements which you must use instead of these INCLUDE and NAME statements.

   ```
    INCLUDE SYSEXITS(exitname)
    NAME exitname(R)
   /*
   ++ MOD(exitname) LEPARM(LET,LIST,NCAL,RENT,XREF) .
   ```

6. Place the object module after the ++MOD statement.

   ```
   6.    (object module)
   ```

Testing Exits

You can use several techniques to make your exit testing safer and easier. They include:

- Protecting the system from exit errors
- Invoking dumps, either to retrieve debugging information or to determine what information is available in system data areas
- Issuing messages from the exits.
- Tracing module flow in OPEN, CLOSE and EOV.

Protecting the System from Exit Errors

To protect the system from exit errors, you should:

- Avoid frequent IPLs during testing because the exits reside in the link pack area (LPA).
- Prevent overwriting of vital storage because exits typically run in protection key zero.
- Limit the scope of the exit so that testing can proceed with minimal impact on other work in the system.

For exits that reside in SYS1.LPALIB you can at least partially resolve these problems by:

- Writing a test exit and placing it in a modified LPA library
- Placing the original exit code in another library, such as SYS1.LINKLIB

You can then limit the front-end test exit in scope (testing for specific job names, for example). Afterwards, when it completes its task, the test exit gives control to the original exit code outside the LPA.

The test exit, after being coded and tested, is unlikely to need frequent changes. The original exit is now in another library, where it can be changed without the assistance of an IPL. This eliminates the need for reentrant code in the original exit during testing, because it is loaded for each invocation. Run additional tests later with the original exit in the LPA to ensure that exits are truly reentrant.

A safety feature of this testing method is that, should something be wrong with the test exit, you can eliminate it by an IPL without the MLPA parameter. You should run an exit from outside the LPA only in a testing environment because of the overhead involved in loading the exit each time it is entered.

After the exit is in production mode, you can prevent unexpected problems by having the exit check the contents of the CVTUSER field (at offset 204 in the CVT). If the contents are zero (the usual 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 or display functions. This temporarily disables the exit code (a re-IPL causes CVTUSER to become zero again, reactivating the exit).

Invoking Dumps

While testing your exit, you might need a dump to debug or to examine data areas to determine where to look for information your exit requires. For general information on using dumps, refer to *z/OS MVS Diagnosis: Tools and Service Aids*. For information on analyzing dumps, refer to *z/OS MVS Diagnosis: Reference*. For information on requesting and reading dumps, refer to *z/OS Problem Management*. The items in this topic contain useful advice for calling dumps.

Issuing the ABEND Macro in an Exit

If an ABEND is issued explicitly from a preprocessing exit entered for allocation (IGGPRE00), you get message IEF197I SYSTEM ERROR DURING ALLOCATION. The job attempting the allocation fails with a JCL error, and a dump is not called. We do not recommend issuing ABEND alone for getting the information you need in that exit, but it is useful in other exits.

Setting CVTSDUMP

The CVTSDUMP flag in the CVT can be set on to cause dumps to SYS1.DUMP to be started 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 with the console alter or display functions. If you are testing under VM, use the appropriate CP commands (such as DISPLAY and STORE).

Issuing the SDUMP Macro

You can call dumps from exits with the SDUMP macro. As an alternative to the CVTSDUMP procedure described in Setting CVTSDUMP, this eliminates modifying storage to start the dump. For information on the syntax and coding of the SDUMP macro, refer to *z/OS MVS Diagnosis: Tools and Service Aids*. For information on analyzing dumps, refer to *z/OS MVS Diagnosis: Reference*. For information on requesting and reading dumps, refer to *z/OS Problem Management* and *z/OS MVS Diagnosis: Reference*. For information on the SVC dump that is produced by issuing the SDUMP macro, refer to *z/OS Problem Management*.

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. From there you can start 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 can enter messages giving the current status of processing. For example, if you are testing a DADSM exit, you can enter 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 appears on the output of the job that issued the DADSM request. Messages can also indicate that certain data areas have been found successfully and can display selected contents of data areas.

When the exit begins handling large numbers of jobs, remove the code that produces these messages. Large numbers of messages consume system message buffers, and the text unnecessarily adds to the user's output. Some exception messages might be required.

Tracing Module Flow in OPEN, CLOSE and EOV

The OPEN, CLOSE and EOV functions support a function called module flow tracing. It is for debugging system code and can help with debugging the OPEN, CLOSE and EOV exits that are described in Chapter 3, “Tape Label Processing Installation Exits,” on page 73. You can request this function by coding DIAGNS=TRACE on the DD statement. You also must start generalized trace facility (GTF) to trace USR events. *z/OS DFSMSdfp Diagnosis* describes OPEN, CLOSE, and EOV module flow tracing. *z/OS MVS Diagnosis: Tools and Service Aids* describes GTF.

Chapter 2. Data Management Installation/Dynamic Exits

This topic discusses how installation-written exit modules can:

- Take control before and after DADSM processing.
- Take control during open for a DCB.
- Recover from errors that can 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 or control interval size.
- Alter messages sent to 3480, 3490, and 3590 tape operators.

You can substitute your own exit routines for the data management replaceable modules that are listed in Table 1 on page 7.

Table 1. Data Management Replaceable Modules

Module Name	Description	When Available
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
IGGDARU3	DADSM RENAME postprocessing exit	Before exit from RENAME
IGGDASU3	DADSM SCRATCH postprocessing exit	Before exit from SCRATCH
IGXMSGEX	Message display exit	Before end of tape cartridge message display processing

The data management replaceable modules that you replace must be named the same as the IBM-supplied modules. In general, the module you replace must:

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

You can also add an exit routine for the data management dynamic exits listed in Table 2 on page 7.

Table 2. Data Management Dynamic Exits

Module Name	Description	When Available
IGGPRE00_EXIT **IGGPOST0_EXIT**	DADSM pre-processing and post-processing dynamic exit routine. The system adds IGGPRE00 and IGGPOST0 as an exit routine to their respective dynamic exit.	DADSM create, extend, scratch, partial release and rename functions.

Table 2. Data Management Dynamic Exits (continued)

Module Name	Description	When Available
IFG_CLOSE_START	Early CLOSE SVC exit.	Early in the CLOSE SVC.
IFG_OPEN_START	Early OPEN SVC exit.	Early in the OPEN SVC.
IGG_STOW_START	Early STOW SVC exit.	Early in the STOW SVC.

Other products can impact the modifications that you install. For example, RACF®, a component of the Security Server for z/OS, takes control at the same time as some of the installation exits. They might compete for resources.

DADSM Installation Exits General Information

This topic addresses considerations specific to the DADSM installation exits.

User Interfaces with DADSM Installation Exits

Design and implement the interface between your DADSM exits and the system so the advantages are apparent and acceptable to users. You might be introducing new restrictions to users' methods and you might want to explain the new advantages.

Messages

You can get DADSM exit messages from:

- The system, from errors or return codes produced by the exits
- Programs that use DADSM functions, which might get new return codes from DADSM because of your exit routines
- The exits themselves, which can issue messages directly.

System Messages

Following are some of the messages the system might issue when DADSM preprocessing and postprocessing exits have been implemented.

IEF197I SYSTEM ERROR DURING ALLOCATION
This message might appear if the exit abends while entered for an allocation request.

IEC223I rc,mod,jjj,sss, ddname[-#],dev,ser,dsname
Where **mod** is IGGPRE00. This message might appear if a program check occurs in the exit during a create request.

IEF274I jjj sss ddn SPACE REQUEST REJECTED BY INSTALLATION EXIT, REASON CODE nnnn
This message is produced when the exit has rejected a DADSM create request without allowing retry on other non-system-managed volumes (the return code in register 15 is 8, as set by the exit). The reason code is the code placed in the installation reject reason code field IEXREASN of the exit's parameter list by the exit before returning to DADSM. The *jjj* value indicates the job, *sss* the step, and *ddn* the ddname.

IEF275I jjj sss ddn SPACE REQUEST CANNOT BE SATISFIED, INSTALLATION EXIT REASON CODE nnnn
This message is produced when the exit has rejected a DADSM Create request and allowed retry on other non-system-managed volumes, but the request could not be satisfied (the return code in register 15 is 4, as set by the exit). The reason code is the code placed in the installation reject reason code field IEXREASN of the exit parameter list by the exit before returning to DADSM. The *jjj* value indicates the job, *sss* the step, and *ddn* the ddname.

Messages from Other Programs

Utility programs can provide nonzero return codes received from DADSM in their messages. Here is a summary:

- Create
  - X'B0' exit unconditionally rejected the request; no further volumes attempted
  - X'B4' exit conditionally rejected the request; try another volume
- Extend
  - –20 exit rejected the request
- Scratch
  - 4 exit rejected the request (in addition to the other meanings for this return code)
- Partial Release
  - 16 exit rejected the request (in addition to the other meanings for this return code)
- Rename
  - 4 exit rejected the request (in addition to the other meanings for this return code).

Exit Messages

If you want to selectively reject allocation requests, you can 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 creating your own messages from the exit. It might be better to have a message or code that is documented in a standard publication and then document the rejection reason codes locally, than to have a completely new message. That way the user can look up the IEF messages in *z/OS MVS System Messages, Vol 8 (IEF-IGD)* to understand why the job failed. If you do create your own message, make the contents self-explanatory so that separate documentation is not necessary.

Sometimes it helps to provide additional information. For example, if you are monitoring how much DASD space is used by a specific user or set of users, you might want to document the current running space total issued when you reject a job that asked for space that would make 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. This type of space management function is better performed using ACS routines, which are described in *z/OS DFSMSdfp Storage Administration*.

If your installation has someone who monitors I/O-related messages, you might want to produce a warning message when a running space total is getting close to being exceeded. There is little 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 alerts a space manager that a space shortage problem is imminent.

Documenting Your Exits

Provide documentation to publicize your space control policies. Logical section headings might be:

The need for space control
Space usage standards
How usage standards are enforced
Space conservation hints
New messages
Installation rejection reason codes

DADSM Pre- and Post processing Dynamic Exits (IGGPRE00_EXIT, IGGPOST0_EXIT)

All DADSM functions (create, extend, scratch, partial release, and rename) call a common pre-processing dynamic exit routine (IGGPRE00_EXIT) and a common post-processing exit routine (IGGPOST0_EXIT). The system adds the IGGPRE00 and IGGPOST0 modules as exit routines to their associated dynamic exits. These two modules are equivalent to what was provided in releases prior to z/OS V1R13. DADSM functions will call each dynamic exit using dynamic exit services (CSVDYNEX) during pre- and post-processing so that each added exit routine to IGGPRE00_EXIT and IGGPOST0_EXIT is called, allowing the exit routine to gain control before and after DADSM processing.

Adding DADSM pre and post processing exit routines

All existing operator commands and system services that apply to managing dynamic exits will apply to these two dynamic exits. For example, to add an exit routine to a dynamic exit the following services could be used:

- A program can use CSVDYNEX REQUEST(ADD).
- The SETPROG EXIT operator command.
- The EXIT statement of a PROGxx PARMLIB member, followed by the SET PROG=xx operator command.

For example using the SETPROG operator command, the following will add the IGGPRE01 exit routine to the pre-processing dynamic exit of IGGPRE00_EXIT

```
SETPROG EXIT,ADD,EXITNAME=IGGPRE00_EXIT,MODNAME=IGGPRE01
```

You can now have multiple exit routines associated with each dynamic exit.

Replacing an already active IGGPRE00 or IGGPOST0 dynamic exit routine without an IPL is also supported. For example, the SETPROG EXIT,DELETE operator command can be issued followed by the SETPROG EXIT, ADD operator command to replace a dynamic exit routine.

Characteristics of the IGGPRE00 and IGGPOST0 Exit Routines

The exit routines run under the following conditions:

- Reenterable
- Protection key zero
- Supervisor state
- With no locks held

The exit routines can run in either 24-bit or 31-bit addressing mode. If they operate in 24-bit mode, consider the following:

For create, extend, and partial release, the DADSM caller passes the address of the job file control block (JFCB), or of a copy of the JFCB, in the IEXPTR1 field of the parameter list. It might be a 31-bit address, so if your exit routine is called for either create, extend, or partial release, and the JFCB resides above the 16MB line, your routine must be in 31-bit addressing mode before using the IEXPTR1 field.

Please note that the references to IGGPRE00 and IGGPOST0 in the following sections generically refer to the exit routine(s) that have been added to the IGGPRE00_EXIT and IGGPOST0_EXIT dynamic exits. The system will add IGGPRE00 and IGGPOST0 as exit routines, but others could be added.

Understanding when IGGPRE00_EXIT and IGGPOST0_EXIT dynamic exits, exit routines are available

IGGPRE00 receives control *after* the successful initial validity check, and *before* the first VTOC update. The parameter list passed to IGGPRE00 contains a function code, which identifies the DADSM function requested, and the addresses of required input data. IGGPRE00 receives control once for each volume in

the volume list provided for scratch and rename. IGGPRE00 can use the IEXRSVWD field in the parameter list to pass data to IGGPOST0.

The operating system gives control to IGGPOST0 after a DADSM function has been completed or attempted. The system gives control to IGGPOST0 if it gives control to IGGPRE00, whether the DADSM function was successful or not. The system does not give IGGPOST0 control if it does not give IGGPRE00 control, nor if the DADSM function terminates abnormally. IGGPRE00 can 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.

The EOF mark that is written during creation of certain data sets (with primary space specified and either DSORG=PS or no DSORG) is written after IGGPOST0 receives control.

DADSM, or the program that invokes DADSM, acquires the system resources it needs to serialize system functions (by issuing enqueues, reserves, or others). These enqueues can prevent other system services from completing successfully. In particular, IGGPRE00 and IGGPOST0 must not issue dynamic allocation, open, close, end-of-volume, locate, or 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 which exit routine is being given control. For example, on entry to IGGPRE00, DADSM holds an enqueue on the VTOC and a reserve on the device for the volume specified by a scratch, rename, or partial release request. DADSM dequeues these resources before IGGPOST0 is given control.

You must anticipate potential system resource contention when IGGPRE00 or IGGPOST0 request services. 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.

Rejecting DADSM requests in IGGPRE00_EXIT dynamic exit, exit routines

To reject a DADSM request in IGGPRE00 provide either return code 4 or 8 to the calling DADSM function. A 4 indicates the request is rejected for the current volume, and might be retried on another volume. An 8 is an unconditional rejection.

Rejecting a Partial Release Request

If you reject a partial release request in IGGPRE00, partial release provides a return code that indicates an I/O error.

Rejecting a Scratch or Rename Request

If you reject a scratch or rename request in IGGPRE00, scratch and rename provide a status code that indicates an I/O error.

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 when deleting a data set. If IGGPRE00 rejects the scratch request, the VVR entry has been deleted, but the format-1 DSCB still exists. This makes the data set inaccessible and corrupts the catalog by creating inconsistencies between the BCS, VVDS, and VTOC. You `are responsible for ensuring that your IGGPRE00 routine does not reject a DADSM scratch request for a VSAM data set.` You can test bit DS1ORGAM in the DSCB, which is on for VSAM data sets.

Rejecting a Create (Allocate) Request in IGGPRE00

If you reject a create (allocate) request in IGGPRE00, create (allocate) provides either return code X'B0' or X'B4' to its caller. You can write IGGPRE00 to provide a reason code in the IEXREASN field of the parameter list, which create returns to its caller along with the X'B0' or X'B4' return code. The reason code appears as the last two bytes of the diagnostic information displayed as part of message IGD17040I.

Scheduler allocation treats return code X'B0' from allocate as an unconditional rejection of the allocate request. Scheduler allocation treats X'B4' as a conditional rejection of the allocate request, only rejecting 'the request' for the volume being processed. If the allocate request is not for a specific volume, another volume can be chosen and the allocate function retried.

Rejecting an Extend Request

If you reject an extend request in IGGPRE00, extend provides error return code X'FFFF FFEC' to its caller. If the caller is end-of-volume, end-of-volume issues an E37-0C abend.

Returning a Model Format-1 DSCB

For create and extend requests on a new volume, IGGPRE00 can return, in the IEXFMT1 field of the parameter list, the address of the data portion of a model format-1 DSCB, which starts with field PD1FMTID, also called DS1FMTID. Move the DSCB to the create or extend work area before you build the format-1 DSCB. The only fields that can be nonzero in the model DSCB are the DS1REFD field (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 abnormally terminate the task or cause unpredictable results. All other fields are initialized by allocate or extend.

IGGPRE00 might not provide a DSCB address in the IEXFMT1 field for a VIO allocate request (indicated by flag, IEXVIO, set to one), or if a partial DSCB has been provided to allocate instead of a JFCB (indicated by flag IEXMF1 being set to one). In the latter case, IEXFMT1 is passed to IGGPRE00, initialized to the address of the PD1FMTID field of the partial format-1 DSCB (supplied to allocate by its caller) in the create work area, and DS1REFD can be initialized by IGGPRE00. If extend is successful, IEXFMT1 is zeroed out before giving control to IGGPOST0.

Registers on Entry to the IGGPRE00 and IGGPOST0 Exit Routines

Following are the registers on entry to the IGGPRE00 and IGGPOST0 exit routines.

Register
Contents

1
Address of the pre- or postprocessing exit parameter list

0, 2-12
Not applicable

13
Address of an 18-word save area

14
Return address to DADSM

15
Address of the entry point to IGGPRE00 or IGGPOST0

IGGPRE00_EXIT and IGGPOST0_EXIT dynamic exits, exit routine Parameter List

Register 1 contains the address of the DADSM preprocessing or postprocessing exit parameter list, obtained from storage below the 16MB line. The IECIEXPL macro maps the parameter list shown in Table 3 on page 12:

Table 3. IGGPRE00, IGGPOST0 Parameter List. Mapped by IECIEXPL.

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	4	IEXID	ID = 'IEPL'
04 (X'04')	1	IEXLENG	Length of parameter list

Table 3. IGGPRE00, IGGPOST0 Parameter List. Mapped by IECIEXPL. *(continued)*

Offset	Length or Bit Pattern	Name	Description
05 (X'05')	1	IEXFUNC	DADSM function code
	0000 0111	IEXVEXT	Extend (VSAM caller without DEB parameter). If IEXVEXT is on, you must ensure that your exit routines do not attempt to use the IEXPTR2 field (DEB address is undefined for the extend function).
	0000 0110	IEXPREL	PARTREL partial release
	0000 0101	IEXREN	Rename
	0000 0100	IEXPR	Partial release
	0000 0011	IEXSCR	Scratch
	0000 0010	IEXEXT	Extend
	0000 0001	IEXALL	Create (allocate)
06 (X'06')	1	IEXEXTCD	Extend code
	1000 0001		Extend VSAM data set on current volume
	0000 0100		Extend non-VSAM data set on new volume
	0000 0001		Extend non-VSAM data set on current volume

Table 3. IGGPRE00, IGGPOST0 Parameter List. Mapped by IECIEXPL. *(continued)*

Offset	Length or Bit Pattern	Name	Description
07 (X'07')	1	IEXFLAG	Flag byte
	1...	IEXENQ	VTOC is enqueued upon entry
	.1..	IEXVIO	VIO data set
	..1.	IEXMF1	IEXFMT1 points to PD1FMTID of a partial format-1 DSCB (partial DSCB passed as input to allocate, and JFCB is not available)
	...1	IEXFDSCB	Full format-1 DSCB (ALLOC=ABS)
	.... xxx.	IEXAVGR	Average record value. These bits apply only to the allocate function (IEXALL) with the JFCB specified (IEXPTR1). At most, one of these three bits might be on. If all are zero, this is not an average record space request.
	.... 1...	IEXAVGRU	Average record space request. Multiply JFCBPQTY and JFCBDRLH to get requested primary space in bytes. Multiply JFCBSQTY and JFCBDRLH to get requested secondary space in bytes.
	.... .1..	IEXAVGRK	Average record space request. Multiply JFCBPQTY, JFCBDRLH, and 1024 to get requested primary space in bytes. Multiply JFCBSQTY, JFCBDRLH, and 1024 to get requested secondary space in bytes.
	.... ..1.	IEXAVGRM	Average record space request. Multiply JFCBPQTY, JFCBDRLH, and 1,048,576 to get requested primary space in bytes. Multiply JFCBSQTY, JFCBDRLH, and 1,048,576 to get requested secondary space in bytes.
	1	IEXFEDT2	Extent descriptor table 2 (pointed to by IEXEDT2) contains valid extent information. This bit is always on for pre- and postexit for scratch and partial release and postexit for create and extend. If you want your exit also to run on earlier systems that do not have an extent descriptor table 2, then IBM recommends this logic for non-VIO data sets: If IEXFEDT2 is on, then use IEXEDT2, otherwise use IEXEDT1.
08 (X'08')	2	IEXREASN	Installation reject reason code
10 (X'0A')	BIT(8)	IEXDSFLG	Data set indicators
	111.	*	RESERVED
	...1	IEXCOMPR	Compressable extended
	.... 1...	IEXPDSE	PDSE data set
	.... .1..	IEXSTRP	Extended format data set
	.... ..1.	IEXPDSEX	HFS data set
	1	IEXVSAM	VSAM data set
11 (X'0B')	1	IEXNUMF9	Number of contiguous format 9 DSCBs at the IEXFMT9 storage address that is provided by the pre-exit for create. In z/OS V1R10, the values of zero and one are valid.
12 (X'0C')	4	IEXUCB	Address of the UCB. It will be an actual 31–bit UCB address above or below the 16MB line. The UCB address is not available to the pre-exit for VIO allocation.

Table 3. IGGPRE00, IGGPOST0 Parameter List. Mapped by IECIEXPL. *(continued)*

Offset	Length or Bit Pattern	Name	Description
16 (X'10')	4	IEXPTR1	Address of one of the following: • JFCB or a copy of the JFCB (create, extend, partial release) • Data set name (PARTREL partial release) • Scratch or rename parameter list (See *z/OS DFSMSdfp Advanced Services* for a description of the scratch or rename parameter list.) When the scheduler work area (SWA) resides above the 16MB line, you might have to modify the exit routine references to the IEXPTR1 field. See "Characteristics of the IGGPRE00 and IGGPOST0 Exit Routines" on page 10 for further information.
20 (X'14')	4	IEXPTR2	Address of one of the following: • DEB (extend on old volume) • DCB (partial release entered by CLOSE). • Partial DCB (PARTREL partial release, caller is VSAM). DCBFDAD and DCBDEBA are defined, the associated DEB has been constructed; DEBDSCBA, DEBNMEXT, and the DEBDASD segments are defined. DEBDVMOD is not defined. • Current volume list entry (scratch, rename)
24 (X'18')	4	IEXDSN	Address of the data set name
28 (X'1C')	4	IEXFMT1	Address of the 96-byte data portion of the format-1 DSCB (pre-exit for scratch; pre- and post-exit for partial release and rename; post-exit for create). Might be supplied by pre-exit of create, and extend on new volume, to serve as a model if IEXMF1 and IEXVIO are zero. The format-1 DSCB is mapped by the IECSDSL1 macro. You can find the fields in *z/OS DFSMSdfp Advanced Services*. For large format data sets, unique bit settings in fields DS1FLAG1 and DS1LSTAR may need to be examined.
32 (X'20')	4	IEXFMT9	Storage address of a format 9 DSCB model. Pre-exit and post-exit for create. Might be supplied by the pre-exit for create to serve as a model if IEXMF1 is zero. Allows the pre-exit to alter the contents of the following fields: DS9ATRI1, DS9ATRV1. DS9ATRI1 and DS9ATRI2 may contain reserved fields. The values in the remaining fields will have no effect. The exit should leave reserved fields as binary zeroes to prevent them from updating the real DSCB when IBM defines new fields in future releases.
36 (X'24')	4	IEXFMT3	Address of the format-3 DSCB (ALLOC=ABS)
40 (X'28')	4	IEXEDT1, IEXEXTBL	Address of extent descriptor table 1 with 2-byte relative track addresses (pre- and postexit for scratch and partial release; postexit for create and extend). For the VIO create post-exit, this is the address of DS1EXT1 in the virtual format-1 DSCB (IEXEDT1 is a synonym for IEXEXTBL). If the volume capacity is more than 65,535 tracks, then this table does not contain extent information. There is no mapping macro for this table. This extent descriptor table is provided for coexistence with previous releases of the system. See the descriptions of IEXFEDT2 and IEXEDT2.

Table 3. IGGPRE00, IGGPOST0 Parameter List. Mapped by IECIEXPL. *(continued)*

Offset	Length or Bit Pattern	Name	Description
44 (X'2C')	4	IEXDCC	DADSM return code (post-exit only). The return code from the called DADSM function is put here.
48 (X'30')	4	IEXRSVWD	Reserved word for use by exit routine
52 (X'34')	4	IEXEDT2	Address of extent descriptor table 2 with 4-byte relative track addresses (pre- and postexit for scratch and partial release; postexit for create and extend). Macro ICVEDT02 maps this table. The format of this table is described in *z/OS DFSMSdfp Advanced Services*. For VIO create postexit, IEXEDT2 is the address of DS1EXT1 in the virtual format-1 DSCB. Also see bit IEXFEDT2.

System Control Blocks Used by IGGPRE00_EXIT and IGGPOST0_EXIT dynamic exits, exit routines

The common IGGPRE00 and IGGPOST0 parameter list contains the addresses of system control blocks. The following table shows the mapping macros:

Mapping Macro	Control Block
DCBD	DCB[1]
ICVEDT02	EDT02
IECIEXPL	DADSM pre- and postprocessing installation exit parameter list
IECPDSCB	Partial DSCB
IECSDSL1	DSCB
IEFJFCBN	JFCB[1]
IEFTIOT1	TIOT[1]
IEFUCBOB	UCB[1]
IEZDEB	DEB[1]
IHADSAB	DSAB[1]

[1]Only certain fields in these control blocks are intended as programming interfaces.

For create, extend, and partial release, the address of the JFCB passed to the user exit points to a copy of the real JFCB. Updating the JFCB copy does not cause 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 might not be turned on in this dummy JFCB. For a create request on an SMS-managed volume, if the pre-processing exit changes any space amount, it will have no effect.

For partial release, the DCB and DEB whose addresses appear in the parameter list are created for internal DADSM processing only.

During extend processing for a VSAM data set, the exit is provided with the address of a dummy DEB. This DEB does not contain any extent information. DEBs for PDSEs and for sequential extended-format data sets contain incomplete extent information.

Registers on Return from the IGGPRE00_EXIT and IGGPOST0_EXIT dynamic exits, exit routines

When IGGPRE00 or IGGPOST0 returns to DADSM, register contents must be as follows:

Register
Contents

0-14
Same as on entry to IGGPRE00, IGGPOST0

15
Return code for IGGPRE00

IGGPRE00_EXIT dynamic exit, exit routine return codes

No return codes are defined for IGGPOST0. These return codes only apply to IGGPRE00.

Return Code
Description

00 (X'00')
Continue processing the DADSM request

04 (X'04')
Reject the request for the current volume. The request might be retried on another volume.

08 (X'08')
Unconditionally reject the DADSM request.

Scratch and Rename Installation Exits (IGGDASU3, IGGDARU3)

You can use the scratch and rename installation exits to do special processing after DADSM scratch and rename processing.

Replacing the Scratch and Rename Exit Routines

See “Replacing an Existing Exit” on page 3.

You can apply PTFs to scratch or rename with SMP/E without modifying your own versions of the two exit routines.

Characteristics of the Scratch and Rename Exit Routines

The exit routines run under the following conditions:

- Reenterable
- Protection key 5
- 31-bit addressing mode
- Standard linkage conventions

When DADSM scratch is called, IGGDASU3 receives control immediately *after* scratch processing completes.

When DADSM rename is called, IGGDARU3 receives control immediately *after* rename processing completes.

The DASSVCEP flag bit in the DASFLAG6 of the scratch parameter list, and the DARSVCEP flag bit in the DARFLAG4 field of the rename parameter list indicate whether the function was called through an SVC. You might test these bits in your exit routines to determine how the DADSM function was called.

Registers on Entry to the Scratch and Rename Exit Routines

Following is the list of registers on entry to the scratch and rename exit routines.

Register
Contents

1
Address of the parameter list used by all exit routines

0, 2-12
Not applicable

13
Address of an 18-word save area

14
Return address to DADSM scratch or rename

15
Address of the entry point to the exit routine

Scratch Parameter List (IGGDASCR)

Register 1 contains the address of the scratch parameter list used by IGGDASU3. See Table 4 on page 18. The address of a list of DASD volumes is also passed to the exit routines in the DASAVOLL field of the scratch parameter list.

Table 4. Scratch Parameter List. Mapped by IGGDASCR.

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	44	IGGDASCR	DADSM scratch parameter list
00 (X'0')	8	DASPLID	ID = 'IGGDASCR'
08 (X'08')	2	DASPVER	Version of parameter list
10 (X'0A')	2	DASPLEN	Length of parameter list
12 (X'0C')	1	DASPKEY	Key of parameter list
	xxxx	DASPSKEY	Storage key of parameter list
13 (X'0D')	1		Reserved
14 (X'0E')	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
	1...	DASSMSMG	System-managed data set
	.1..	DASUNCAT	Uncataloged system-managed data set
	..1.	DASCATCL	Catalog is the caller
	...1	DASGDGRO	GDG rolloff in progress
	.... 1...	DASGDGRN	GDG rolloff noscratch

Table 4. Scratch Parameter List. Mapped by IGGDASCR. *(continued)*

Offset	Length or Bit Pattern	Name	Description
22 (X'16')	1	DASFLAG3	Functionally authorized request flags (Part 1)
	1...	DASFAUTH	Caller guarantees functional authorization of request
	.1..	DASSAUTH	Caller guarantees security authorization of request
	..1.	DASPROFM	RACF profile managed by caller
	...1	DASBTIOT	Bypass SYSZTIOT ENQ
	.... 1...	DASBDSN	Bypass SYSDSN ENQ
	.... .1..	DASBDEB	Bypass DEB check (DSN not open)
	.... ..1.	DASNVALL	Do not dynamically allocate volumes
	1	DASNERAS	Do not erase any extents
23 (X'17')	1	DASFLAG4	Functionally authorized request flags (Part 2)
	1...	DASVOLMT	Caller guarantees that the volume is already mounted
24 (X'18')	1	DASFLAG5	Authorization not required request flags
	1...	DASOVRPD	Override purge date
	.1..	DASVRFRD	Verify last-referenced-date unchanged (DFSMShsm)
	..1.	DASERASE	Erase all extents
25 (X'19')	1	DASFLAG6	Parameter flag byte
	x...	DASSVCEP	**If** **Then** **0** Scratch entered with SMS branch entry **1** Scratch entered at SVC entry point
26 (X'1A')	2		Reserved
28 (X'1C')	4	DASUCB	Address of primary mountable UCB. Can be captured or actual below the 16 MB line.
28 (X'1C')	4	DASVDSCB	Address of UCB for a VIO data set
32 (X'20')	4	DASAVOLL	Address of volume list mapped by (IGGDAVLL); see Table 6 on page 21.
36 (X'24')	4	DASADSN	Address of data set name
40 (X'28')	3	DASREFDT	Reference date to check (DFSMShsm)
43 (X'2B')	1		Reserved

Rename Parameter List (IGGDAREN)

Register 1 contains the address of the rename parameter list used by IGGDARU3. The IGGDAREN macro maps the rename parameter list, shown in Table 5 on page 20. The address of a list of DASD volumes is also passed to the exit routines in the DARAVOLL field of the rename parameter list.

Table 5. Rename Parameter List. Mapped by IGGDAREN.

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	44	IGGDAREN	DADSM rename parameter list
00 (X'00')	8	DARPLID	ID = 'IGGDAREN'
08 (X'08')	2	DARPVER	Version of parameter list
10 (X'0A')	2	DARPLEN	Length of parameter list
12 (X'0C')	1	DARPKEY	Key of parameter list
	xxxx	DARPSKEY	Storage key of parameter list
13 (X'0D')	1		Reserved
14 (X'0E')	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
	1...	DARSMSMG	SMS-managed data set
	.1..	DARUNCAT	Rename uncataloged data sets only
22 (X'16')	1	DARFLAG3	Functionally authorized request flags (Part 1)
	1...	DARFAUTH	Caller guarantees functional authorization of request
	.1..	DARSAUTH	Caller guarantees security authorization of request
	..1.	DARPROFM	RACF profile managed by caller
23 (X'17')	1	DARFLAG4	Functionally authorized request flags (Part 2)
	x...	DARSVCEP	**If** **Then** **0** Rename entered with SMS branch entry **1** Rename entered at SVC entry point
	.x..	DARBPDSC	When this bit is on, do not update "data set changed" bit in the format-1 DSCB
24 (X'18')	4		Reserved
28 (X'1C')	4	DARUCB	Address of primary mountable UCB. Can be captured or actual below the 16MB line.
32 (X'20')	4	DARAVOLL	Address of volume list mapped by (IGGDAVLL); see Table 6 on page 21.
36 (X'24')	4	DARADSN	Address of old data set name
40 (X'28')	4	DARANDSN	Address of new data set name

DADSM Volume List (IGGDAVLL)

Table 6 on page 21 shows the DADSM volume list mapped by IGGDAVLL.

Table 6. DADSM Volume List (Mapped by IGGDAVLL)

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	16	IGGDAVLL	DADSM volume list
00 (X'00')	16	DAVLLHDR	Volume list header
00 (X'00')	8	DAVLLID	ID = 'IGGDAVLL'
08 (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
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 or rename status byte

Registers on Return from the Scratch and Rename Exits

When you return control to DADSM, register contents must be as follows:

Register
Contents

0-14
Same as on entry to the exits

DASD Calculation Services Installation Exits (IGBDCSX1, IGBDCSX2)

Many of the system's services use DASD calculation services (DCS) to determine optimum block sizes for non-VSAM data sets and optimum control interval sizes for VSAM data sets. You can use the DASD calculation services installation exits to influence the size chosen. The precalculation exit routine, IGBDCSX1, indicates to DCS which value to use when calculating the optimum block or control interval size. The postcalculation exit routine, IGBDCSX2, lets you override the DCS-calculated optimum block or control interval size with your own value.

Note: After z/OS V2R2, DFSMS no longer uses the DCS optimal CI size function when calculating the CI size during the define of a VSAM data set.

Replacing the IGBDCSX1 and IGBDCSX2 Exit Routines

See “Replacing an Existing Exit” on page 3. Your routine must have an entry point name that matches the exit name. Your CSECT is linked together with certain system CSECTs into a single load module.

Characteristics of the IGBDCSX1 and IGBDCSX2 Exit Routines

IGBDCSX1 and IGBDCSX2 run under the following conditions:

- Reenterable
- Calling program's protection key
- Runs AMODE (31), RMODE (ANY), and resides in ELPA storage
- Calling program's system state (problem or supervisor)
- DCS provides 1KB of working storage for each exit routine

IGBDCSX1 gains control *before* DCS calculates the optimum block size or control interval size. You can use IGBDCSX1 to either bypass or limit the DCS-calculated optimum block or control interval size. IGBDCSX2 gains control *after* DCS calculates the statistics you requested. You can use it to override the DCS-calculated optimum block or control interval size with a value of your own.

DCS retrieves DASD data set information, performs calculations, and returns statistics to its caller. These statistics are primarily provided for display by ISMF (Interactive Storage Management Facility). The values returned are designated in kilobytes or bytes, rather than cylinders or tracks, to eliminate device dependency.

IGBDCSX1 and IGBDCSX2 allow you to exercise control over the values returned. However, because the access methods limit maximum block size to 32,760, if an exit routine returns an override or limit greater than this, DCS sets the block size to 32,760. DCS also verifies that exit-supplied control interval size override values do not violate VSAM restrictions.

Registers on Entry to the IGBDCSX1 and IGBDCSX2 Exit Routines

The following is a list of the registers on entry to the IGBDCSX1 and IGBDCSX2 exit routines.

Register
Contents

1
Address of a 4-byte field containing the address of the parameter list

0, 2-12
Not applicable

13
Address of an 18-word save area

14
Return address to DCS

15
Address of the entry point to IGBDCSX1 or IGBDCSX2

IGBDCSX1 and IGBDCSX2 Parameter List

Register 1 contains an address of a 4-byte field that has the address of the DCS pre- and postcalculation exit parameter list. The IGBDCSIE macro maps the parameter list, shown in Table 7 on page 22. For PDSEs and extended format data sets the block size is simulated and DASD calculation service returns device-independent larger values.

Field DCSIEVRS contains a version number. A version value in the range of 1-20 indicates that the record format (DCSIERFM) is present in the parameter list; any other version value indicates it is not.

Table 7. IGBDCSX1/IGBDCSX2 Parameter List

Offset	Length or Bit Pattern	Name	Description
00 (X'00')		DCSIEPL	DCS installation exit parameter list
00 (X'00')	44	DCSIEDSN	Data set name

Table 7. IGBDCSX1/IGBDCSX2 Parameter List (continued)

Offset	Length or Bit Pattern	Name	Description
44 (X'2C')	2	DCSIEDSO	Data set organization (only physical sequential, partitioned and VSAM should come to this exit). The unmovable bit might be on if not SMS-managed.
	X'8000'	DCSIEIS	Indexed sequential organization
	X'4000'	DCSIEPS	Physical sequential organization, standard or extended format, possibly compressed.
	X'2000'	DCSIEDA	Direct organization (BDAM)
	X'1000'	DCSIECX	BTAM or QTAM line group
	X'0200'	DCSIEPO	Partitioned organization (partitioned data set or PDSE)
	X'1011'	DCSIEU	U-unmovable. The data contains location-dependent information. This bit can be on for all data set organizations except VSAM.
	X'0080'	DCSIEGS	Graphics organization
	X'0008'	DCSIEAM	VSAM data set
46 (X'2E')	1	DCSIEVRS	Version
47 (X'2F')	1	DCSIERFM	Record format
48 (X'30')	4	DCSIEKP	Key position
52 (X'34')	4	DCSIELRL	Logical record length (average record length if VSAM)
56 (X'38')	4	DCSIETC	Track capacity
60 (X'3C')	4	DCSIEBUF	Buffer space
64 (X'40')	4	DCSIESTG	Exit workspace address
68 (X'44')	2	DCSIEKL	Key length
70 (X'46')	2	DCSIEBS	Block size (current physical block size if VSAM)
72 (X'48')	2	DCSIECOB	Calculated optimum block size
74 (X'4A')	6	DCSIEVSN	Volume serial number, if available, or blanks

Registers on Return from the IGBDCSX1 and IGBDCSX2 Exit Routines

Before returning control to DCS, the exit routines must set up these registers:

Register
Contents

0
Block or control interval size, if the return code is non-zero.

1-14
Same as on entry to the exit routine

15
Return code for the exit

IGBDCSX1 and IGBDCSX2 Return Codes

When you return control to DASD calculation services, you must place one of the following return codes in register 15, depending on the exit.

IGBDCSX1

Return Code
Description

00 (X'00')
DCS can proceed normally.

04 (X'04')
DCS can proceed, using the unsigned value in register 0 as the maximum possible value. Size provided in register 0.

08 (X'08')
Bypass DASD calculation services and use the size you put in register 0.

IGBDCSX2

Return Code
Description

00 (X'00')
You accepted the calculated block or control interval size.

08 (X'08')
You are overriding the DCS-calculated block or control interval size with the value you provided in register 0.

Example of the IGBDCSX1 Exit Routine

Figure 3 on page 25 is a sample of IGBDCSX1. It returns the installation-defined maximum block size for non-VSAM data sets whose first four characters are SYS2 or SYS3. For non-VSAM data sets whose first four characters are SYS4 or SYS5, IGBDCSX1 returns the only block size allowed. For all other data sets, no restrictions apply. The figure 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.

```
*********************************************************************
*                                                                   *
*  $MOD(IGBDCSX1): DASD CALCULATION SERVICES PRE-CALCULATION EXIT   *
*                                                                   *
*  DESCRIPTIVE NAME = DCS PRE-CALCULATION EXIT                      *
*                                                                   *
*  COPYRIGHT = NONE                                                 *
*                                                                   *
*  FUNCTION = SEE TEXT FOR ENTRY AND EXIT INFORMATION               *
*                                                                   *
*     PROCESSOR = ASSEMBLER                                         *
*                                                                   *
*     ATTRIBUTES = CALLER KEY, CALLER STATE, ENABLED,               *
*                  AMODE(31),RMODE(ANY)                             *
*********************************************************************
          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-FOUR CHARACTERS 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
```

Figure 3. Sample Listing of IGBDCSX1 Part 1 of 2

```
          B       EXIT
*
*   CHECK FIRST-FOUR CHARACTERS 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-FOUR CHARACTERS 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-FOUR CHARACTERS 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
*
*   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 *
MAXSYS2   DC      F'5120'
MAXSYS3   DC      F'10240'
SYS4BSZ   DC      F'4096'
SYS5BSZ   DC      F'8192'
*
*
          IGBDCSIE
*
          END     IGBDCSX1
```

Figure 4. Sample Listing of IGBDCSX1 Part 2 of 2

Example of the IGBDCSX2 Exit Routine

Figure 5 on page 27 is a sample of IGBDCSX2. It returns a block size to override the calculated optimal block size of a non-VSAM data set if the DCS-calculated block size exceeds an installation limit and the data set resides on a particular volume. The figure is a learning aid for the installation system programmer and is not guaranteed to run on a particular system without some modification.

```
*********************************************************************
*                                                                   *
*  $MOD(IGBDCSX2):  DASD CALCULATION SERVICES POST-CALCULATION EXIT *
*                                                                   *
*  DESCRIPTIVE NAME = DCS POST-CALCULATION EXIT                     *
*                                                                   *
*  COPYRIGHT = NONE                                                 *
*                                                                   *
*  FUNCTION = SEE TEXT FOR ENTRY AND EXIT INFORMATION               *
*                                                                   *
*     PROCESSOR = ASSEMBLER                                         *
*                                                                   *
*     ATTRIBUTES = CALLER KEY, CALLER STATE, ENABLED,               *
*                  AMODE(31),RMODE(ANY)                             *
*********************************************************************
IGBDCSX2  CSECT
IGBDCSX2  AMODE 31
IGBDCSX2  RMODE ANY
*
```

Figure 5. Sample Listing of IGBDCSX2 Part 1 of 2

```
*   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 6. Sample Listing of IGBDCSX2 Part 2 of 2

Data Management Abend Installation Exit (IFG0199I)

You can use the IBM-provided IFG0199I installation exit to recover from or record abend situations. An example is tape positioning errors. They might occur during the opening, closing, or processing of an end-of-volume condition for a non-VSAM data set associated with your application program.

The exit might need to examine the DCB, DCBE or DEB to ensure that fields were used correctly for the type of data set. For mappings of the DCB and DCBE fields, see *z/OS DFSMS Macro Instructions for Data Sets*. For mappings of the DEB fields, see *z/OS DFSMSdfp Advanced Services*.

Replacing the IFG0199I Exit Routine

The source code (written in assembler language) that is similar to the IBM-provided IFG0199I exit routine is in member OPENEXIT of SYS1.SAMPLIB. You can modify it to handle other data management abend

situations and replace the IBM-provided IFG0199I module in SYS1.LPALIB with the modified routine. If you replace the IBM-provided IFG0199I, the replacement module you supply must have the entry point name IFG0199I.

For additional information, see "Replacing an Existing Exit" on page 3.

Characteristics of the IFG0199I Exit Routine

IFG0199I runs under the following conditions:

- Reenterable
- Protection key zero
- 24-bit addressing mode
- Supervisor state

When an abnormal condition occurs, control first passes to the DCB abend *user* exit routine, if its address is provided in the DCB exit list. If a DCB abend user exit routine is not provided, or if it requests immediate abnormal termination of the task, the system passes control to IFG0199I. For a description of the DCB exit list and the DCB abend user exit routine, see *z/OS DFSMS Using Data Sets*.

The IBM-supplied IFG0199I checks the system completion code and the return code to determine whether the abend condition is the result of a tape positioning error. If the system completion code is other than 613 with return code X'08' or X'0C', IFG0199I returns control to the calling module with return code 0, indicating that abend processing should continue.

Otherwise, IFG0199I checks the counter in the 4-byte work area to determine whether an attempt to reposition the tape has been made. If not, IFG0199I provides return code 4, indicating that repositioning should be attempted. If it has, IFG0199I issues message IEC613A to the operator asking whether to attempt repositioning again.

If the operator requests another try, IFG0199I provides return code 4 to the calling module, indicating that open should rewind the tape and attempt positioning. If the operator specifies that tape positioning should not be retried, IFG0199I provides return code 0 to the calling module, and abend processing continues.

You can modify IFG0199I to perform recovery actions for other data management abend conditions. For abend codes for which you can use IFG0199I to attempt recovery, see *z/OS DFSMS Using Data Sets*.

Registers on Entry to the IFG0199I Exit Routine

Following are the registers on entry to the IFG0199I exit routine.

Register
Contents

1
Address of the data management abend installation exit parameter list

0, 2-12
Not applicable

13
Address of an 18-word save area

14
Return address to open or end-of-volume

15
Address of the entry point to IFG0199I

IFG0199I Parameter List

Register 1 contains the address of the Data Management abend parameter list. The OAIXL DSECT maps the parameter list, shown in Table 8 on page 30.

Table 8. IFG0199I Parameter List. Mapped by OAIXL DSECT. See Figure 12 on page 36 for an example of the assembler code for OAIXL.

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	0	OAIXL	Data management abend installation exit parameter list
00 (X'00')	1	OAIXUKEY	User protection key
01 (X'01')	1	OAIXFLGS	Option flags
	1...	OAIXEXIT	DCB user abend exit was taken. If 0, the user exit was not taken. The system sets this bit.
	.1..	OAIXREW	Indicates that you want the tape rewound. The bit is zero at entry to IFG0199I. If you leave it zero, the tape is not rewound.
02 (X'02')	2		Reserved
04 (X'04')	4	OAIXPDCB	Address of protected copy of DCB used by open
08 (X'08')	4	OAIXUDCB	Address of user DCB related to the abend
12 (X'0C')	4	OAIXUCBA	Address of UCB related to the abend. Can be zero for certain data sets such as dummy, spooled, TSO terminal, or z/OS UNIX file. If the application program specified the UCB nocapture (S99ACUCB) option, the UCB address might point above the 16 MB line.
16 (X'10')	4	OAIXJFCB	Address of JFCB related to the abend
20 (X'14')	4	OAIXTIOT	Address of TIOT entry related to the abend
24 (X'18')	4	OAIXCODE	Abend code, for example X'6130000C'
28 (X'1C')	4	OAIXAREA	4-byte work area
		OAIXLEN	Length of parameter list (32)

Registers on Return from the IFG0199I Exit Routine

Before you return control to open or end-of-volume, set up these registers:

Register
Contents

2-12
Same as on entry to IFG0199I

15
Return code (see following description)

IFG0199I Return Codes

Before you return control to open, place one of the following codes in register 15:

Return Code
Description

00 (X'00')
Continue with abend processing

04 (X'04')
The system attempts to recover from the abend. If the option flag bit one is on (OAIXREW) the system rewinds the tape volume and sets the UCBFSCT and UCBFSEQ fields in the UCB to zero before attempting to recover from the abend.

If the option flag bit 1 is off, the system attempts to recover from the abend.

Example of the IFG0199I Exit Routine

Figure 12 on page 36 is a sample of IFG0199I. Its source is available in SYS1.SAMPLIB member OPENEXIT.

```
*********************************************************************
*                                                                   *
* MODULE NAME = IFG0199I                                            *
*                                                                   *
* DESCRIPTIVE NAME = DATA MANAGEMENT ABEND INSTALLATION EXIT        *
*                                                                   *
* FUNCTION = THE SYSTEM CALLS THIS EXIT FOR MOST ABENDS ISSUED      *
*    IN OPEN,CLOSE,EOV FOR THE ACCESS METHODS OTHER THAN VSAM.      *
*    IF THE APPLICATION PROGRAM SUPPLIED A DCB ABEND EXIT, THEN     *
*    IT DID NOT RECOVER FROM OR PREVENT THE ABEND.  THIS MODULE     *
*    DOES THE FOLLOWING:                                            *
*                                                                   *
*    1. GET STORAGE FOR A WORK AREA.                                *
*    2. BRANCH TO EXIT LOGIC TO CONTINUE THE ABEND IF IT IS NOT     *
*       A 613-08 OR 613-0C ABEND OR IF THE USER DCB ABEND EXIT      *
*       WAS CALLED.  THESE TWO ABENDS HAVE TO DO WITH TAPE          *
*       POSITIONING ERRORS.                                         *
*    3. IF THIS IS THE FIRST INSTANCE OF ONE OF THESE TWO ABENDS    *
*       DURING THIS OPEN, THEN BRANCH TO EXIT LOGIC TO RETRY.       *
*    4. IF THIS IS THE SECOND TIME IT HAS OCCURRED IN THIS OPEN,    *
*       ENTER WTOR TO ASK AN OPERATOR WHETHER TO RETRY UP TO TWO    *
*       MORE TIMES OR TO CONTINUE THE ABEND.                        *
*                                                                   *
*    PATCH LABEL = PATCH                                            *
*                                                                   *
*    ATTRIBUTES = REENTRANT, REFRESHABLE, ENABLED, READ ONLY,       *
*                 PRIVILEGED, SUPERVISOR STATE, KEY ZERO,           *
*                 LINK PACK AREA RESIDENT/PAGEABLE, 24-BIT          *
*                 ADDRESSING MODE                                   *
*
```

Figure 7. Sample Listing of IFG0199I Part 1 of 7.

```
*    LINKAGE = BALR  R14,R15   BRANCH AND LINK                      *
*                                                                   *
*    INPUT REGISTERS =                                              *
*         1 - ADDRESS OF PARAMETER LIST MAPPED BY MACRO IECOIEXL     *
*         13 - ADDRESS OF STANDARD SAVE AREA                        *
*         14 - ADDRESS OF CALLER                                    *
*         15 - ADDRESS OF ENTRY POINT IN THIS MODULE.               *
*                                                                   *
*    CONTROL BLOCK = JFCB, DCB, UCB, TIOT, OAIXL                    *
*********************************************************************
IFG0199I CSECT
START    STM   R14,R12,12(R13)          SAVE SYSTEM REGISTERS
         BASR  R11,0                    LOAD PROGRAM BASE
         USING *,R11                    USING R11 AS BASE REGISTER
         L     R0,SIZDATAD              GET DSECT SIZE
         GETMAIN R,LV=(0)               GET DSECT STORAGE
         LR    R10,R1                   SAVE GETMAINED AREA
         USING DATAD,R10                REGISTER 10 DSECT REGISTER
         ST    R13,SAVEAREA+FOUR        SAVE R13 FOR BACK POINTER
         LM    R15,R1,16(R13)           RELOAD CALLERS REGISTERS
         ST    R10,8(R13)               SAVE R10 FOR FORWARD POINTER
         LR    R13,R10                  POINT TO NEW SAVE AREA
         LR    R12,R1                   LOAD OAIXL REGISTER FROM PARM
         USING OAIXL,R12                DEFINE BASE TO OAIXL
PSATOLD  EQU   X'21C'                   ADDRESS OF CURRENT TCB
TCBTIO   EQU   X'00C'                   DISPLACEMENT IN TCB
         L     R8,PSATOLD               LOAD TCB ADDR FROM PSA
         L     R7,TCBTIO(,R8)            LOAD TIOT ADDR FROM TCB
         USING TIOT,R7                  DEFINE BASE TO TIOT
*
         L     R6,OAIXUCBA              LOAD UCB ADDR FROM PARAMETERS

         USING UCB,R6                   DEFINE BASE TO UCB
*
*
*********************************************************************
*        CHECK THE ABEND CODE TO BE SURE THIS IS A 613-08/0C ABEND  *
*********************************************************************
CHKABEND LH    R8,OAIXCODE              LOAD FIRST TWO BYTES OF CODE
         CH    R8,HEX613                COMPARE CODE TO ABEND 613
         BNE   CONTINUE                 CONTINUE WITH ABEND
         CLI   OAIXCODE+THREE,HEX08     COMPARE CODE TO ABEND 613-08
         BE    CHKEXIT                  YES, CHECK EXIT TAKEN
         CLI   OAIXCODE+THREE,HEX0C     COMPARE CODE TO ABEND 613-0C
```

Figure 8. Sample Listing of IFG0199I Part 2 of 7.

```
         BNE   CONTINUE                NO, CONTINUE WITH ABEND
*
*********************************************************************
*        CHECK IF THE DCB USER ABEND EXIT WAS TAKEN                 *
*********************************************************************
CHKEXIT  TM    OAIXFLGS,OAIXEXIT       TEST IF DCB USER EXIT TAKEN
         BO    CONTINUE                DO NOT OVERRIDE THE USER EXIT'S
*                                      DECISION TO ABEND
*
*********************************************************************
*        CHECK THE COUNTER TO BE SURE WE HAVE RETRIED ONE TIME      *
*********************************************************************
CHKAREA  L     R8,OAIXAREA             LOAD AREA COUNTER INTO REG 8
         LA    R8,ONE(R8)              ADD ONE TO AREA COUNTER
         ST    R8,OAIXAREA             STORE NEW SUM INTO COUNTER
         CLI   OAIXAREA+THREE,MAXTRIES COMPARE COUNTER TO TWO
         BL    RETRY                   LOW, CONTINUE TO RETRY
         SLR   R9,R9                   CLEAR REGISTER 9
         ST    R9,OAIXAREA             STORE ZERO INTO COUNTER
*
*********************************************************************
*        SETUP TO ENTER THE WTOR                                    *
*********************************************************************
TRYAGAIN SLR   R9,R9                   CLEAR REGISTER 9
         ST    R9,REPLYECB             STORE ZERO INTO REPLY ECB
*
         MVC   WTORAREA(WTORLEN),WTORLIST MOVE IN WTOR LIST
*

         MVC   WTOJOB,TIOCNJOB         MOVE JOB NAME TO WTO AREA
         OC    WTOJOB,BLANKS           FOLD TO UPPER CASE
         CLI   WTOJOB,BLANK            JOB NAME BLANK?
         BNE   BLANKJOB                NO, BRANCH
         MVI   WTOJOB,COMMA            INDICATE MISSING JOB NAME
BLANKJOB EQU   *
         MVC   WTOSTEP,TIOCSTEP        MOVE STEP NAME - WTO AREA
         OC    WTOSTEP,BLANKS          FOLD TO UPPER CASE
*
         UNPK  WTODEV(L'WTODEV+1),UCBCHAN(L'UCBCHAN+1) SPREAD DIGITS
         MVI   WTODEV+L'WTODEV,C','    FIX SIGN GARBAGE FROM UNPK
         TR    WTODEV,HEXTABLE-C'0'    CONVERT TO PRINTABLE HEX CHARS
         CLI   WTODEV,C'0'             TEST FOR LEADING ZERO DIGIT
         BNE   GETVOL                  BRANCH IF FOUR-DIGIT HEX NUMBER
         MVI   WTODEV,C' '             BLANK THE LEADING ZERO
GETVOL   EQU   *
```

Figure 9. Sample Listing of IFG0199I Part 3 of 7.

```
         MVC   WTOVOL,UCBVOLI           MOVE VOLUME SERIAL

         OC    WTOVOL,BLANKS            ENSURE UPPER CASE
*
         LA    R2,REPLY                 LOAD ADDRESS OF REPLY
         LA    R3,REPLYECB              LOAD ADDRESS OF REPLY ECB
*
*********************************************************************
*        ENTER MESSAGE TO THE OPERATOR, AND WAIT FOR                *
*        HER REPLY TO 'U' CONTINUE OR 'R' RETRY.                    *
*********************************************************************
         WTOR  ,(R2),,(R3),MF=(E,WTORAREA) ISSUE WTOR SVC
         LR    R9,R1                    SAVE MSG ID FOR DOM
*********************************************************************
*        ISSUE WAIT                                                 *
*********************************************************************
         WAIT  ECB=REPLYECB             WAIT ON REPLY
*
*********************************************************************
*        ISSUE DOM                                                  *
*********************************************************************
         DOM   MSG=(R9)                 DOM MESSAGE
*
*********************************************************************
*        CHECK REPLY FROM OPERATOR                                  *
*********************************************************************

         OC    REPLY,BLANKS             MAKE REPLY UPPER CASE
         CLI   REPLY,C'U'               WAS REPLY A 'U'
         BE    CONTINUE                 GO AND CONTINUE WITH ABEND
         CLI   REPLY,C'R'               WAS REPLY A 'R'
         BE    RETRY                    GO AND CONTINUE TO RETRY
         B     TRYAGAIN                 INVALID RESPONSE, TRY AGAIN
*********************************************************************
*        SET THE RETURN CODE FOR RETRY OR CONTINUE WITH ABEND       *
*********************************************************************
RETRY    LA    R15,FOUR                 SET RETURN CODE TO FOUR
         OI    OAIXFLGS,OAIXREW         SET ON REW BEFORE RETRY FLAG
         B     EOJ                      BRANCH TO END OF JOB
CONTINUE LA    R15,ZERO                 SET RETURN CODE TO ZERO
EOJ      L     R13,SAVEAREA+FOUR        GET CALLERS SAVE AREA ADDRESS
         ST    R15,16(R13)              SAVE RETURN CODE REGISTER
         L     R0,SIZDATAD              GET DSECT STORAGE SIZE
```

Figure 10. Sample Listing of IFG0199I Part 4 of 7.

```
         FREEMAIN R,LV=(0),A=(R10)      ISSUE FREEMAIN
         LM    R14,R12,12(R13)          RESTORE ALL REGISTERS
         BR    R14                      RETURN TO CALLER
*
*********************************************************************
*        CONSTANTS                                                  *
*********************************************************************
*
WTORLIST WTOR  'IEC613A JJJJJJJJ,SSSSSSSS,DEVN,VOLSER TAPE POSITION ERRX
               OR -- REPLY 'R'' RETRY OR ''U'' CONTINUE WITH ABEND',  X
               ,4,ROUTCDE=(1,3,5),MF=L
WTORLEN  EQU   *-WTORLIST               LENGTH OF WTOR
*
         DS    0H
HEX613   DC    X'6130'                  CONSTANT FOR 613 ABEND
*
BLANKS   DC    C'        '              BLANKS FOR REPLY
*

HEXTABLE DC    C'0123456789ABCDEF'      TO TRANSLATE TO PRINTABLE HEX

*
PATCH    DC    ((*-START)/20)X'00'      PROGRAM PATCH AREA
*********************************************************************
*        DSECT STORAGE                                              *
*********************************************************************
*
DATAD    DSECT
         DS    0D
WTORAREA DS    CL20
WTOJOB   DS    CL8                      JOB NAME
         DS    C
WTOSTEP  DS    CL8                      STEP NAME
         DS    C
WTODEV   DS    CL4                      DEVICE NUMBER
         DS    C
WTOVOL   DS    CL6                      VOLUME SERIAL
         DS    CL74                     REMAINDER OF MESSAGE
*
REPLYECB DS    F                        REPLY ECB
REPLY    DS    CL4                      REPLY RETURN AREA
*
SAVEAREA DS    18F                      PROGRAM SAVE AREA
*
R0       EQU   0
```

Figure 11. Sample Listing of IFG0199I Part 5 of 7.

```
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
ZERO     EQU   0
ONE      EQU   1
MAXTRIES EQU   2                       MAXIMUM NUMBER OF TRIES
TWO      EQU   2
THREE    EQU   3
K3       EQU   3
FOUR     EQU   4
FIVE     EQU   5
K6       EQU   6
K8       EQU   8
HEX08    EQU   X'08'
HEX0C    EQU   X'0C'
BLANK    EQU   C' '                    CHARACTER ' ' (BLANK)
COMMA    EQU   C','                    CHARACTER ',' (COMMA)
*********************************************************************
 *        OPEN ABEND INSTALLATION EXIT PARAMETER LIST                *
*********************************************************************
*
OAIXL    DSECT                   ABEND INSTALLATION EXIT LIST
OAIXUKEY DS    XL1               PROTECT KEY OF USER DCB
OAIXFLGS DS    XL1               OAIXL FLAG BYTE
OAIXEXIT EQU   X'80'             DCB USER EXIT TAKEN (SET BY SYSTEM)
*                                  00 = DCB USER EXIT NOT TAKEN
*                                  80 = DCB USER EXIT TAKEN
OAIXREW  EQU   X'40'             REWIND TAPE BEFORE RETRY (CAN BE SET
                                        BY THE EXIT ROUTINE)
*                                  00 = DO NOT REWIND THE TAPE
*                                  40 = REWIND THE TAPE
OAIXRESV DS    H                 Reserved
OAIXPDCB DS    A                 ADDRESS OF PROTECTED COPY OF THE DCB
```

Figure 12. Sample Listing of IFG0199I Part 6 of 7.

```
OAIXUDCB DS    A                  ADDRESS OF THE USER'S DCB
OAIXUCBA DS    A                  UCB ADDRESS
OAIXJFCB DS    A                  JFCB ADDRESS
OAIXTIOT DS    A                  TIOT ADDRESS
OAIXCODE DS    F                  ABEND CODE- EXAM X'6130000C'
OAIXAREA DS    F                  INSTALLATION WORK AREA
OAIXLEN  EQU   *-OAIXL            LENGTH OF OAIXL
*
*********************************************************************
*        DCB - THE DCBD MACRO IS IN SYS1.MACLIB
*********************************************************************
         DCBD DSORG=(PS,IS,DA,TQ),DEVD=(DA,TA)        MAP FOR DCB
*********************************************************************
*        UCB - THE IEFUCBOB MACRO IS IN SYS1.AMODGEN                *
*********************************************************************
UCB      DSECT
         IEFUCBOB LIST=YES
*********************************************************************
*        TIOT - THE IEFTIOT1 MACRO IS IN SYS1.AMODGEN               *
*********************************************************************
TIOT     DSECT
         IEFTIOT1
*********************************************************************
*        JFCB - THE IEFJFCBN MACRO IS IN SYS1.AMODGEN               *
*********************************************************************
JFCB     DSECT
         IEFJFCBN LIST=YES
*********************************************************************
*        DATA DEFINITIONS FOR DYNAMIC STORAGE AREA                  *
*********************************************************************
IFG0199I CSECT
         DS    0F
SIZDATAD DC    AL1(230)                SUBPOOL NUMBER
         DC    AL3(ENDDATA-DATAD)      SIZE OF DSECT
DATAD    DSECT
         ORG   *+1-(*-DATAD)/(*-DATAD) INSURE DSECT DATA
ENDDATA  EQU   *
         END
```

Figure 13. Sample Listing of IFG0199I Part 7 of 7.

DCB Open Installation Exit (IFG0EX0B)

You can use the IFG0EX0B installation exit to:

- Specify an installation-determined value for the number of buffers for QSAM DCBs.
- Direct the system to determine the block size for an output data set.
- Modify the JFCB to:
  - Request partial release of DASD space
  - Change the DASD secondary space request parameters.

Replacing the IFG0EX0B Exit Routine

See “Replacing an Existing Exit” on page 3. Your routine must have an entry point name that matches the exit name. Your CSECT is linked together with certain system CSECTs into a single load module.

Characteristics of the IFG0EX0B Exit Routine

IFG0EX0B runs under the following conditions:

- Protection key 0
- 24-bit addressing mode
- Supervisor state
- No locks held

During IFG0EX0B-related processing, system enqueues are issued to serialize system functions. These enqueues might prevent other system services from completing successfully. In particular, IFG0EX0B should not start dynamic allocation, open, close, end-of-volume, or DADSM functions, because of an enqueue on the SYSZTIOT resource.

If IFG0EX0B requires access to an installation data set, the control blocks needed to access that data set (DCB, DEB) should be built during system initialization. RACF macros can be called from the exit.

If your exit does not update a particular DCB during a parallel open and that DCB does not have an exit list with a DCB OPEN exit entry or a JFCBE entry, then when your exit is processing a different DCB for the same OPEN, the two copies of the first DCB might differ. OPEN will make them consistent later.

Understanding IFG0EX0B Execution Environment

Table 9 on page 38 shows when IFG0EX0B receives control, and the processing that occurs before and after it is called.

Table 9. IFG0EX0B's Execution Environment

Step	Action
1	Begin open processing **For DASD data sets:** • Mount the volume • Read the format -1, -2, and -3 DSCBs • Merge information from the format 1 DSCB to the JFCB **For tape data sets:** • Mount the volume • Verify the header labels • Merge information from the header labels to the JFCB **For all data sets:** • Merge information from the JFCB to the DCB • Take the DCB open user exit or the JFCBE user exit, if either is specified in the DCB's exit list. • Perform RACF or password verification processing.
2	Call IFG0EX0B in the following cases: • When the OPEN macro (with or without the TYPE=J parameter) processes a DCB, including when two or more DCBs are opened in parallel through a single invocation. • When a program is reading a sequential concatenation of data sets with unlike attributes. IFG0EX0B receives control as the program begins reading each data set. Data sets are considered unlike only if the program has turned on the DCBOFPPC flag in the DCBOFLGS field. For example, data sets with different record format are considered unlike.
3	Complete open processing: • Calculated the block size if the block size in the DCB or DCBE is zero. • Merge information from the DCB to the JFCB (not all fields are merged). • Merge information from the JFCB to the format-1 DSCB for DASD data sets (not all fields are merged). • Write header labels for output tape data sets. • Perform access-method-dependent processing (obtain buffers, DEB, and other control blocks). • Update the system copy of the JFCB.

Using the Data Control Block (DCB)

Open maintains a protected copy of the user DCB (in the user's program) in the combined work and control block area. IFG0EX0B can use open's copy of the DCB to test DCB field values. If you use IFG0EX0B to modify either the user DCB or OPEN's copy, you must also modify the other. The protection key of the user DCB is supplied in the parameter list passed to IFG0EX0B. IFG0EX0B must use this key to get information from, or modify the user DCB. Open does not copy the DCBE. OPEN requires that the DCBE be writable in the same key as the program that issued the OPEN macro. To maintain system integrity, your exit routine must switch to the key of the issuer of OPEN whenever accessing the user's DCB or DCBE.

If you use IFG0EX0B to change values in the DCB for a data set that has been allocated to a system-managed volume, you must not specify values that would change the data set to a type that cannot be SMS managed. Doing so causes abnormal termination of the task. For example, you cannot specify an unmovable data set organization.

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 "Example of the IFG0EX0B Exit Routine" on page 43 provides an example of excluding all DCBs from processing except for QSAM DCBs being opened to a DASD or tape device.

Specifying a Value for the Number of Buffers

You can use IFG0EX0B to provide an installation-determined value for the number of buffers for QSAM DCBs, if a value has not yet been supplied. See "Example of the IFG0EX0B Exit Routine" on page 43.

- For QSAM DCBs, do not override a nonzero value in DCBBUFNO without knowing what dependency the user program might have on that value. If the DCBBUFCA field in the DCB contains a buffer pool control block address, it indicates that buffers have been acquired before open. Thus, you cannot override DCBBUFNO. After return from IFG0EX0B, if DCBBUFCA is zero, open turns on the low order bit (the absence of a buffer pool control block is indicated by the low order bit of DCBBUFCA being one). Unless the user specified RMODE31=BUFF in the DCBE, the user program is responsible for freeing buffer space after closing the DCB.
- For BSAM DCBs with DCBBUFCA set to one, do not override a zero value in DCBBUFNO without knowing what dependency the user program might have on that value. 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.

Modifying the Job File Control Block (JFCB)

The JFCB address provided in the OIEXJFCB field of the parameter list points to a copy of the JFCB in the combined work and control block area. If you are accessing concatenated partitioned data sets, other JFCBs might be associated with the OPEN macro.

If you modify the JFCB through IFG0EX0B, you must provide return code 4 to open. This causes open to update the system copy of the JFCB. Do not modify the JFCB through IFG0EX0B if the user program has set JFCNWRIT (bit 4) in byte JFCBTSDM, because this indicates that the system copy of the JFCB should not be updated. A sample IFG0EX0B routine that modifies the JFCB is shown in "Example of the IFG0EX0B Exit Routine" on page 43.

If the OPEN macro specifies EXTEND or OUTINX, and the disposition in the JFCB is not MOD, open temporarily changes its JFCB copy to specify DISP=MOD.

Requesting Partial Release of DASD Data Set Space

The best way to control partial release is to assign SMS management classes to data sets. The PARTIAL RELEASE management class attribute is easier to define and maintain than using IFG0EX0B, and the management classes give you greater flexibility in defining and changing your partial release policy. You can modify the JFCB in the open work area to specify partial release, as shown in "Example of the

IFG0EX0B Exit Routine" on page 43. The example sets JFCRLSE (bits 0 and 1; mask X'C0') to 1 in the JFCBIND1 field, indicating a request for partial release. Do this only for DASD physical-sequential data sets, sequential extended-format data sets, or partitioned data sets opened for OUTPUT, OUTIN, EXTEND, and OUTINX and processed by one of the following:

- EXCP with a 5-word device-dependent section present in the DCB
- BSAM
- QSAM
- BPAM

Be careful when you change the JFCB release bits. If you frequently open a data set for output and write varying amounts of data each time, DADSM can extend the data set after each OPEN and create many small extents, and perhaps reach the extent limit for the data set. This could result in a B37 abend.

Also, use caution when setting the JFCBSPAC bits to define the space quantity units when the partial release flag, JFCBRLSE, is also set on.A cylinder-allocated extent can be released on a track boundary if 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) can prevent such a B37 abend. Setting the release indicator could result in more space being made available to other users sharing the volume.

Updating DASD Secondary Space Data

You can modify the JFCB in open's work area to update the secondary space quantity for DASD data sets, as shown in "Example of the IFG0EX0B Exit Routine" on page 43. The JFCBCTRI field contains the space request type coded in the DD statement or merged from the format-1 DSCB. The JFCBSQTY field contains the amount of secondary space requested. The JFCBPQTY field contains the amount of primary space requested. The JFCBCTRI field specifies the units in cylinders, tracks, or average blocks. If bit JFCBAVR is on, then the unit is either average blocks or average records.

Changing the value of the JFCONTIG bit has no effect on the space allocation because that bit affects only the primary space, which has already been allocated before IFG0EX0B is called.

Using the Unit Control Block (UCB)

For BDAM and concatenated partitioned data sets, the UCB whose address is supplied to the exit might not be the only UCB that is associated with the DCB that is being opened. Do not modify the UCB. Data sets such as dummy, spooled, TSO terminal, and z/OS UNIX file have a zero UCB address.

Finding the Task Input/Output Table (TIOT) Entry

The TIOT address provided in the OIEXTIOT field points to a TIOT entry (TIOENTRY) label in the data area mapped by the IEFTIOT1 macro. For concatenated partitioned data sets, other TIOT entries might be associated with the DCB being opened. If the application program is opening a concatenation of unlike attributes, the TIOT entry can have a blank DDNAME field.

Finding the Data Set Control Blocks (DSCBs)

The format-1 or format-8 DSCB address provided in the OIEXDSCB field points to the start of the DS1FMTID field for the DSCB in the common work and control block area. Format-8, -9 and -3 DSCBs might be associated with the format-1 or format-8 DSCB. For BDAM and concatenated partitioned data sets, other format-1, -8, -9 and -3 DSCBs might be associated with the DCB being opened.

A format-1 or -8 DSCB may have several fields set uniquely if the data set is a large format data set. The fields affected are DS1FLAG1 and DS1LSTAR. For specific values in these fields that are unique to large format data sets, see *z/OS DFSMSdfp Advanced Services*. For a large format data set, the DCBE might also have BLOCKTOKENSIZE=LARGE specified to indicate that the program conforms to interfaces that are required for some uses of large format data sets. These interfaces, which support increased

track numbers, are required for EXCP, or BSAM with NOTE or POINT, or if the IGDSMS*xx* member of SYS1.PARMLIB specifies BLOCKTOKENSIZE(REQUIRE). For a mapping of the DSCB, see *z/OS DFSMSdfp Advanced Services*.

If the VTOC is being opened, OIEXDSCB points to a format-4 DSCB. You can determine if the DSCB is a format-4 by testing the DS1FMTID field for a value of X'F4', or by testing the data in the JFCBDSNM field for 44 bytes of X'04'. For IFG0EX0B to receive control when processing a VSAM data space, you must provide a DCB, not an access method control block (ACB).

Directing the System to Determine Block Size

If DCBLRECL has been set up and DCBRECFM specifies either F, V, or D you can write a routine as part of the IFG0EX0B exit to have the system determine the block size for the data set. Do this by setting the user's block size field to zero before the exit returns to its caller. If you are requesting a system determined block size, the DCBBLKSI field in the DCB and the DCBEBLKSI field in the DCBE, if present, should be set to zero. A sample is shown in “Example of the IFG0EX0B Exit Routine” on page 43.

The user's block size value is in the 2-byte DCBBLKSI in the DCB if both of the following are true:

- Bit DCBMRECP is off, meaning it is not an EXCP DCB.
- Any of DCBH0, DCBH1, DCBESLBI and DCBEULBI is off. The user is not using LBI, large block interface.

The user's block size value is in the 4-byte DCBEBLKSI in the DCBE if all of the following are true:

- The DCB is not for EXCP, meaning that bit DCBMRECP is off or it is on and the 5-word device interface section of the DCB is valid, meaning bit DCBMR5WD is on.
- Bits DCBH0 and DCBH1 are on, meaning that DCBDCBE points to the DCBE.
- Bit DCBESLBI is on, meaning that the system supports LBI for this DCB.
- Bit DCBEULBI is on, meaning that the user requests LBI.

For an EXCP DCB that is not using the DCBE BLKSIZE field, your exit can use the BLKSIZE field in the JFCB but the application program might not use it.

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 for OUTPUT or OUTIN and the merging of the block size fields occurs, open checks the indicator. If the block size is a system-determined block size, and the logical record length or record format have changed from those specified in the data set label, open determines the block size again.

Registers on Entry to the IFG0EX0B Exit Routine

Following are the registers on entry to the IFG0EX0B exit routine.

Register
Contents

1
Address of the DCB open installation exit parameter list

0, 2-12
Not applicable

13
Address of an 18-word save area

14
Return address to open

15
Address of the entry point to IFG0EX0B

IFG0EX0B Parameter List

Register 1 contains the address of the DCB open installation exit parameter list. The IECOIEXL macro maps the parameter list, shown in Table 10 on page 42.

Table 10. IFG0EX0B Parameter List. Mapped by IECOIEXL.

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	0	OIEXL	DCB Open installation exit parameter list
00 (X'00')	1	OIEXOOPT	Open option (last four bits)
	 1111	OIEXOOUT	OUTPUT or EXTEND
	 0111	OIEXOOIN	OUTIN or OUTINX
	 0100	OIEXOUPD	UPDAT
	 0011	OIEXOINO	INOUT
	 0001	OIEXORDB	RDBACK
	 0000	OIEXOINP	INPUT
01 (X'01')	1	OIEXUKEY	User protection key (key of user DCB)
02 (X'02')	2	OIEXLTH	Length of open installation exit parameter list (OIEXL)
04 (X'04')	4	OIEXUDCB	Address of user DCB in user protection key (OIEXUKEY)
08 (X'08')	4	OIEXPDCB	Address of protected copy of DCB used by open
12 (X'0C')	4	OIEXJFCB	Address of JFCB
16 (X'10')	4	OIEXDSCB	Address of data portion of the format-1 or format-4 DSCB
20 (X'14')	4	OIEXTIOT	Address of TIOT entry
24 (X'18')	4	OIEXUCB	Address of UCB. Can be nocaptured (above the 16 MB line), captured (below the 16 MB line), or actual below the 16 MB line.

Registers on Return from the IFG0EX0B Exit Routine

Before you return control to the caller, set up the registers as follows:

Register
Contents

0-14
Same as on entry to IFG0EX0B

15
Return code

IFG0EX0B Return Codes

Following are the return codes for IFG0EX0B exit routine.

Return Code
Description

00 (X'00')
You have not modified the JFCB.

04 (X'04')
You have modified the JFCB.

Processing: BUFNO Subroutine

If the number of buffers indicated in the QSAM DCBBUFNO field for certain DASD or tape data sets is zero when IFG0EX0B gets control, the BUFNO subroutine defaults the number of buffers. The block size in the DCB (DCBBLKSI) is used, together with a fixed amount of storage (64 KB in the example) to determine a buffer number. The 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 (1 KB) bytes. You can use IFG0EX0B to alter the values in the DAMAX and TPMAX tables.

Processing: SCREEN Subroutine

The SCREEN subroutine determines whether the RLSE or SQTY subroutines should be run. DASD sequential and partitioned data sets being processed by BSAM or QSAM and opened for OUTPUT or OUTIN are selected. The VTOC and data sets starting with SYS1. are excluded. You can write IFG0EX0B to make further selection tests or add other support, for example:

- You can handle secondary allocation differently for an sequential extended-format data set. They can have up to 123 extents per volume, whereas physical sequential data sets can have up to 16 user data extents.
- Default the number of buffers larger if the data set is an sequential extended-format data set.
- For a compressed data set, a spooled data set, dummy data set, or z/OS UNIX file it is not useful to override the QSAM BUFNO default of 1. The extra buffers are not used productively. If you know the application program has no dependence on the number of buffers, your exit can decrease a user-provided BUFNO.

Processing: RLSE Subroutine

The RLSE subroutine sets the partial release indicators to 'on' in the JFCB if the number of extents in the data set is less than a fixed value (8 in the example). It sets the partial release indicators in the JFCB to 'off' if the number of extents in the data set is equal to or greater than a fixed value (8 in the example). Partitioned data sets are not processed because they can be opened many times to write one new member for each OPEN-CLOSE macro sequence.

Processing: SQTY Subroutine

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, the secondary is set to a fixed default number of tracks (5 in the example). If the primary quantity is one, the secondary is set to the same fixed default (5); the secondary quantity shown 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; thesecondary quantity shown is in units of tracks, cylinders, or average blocks, depending on that of the primary quantity.

You can provide space defaults more efficiently by using data classes. This allows greater selectivity according to the type of data set. To use data classes, SMS must be running, but the data set does not have to be SMS-managed.

Example of the IFG0EX0B Exit Routine

The following program listing is a sample of IFG0EX0B. The four subroutines (BUFNO, SCREEN, RLSE, and SQTY) show examples of the process you can do in your installation's version of IFG0EX0B.

```
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                             *
```

Figure 14. Sample Listing of IFG0EX0B Part 1 of 13.

```
*        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 =                                                   *
```

Figure 15. Sample Listing of IFG0EX0B Part 2 of 13.

```
*           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                                   *
*           R9  JFCB ADDRESS                                         *
*           R10 FORMAT 1 DSCB ADDRESS                                *
*           R11 SAVE RETURN CODE                                     *
*           R13 SAVE AREA ADDRESS                                    *
*           R14 RETURN ADDRESS                                       *
*           R15 BASE REGISTER                                        *
*                                                                    *
*    PATCH LABEL = PATCH                                             *
*                                                                    *
*    ATTRIBUTES = REENTRANT, REFRESHABLE,READ-ONLY, ENABLED,         *
*                 PRIVILEGED, SUPERVISOR STATE, KEY ZERO,            *
*                 LINK PACK AREA RESIDENT/PAGEABLE                   *
*                                                                    *
* ENTRY POINT = IFG0EX0B                                             *
*                                                                    *
*    PURPOSE = SEE FUNCTION                                          *
*                                                                    *
*    LINKAGE =                                                       *
*          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                                       *
*                                                                    *
```

Figure 16. Sample Listing of IFG0EX0B Part 3 of 13.

```
* EXIT-NORMAL =                                                        *
*          BR 14                                                       *
*                                                                      *
* EXIT-ERROR =                                                         *
*        NONE                                                          *
*                                                                      *
* EXTERNAL REFERENCES = SEE BELOW                                      *
*                                                                      *
*    ROUTINES = NONE                                                   *
*                                                                      *
*    DATA AREAS = NONE                                                 *
*                                                                      *
*    CONTROL BLOCK = DCB, JFCB, UCB, TIOT, DSCB                        *
*                                                                      *
* 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
*
```

Figure 17. Sample Listing of IFG0EX0B Part 4 of 13.

```
*********************************************************************
MODJFCB  EQU   4                  RETURN CODE IF JFCB MODIFIED
         USING IFG0EX0B,RCODE
*********************************************************************
*
*        START OF SAMPLE PROGRAM
*
*********************************************************************
         B     AFTRID1
         DC    C'IFG0EX0B JDM1137 &SYSDATE'
AFTRID1  SAVE  (14,12)            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
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 IHADCB,RDCB
```

Figure 18. Sample Listing of IFG0EX0B Part 5 of 13.

```
         ********************************************************************
*        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?
         BZ    BUFTAPE            BRANCH IF NOT DASD
```

Figure 19. Sample Listing of IFG0EX0B Part 6 of 13.

```
         LA    RMAX,DAMAX         MAX TABLE FOR DASD
         L     RDSCB,OIEXDSCB     POINT TO DSCB
         USING DS1FMTID,RDSCB
         CLI   DS1FMTID,C'1'      BRANCH IF NOT FORMAT 1
         BNE   CALC
         TM    DS1SMSFG,DS1STRP   GO IF EXTENDED SEQ., SYSTEM
         BO    RETBUFNO           DEFAULT BUFNO IS BETTER
         B     CALC               GO CALCULATE BUFNO
BUFTAPE  EQU   *
         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 MAX 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
         STC   RODD,DCBBUFNO      PUT IN USER DCB
         MODESET EXTKEY=ZERO      BACK TO KEY ZERO
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
```

Figure 20. Sample Listing of IFG0EX0B Part 7 of 13.

```
*********************************************************************
*
*        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 RUN
*********************************************************************
*        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
```

Figure 21. Sample Listing of IFG0EX0B Part 8 of 13.

```
         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
*                    IS 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 CAUSES THIS ROUTINE TO
*          NEVER PREVENT A REQUEST FOR PARTIAL RELEASE
*
*        SETTING RLSE1 TO 0 CAUSES THIS ROUTINE TO
*          NEVER FORCE A REQUEST FOR PARTIAL RELEASE
*
*********************************************************************
```

Figure 22. Sample Listing of IFG0EX0B Part 9 of 13.

```
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 IS SET
EXTRLSE0 DC    AL1(RLSE0)        IF THIS NUMBER OR MORE EXTENTS,      *
                                 PARTIAL RELEASE IS 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
```

Figure 23. Sample Listing of IFG0EX0B Part 10 of 13.

```
         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
```

Figure 24. Sample Listing of IFG0EX0B Part 11 of 13.

```
         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
*********************************************************************
*
*        DCB - THE DCBD MACRO
*
*********************************************************************
         DCBD DSORG=PS,DEVD=DA
*********************************************************************
*
*        UCB - THE IEFUCBOB MACRO
*
*********************************************************************
UCB      DSECT
         IEFUCBOB LIST=YES
*********************************************************************
*
```

Figure 25. Sample Listing of IFG0EX0B Part 12 of 13.

```
*        TIOT - THE IEFTIOT1 MACRO
*
*********************************************************************
TIOT     DSECT
         IEFTIOT1
*********************************************************************
*
*        JFCB - THE IEFJFCBN MACRO
*
*********************************************************************
JFCB     DSECT
         IEFJFCBN LIST=YES
*********************************************************************
*
*        FORMAT 1 DSCB - THE IECSDSL1 MACRO *
*********************************************************************
F1DSCB   DSECT
         IECSDSL1 (1)
         END
```

Figure 26. Sample Listing of IFG0EX0B Part 13 of 13.

VSAM EOV Installation Exit (IDAEOVXT)

This module allows users to dynamically add to allocation control block information during extends of certain VSAM data sets. For information about supported data sets, see "Characteristics of the IDAEOVXT Exit Routine" on page 56.

Replacing the IDAEOVXT Exit Routine

Products or installations that replace the VSAM version of IDAEOVXT with their own module must link-edit their version into LPALIB as a separate load module that is named IDAEOVXT. For more information about replacing exit routines, see "Replacing an Existing Exit" on page 3.

Characteristics of the IDAEOVXT Exit Routine

IDAEOVXT is packaged as a single load module. It is entered in 31-bit addressing mode and must return in 31-bit addressing mode. You can replace IDAEOVXT with another module that sets a return code of 0 and causes VSAM EOV to try the extend. The replacement IDAEOVXT module must be reentrant and is entered in the supervisor state and key 0. Return to VSAM EOV must be in supervisor state and key 0.

IDAEOVXT receives control when all candidate volumes have been used and dynamic volume count usage is unsuccessful. IDAEOVXT receives control for all extended format VSAM data sets and VSAM data sets that are SMS managed, except for the following data sets:

- Keyrange data set
- IMBED data set
- Temporary data set
- Catalog
- Catalog opened as a data set
- "System" data set
- VVDS
- Data set that was opened for improved control interval processing
- Data set using control blocks in common
- SNAPSHOT EOV

IDAEOVXT is not used for VSAM RLS.

Registers on Entry to the IDAEOVXT Exit Routine

The following is a list of the registers on entry to the IDAEOVXT exit routine.

Register
Contents

1
Address of the VSAM EOV installation exit parameter list

0, 2–12
Not applicable

13
Address of an 18-word save area

14
Return address to VSAM EOV

15
Address to the entry point to IDAEOVXT

IDAEOVXT Parameter List

Register 1 contains the address of the VSAM OEV parameter list, which is shown in Table 11 on page 57.

Table 11. IDAEOVXT Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of a 1-byte area that contains the indicator of the condition that caused IDAEOVXT to be entered: • X'01' — Out of space and no volumes are available with space for the required allocation • X'00', X'02'-X'FF' — Reserved
04 (X'04')	4	The address of a 44-byte area that contains the VSAM component name.
08 (X'08')	4	The address of a 4-byte area that contains the DSAB address of the DSAB being used by VSAM EOV for the current extend. It might not be the DSAB for the ACB that actually caused EOV to be entered.

Registers on Return from the IDAEOVXT Exit Routine

When IDAEOVXT returns to VSAM EOV, the contents of register 15 are as follows:

Register
Contents

15

0
VSAM EOV must try the extend.

4
VSAM EOV must continue with EOV termination.

All other values are reserved for future use.

Tape Cartridge Message Display Installation Exit (IGXMSGEX)

You can use the message display installation exit to:

- Customize messages for display on an IBM 3480, 3490, or 3590 tape drive by modifying the message text pointed to by the parameter list, based on job name, step name, or some other means. The message text includes the parameters from the MSGDISP macro. The system issues the MSGDISP macro for system-managed drives the same as it does for non-managed drives.
- Specify no automatic cartridge loading.

If you plan to use DFSMSrmm, you must use the IGXMSGEX exit routine it provides. If you have already written your own IGXMSGEX routine, you might have to merge it with the source code provided by DFSMSrmm. See *z/OS DFSMSrmm Implementation and Customization Guide* for more information.

Installing the IGXMSGEX Exit Routine

See “Replacing an Existing Exit” on page 3. Your routine's entry point name must be IGXMSGEX. The MSGDISP function calls IGXMSGEX for MOUNT, DEMOUNT, VERIFY, and GEN.

Characteristics of the IGXMSGEX Exit Routine

IGXMSGEX runs under the following conditions:

- Reenterable
- Supervisor state

- Protection key 0
- 31-bit addressing mode
- Any residency mode

If you want to process differently based on the job or step name, use the PSATOLD pointer to the current TCB. If TCBTIO is nonzero, then it points to the TIOT which contains the job name and the step name. See Figure 27 on page 60 to see how to use PSATOLD and the TIOT to find job and step names.

Controlling the Automatic Cartridge Loader

To request that no automatic cartridge loading be performed, use the IGXMSGEX routine to set bit 7 in the format control byte to zero. To request automatic cartridge loading, set bit 7 to one.

When IGX00030 receives control back from IGXMSGEX, it processes according to the fields that IGXMSGEX might have modified in the two display data fields and in bit 7 in the format control byte when it issues the LOAD DISPLAY channel command. IGX00030 also checks to see if the device in the UCB in the parameter list supports automatic cartridge loading if it is active. If the device does not support automatic cartridge loading and IGXMSGEX set bit 7 to one, IGX00030 resets the bit to zero before issuing the LOAD DISPLAY channel command.

If IGXMSGEX is not link-edited with IGX00030, MSGDISP allows automatic cartridge loading only if the device supports automatic cartridge load, and if the volume serial number is either SCRTCH or PRIVAT.

Registers on Entry to the IGXMSGEX Exit Routine

Register
Contents

0
Not applicable

1
Address of the tape cartridge message display parameter list

2-12
Not applicable

13
Address of a save area

14
Return address to the calling program

15
Address of the entry point to IGXMSGEX

IGXMSGEX Parameter List

Register 1 contains the address of the IGXMSGEX parameter list, which contains the following fields:

Table 12. IGXMSGEX Parameter List

Offset	Length	Description
00 (X'00')	4	Address of the UCB. Might be above or below the 16 MB line. If it is below the line, it might be a captured or actual address.
04 (X'04')	4	Address of MSGTEXT

The second word of the parameter list points to MSGTEXT, which is shown in Table 13 on page 59.

Table 13. MSGTEXT for IGXMSGEX

Offset	Length	Description
0 X'00'	1	Control byte. You can only change the following bit:
	1	Use the automatic cartridge loader, if available. Set this bit to 0 to disable automatic cartridge loading.
01 X'01'	8	First message text. You can modify this text.
09 X'09'	8	Second message text. You can modify this text.

Registers on Return from IGXMSGEX

Before you return control to the caller, set up the registers as follows:

Register
Contents

0-14
Same as on entry

15
Not applicable. There are no return codes for this exit.

Example of the IGXMSGEX Exit Routine

Figure 27 on page 60 is an example of the IGXMSGEX exit routine.

```
IGXMSGEX  CSECT
IGXMSGEX  AMODE 31
IGXMSGEX  RMODE ANY
          STM    14,12,12(13)       save callers registers
          LR     12,15              establish base register
          USING  IGXMSGEX,12
*   NOTE: This module must be RE-ENTRANT!!! This example does not
*   obtain a save area, but if one is necessary, it would need to
*   be GETMAINed.
*
*   To test for a specific job name or step name the following
*   code sequence may be used
*   NOTE: This code assumes that a deferred tape mount is done
*   if deferred tape mounts are not requested, the TIOCNJOB
*   field will contain 'INIT    ' and the TIOCSTEP+8 field
*   will contain the 8 byte job name
          USING  PSA,0
          L      15,PSATOLD         get TCB address
          USING  TCB,15             map TCB
          L      15,TCBTIO          get TIOT address
          USING  TIOT,15            map TIOT
          LTR    15,15              is there a TIOT
          BZ     NOTIOT             no, branch
          CLC    TIOCNJOB,MYJOB     check for interesting job
          BNE    NOTMYJOB           branch if "not my job"
          CLC    TIOCSTEP(8),MYSTEP  check for interesting step
          BNE    NOTMYSTP           branch if "not my step"
          DROP   15
*   Of the bits in the control byte, only the ACL request bit
*   may be modified. If any other bit is changed, IGX00030 will
*   reset it to the original state. If the ACL request bit is
*   set on, it will only be honored if the device actually
*   supports the automatic cartridge load (ACL) feature. The UCB
*   can be tested to ensure that the ACL feature is supported.
*   The system will have set ACL request bit on if the device
*   supports ACL, the ACL load is ready, and the volser is SCRTCH or
*   PRIVAT.
          USING  PLIST,1            map input parameters
          L      2,MSGTEXTA         get message text address
          USING  MSGTEXT,2          map message text
          TM     CONTROL,ACL        is ACL request bit on
          BO     ACLON              yes branch
          L      3,UCBA             get UCB address from PLIST
          USING  UCB,3
```

Figure 27. Sample Listing of IGXMSGEX Part 1 of 2

```
*   We could just always set the bit on and let the calling module
*   (IGX00030) test to see if the ACL is supported, but we will test
*   the UCB just as an example.
          TM     UCBTFL1,UCBCSL     is ACL supported
          BNO    ACLOFF             no, branch
          OI     CONTROL,SETACLON   set ACL request bit on
SETACLON  EQU    X'01'              ACL on bit
ACLON     EQU    *                  ACL on
ACLOFF    EQU    *                  ACL off
*
NOTMYJOB  EQU  *
NOTMYSTP  EQU  *
NOTIOT    EQU  *
*   NOTE: This exit could also be used to update the message text
*   in the parameter list (i.e. MSG1 and MSG2 fields). An example
*   of this has not been included because the update would be
*   specific to implementation details of a specific installation.
          LM     14,12,12(13)       restore callers registers
          SR     15,15              set return code, (not used by
*                                     IGX00030, just good practice)
          BR     14                 return to caller
MYJOB     DC     CL8'MYJOB'         job name of interest
MYSTEP    DC     CL8'MYSTEP'        step name of interest
*  Parameter list pointed to be register 1
PLIST     DSECT
UCBA      DS     A                  UCB address
MSGTEXTA  DS     A                  Message text address
MSGTEXT   DSECT
CONTROL   DS     CL1
ACL       EQU    X'01'              ACL request bit
MSG1      DS     CL8                first message
MSG2      DS     CL8                second message
****** Include control block mapping macros
          IHAPSA                    PSA mapping
          IKJTCB                    TCB mapping
TIOT      DSECT                     provide DSECT statement for TIOT
          IEFTIOT1                  TIOT mapping
UCB       DSECT                     provide DSECT statement for UCB
          IEFUCBOB                  UCB mapping
          END
```

Figure 28. Sample Listing of IGXMSGEX Part 2 of 2

OPEN/CLOSE/EOV and access method SVC exits

The OPEN, STOW, and CLOSE SVCs invoke these SVC installation exits:

IFG_OPEN_START
Near the beginning of the OPEN SVC, which is SVCs 19 and 22. SVC 22 is for OPEN TYPE=J, where the caller is providing a JFCB update. The system also calls this exit when processing a sequential concatenation. This call is when the application program begins reading another data set. The call is not when the application program issues the OPEN macro. For a partitioned concatenation, this exit is called only when the application program issues the OPEN macro.

IGG_STOW_START
Near the beginning of STOW, SVC 21. Programs that perform operations on PDS or PDSE members normally issue this macro, but the binder does not use STOW for program objects. Program objects reside only in PDSEs. For program objects, the binder uses the DESERV FUNC=PUT macro. DESERV calls a different installation exit that is dynamically installed, but it does not use the CSVDYNEX macro. For more information, see *z/OS DFSMSdfp Advanced Services*.

IGG_CLOSE_START (SVC 20)
Near the beginning of CLOSE SVC 20. This does not include CLOSE TYPE=T (SVC 23) because it is not really closing the data set. It is only repositioning access to the data set. The system also calls this exit when processing a sequential concatenation. In a sequential concatenation, this call is when the application program finishes reading a data set other than the last data set. Later, this exit is called when the application program issues the CLOSE macro or task termination closes the data set.

These exits use the dynamic exits service, CSVDYNEX, which is described in *z/OS MVS Programming: Authorized Assembler Services Guide* and *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN*. The dynamic exits facility allows multiple exit routines to be simultaneously defined to a single exit point. Using this facility, the system programmer can associate new exit routines with these exit points. IBM does not provide default exit routines for these exits.

z/OS also supports an installation exit called IFG0EX0B. It is not dynamic. For more information, see "DCB Open Installation Exit (IFG0EX0B)" on page 37.

The purposes of the exits might be to monitor activity for certain data sets or to enforce restrictions.

These exits are implemented by a PTF for z/OS 2.5. If the DFAOCEExits bit in the DFAFEAT11 byte in the DFA is on, it means that the dynamic exit support for open, STOW, and CLOSE is installed and available. For more information about the IHADFA macro, see *z/OS DFSMSdfp Advanced Services*.

General programming considerations

The exit routines run in task mode, protection key 5, and supervisor state. As with all subroutines, do not bind installation exits with APF authorization.

The exit routines are entered with AMODE 31 and enabled for interrupts. Primary=Home=Secondary. Primary ASC mode.

The parameter list passed to the exit routines resides in non fetch-protected, key 5 storage. The only fetch-protected area that it points to is the DCB of the user.

Although the service does not enforce reentrancy, make sure that the routine is reentrant.

The exit routines must not dequeue SYSZTIOT (in open or close) or unlock the DEB (in STOW or close). The results are unpredictable.

The caller of the exit routine provides ESTAE protection. This is to support the FASTPATH option. If there is an abend in open or close, the ESTAE recovery routine issues CSVDYNEX REQUEST=RECOVER to clean up and write an IEC997I message and otherwise ignore this problem. Resources that are acquired by the exit routine are lost unless the exit routine uses ESTAE to recover. An abend in an exit routine does not affect calls to other exit routines at that exit point, and does not affect that user program call to the SVC. For more information about abends, see:

- "Controlling the OPEN/CLOSE/EOV and access method SVC exits through the dynamic exits facility" on page 62
- "Registers on exit from the OPEN/CLOSE/EOV and access method dynamic exits" on page 65
- "General programming considerations" on page 65

Controlling the OPEN/CLOSE/EOV and access method SVC exits through the dynamic exits facility

Installations can associate their own exit routines with these exit points. For more information, see Adding an Exit Routine to an Exit in *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN*. The dynamic exits facility allows multiple exit routines to be defined to a single exit.

Also, for more information, see CSVDYNEX - Provide dynamic exit services in *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN*.

Operator commands and system services that apply to managing dynamic exits apply to these dynamic exits. For example, to add an exit routine to a dynamic exit, you can use the following services:

- A program can use CSVDYNEX REQUEST=ADD macro. For example,

```
CSVDYNEX REQUEST=ADD,EXITNAME=ONAME,MODNAME=O Module
STATE=ACTIVE, DSNAME=ODSNAME,ADDABENDNUM=10, PARAM=YE S, *
SERVICEMASK=OSMask
. . .
ONAME
DC CL16'IFG_OPEN_START'
OModule DC CL8'OPENEX1'
```

```
ODSNAME DC CL44'DEPT.MAIN.LIBRARY'
OSMask DC AD(B'1') 1 padded on left with 63 zeroes
Yes DC
CL8'YES'
```

- The operator can issue the SETPROG EXIT command. For example,

```
SETPROG EXIT,ADD,EXITNAME=IFG_OPEN_START,MODNAME=OPENEX1,STATE=(ACTIVE),
DSNAME=DEPT.MAIN.LIBRARY,
ADD ABENDNUM=10 ,PARAM=YES ,SERVICEMASK=1
```

  The module OPENEX1 is a member in data set DEPT.MAIN.LIBRARY. The number of abends in the exit routine that the system tolerates before making it inactive is ten.

- The system programmer can add an EXIT statement in a PROG*xx* PARMLIB member and then issue the SET PROG=*xx* operator command. For example, an EXIT statement might be:

```
EXIT ADD EXITNAME(IFG_OPEN_START) MODNAME(OPENEX1) STATE(ACTIVE)
DSNAME DEPT.MAIN.LIBRARY ) ADDABENDNUM 10 ) PARAM(YES) SERVICEMASK 1
```

The options that you can specify are:

ADDABENDNUM
Because of the FASTPATH=YES option on CSVDYNEX REQUEST=DEFINE and on CSVDYNEX REQUEST=CALL, if an abend occurs in an exit routine, the ESTAE recovery routine for open, STOW or close has control. Message IEC997I is then written and issues CSVDYNEX REQUEST=RECOVER so that CSVDYNEX calls any exit routines that are not yet called for this instance of OPEN, STOW, or CLOSE.

An abend in an exit routine does not affect the other exit routines that are defined for that exit or affect the application program. However, you can specify the number of abends in the exit that the system tolerates since IPL. If this limit is reached, the system makes the exit module inactive. The default for these exits is 2. If you code ADDABENDNUM with CSVDYNEX REQUEST=ADD, it overrides the default value of ABENDNUM=2. The system programmer might want to use message automation to handle this condition.

PARAM
A latent parameter. For more information, see "Latent parameters" on page 64.

SERVICEMASK
A bit string of up to 64 bits that are padded on the left by zeroes. For each invocation of these exits, the system sets a 64-bit service ID (identifier). You can associate a service mask with your exit routine. Only if the ANDed value of the service mask for your exit routine and the service ID for a given call is nonzero the system calls the exit routine. Therefore, the system calls the exit routine only if at least one corresponding bit is 1 in both masks. If you do not specify SERVICEMASK, the default is that the system calls this exit routine.

For IFG_OPEN_START and IFG_CLOSE_START, the rightmost bit means that the system calls this exit routine only for opens of a DCB for a DASD data set. This excludes spooled data sets, subsystem data sets, and z/OS UNIX files and directories even though they might be on DASD. The only reason to specify a service mask for this case is to prepare for the future when IBM might define more bits to cover more kinds of data set. If you do not expect that you want the system to call your exit routine for other than DASD non-VSAM data sets, then you should set the rightmost mask bit to 1.

For purposes of the open and close exits, a z/OS UNIX directory is not regarded as DASD, but there is one exception. If it is part of a partitioned concatenation being opened with a BPAM DCB and the concatenation includes at least one PDS or PDSE, then each z/OS UNIX directory is represented by a pseudo DSCB. For example,

The DS1PDSEX bit is on. This bit normally means that the data set is an HFS file system but BPAM does not support opening such a data set. BPAM supports opening a z/OS UNIX directory.
The data set name in the DSCB and JFCB is not real.

For IGG_STOW_START, the rightmost bit means to invoke the exit routine for PDSs. The next higher bit means to invoke the exit routine for PDSEs. For example, if you want the system to call the exit only for PDSEs, code any of these values: `10, 00000010, 0000000000000010,` and so on.

For example, using the SETPROG operator command, the following adds the exit routine to the dynamic exit IFG_OPEN_START:

```
SETPROG EXIT,ADD,EXITNAME=IFG_OPEN_START,OPENSTRT
```

You can have multiple exit routines that are associated with each dynamic exit. If you do that without specifying FIRST or LAST, the order in which the system calls them is unpredictable.

You can replace already active dynamic exit routine without an IPL. For example, you can issue the SETPROG EXIT,DELETE operator command and the SETPROG EXIT,ADD operator command to replace a dynamic exit routine. For more information about the SETPROG command, see *z/OS MVS System Commands*.

For more information about the CSVDYNEX macro, see *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN*. For more information about the PROGxx PARMLIB member, see *z/OS MVS Initialization and Tuning Reference*.

Effects of concatenation

When the user opens a sequential concatenation, the close and open exits are called for each data set as the user makes the transition to the next data set. Therefore, the application program did not issue a CLOSE or OPEN macro for this transition.

Application programs cannot close a DCB under a task that differs from the task that opened the DCB, but the application program can read the data set in a task that differs from the task that opened the DCB. When making the transition to the next sequential data set or partitioned member, the system can close and reopen the DCB. In this case, the system calls the close and open exits under the reading task. The application program issued an access method macro (CHECK, GET, or FEOV) that caused the transition to the next data set or the application program issued an EOV macro for an EXCP DCB.

The DEB and the SVC exit parameter list always contain the address of the TCB that opened the data set.

When opening a partitioned concatenation, the exit receives a list of DSCBs for the data sets. The exit routines are not called separately for each data set.

Latent parameters

Your exit routine can use a *latent* parameter. Use 8 bytes to supply with the PARAM option when defining your exit routine to the exit. On each call to the exit, a copy of the first 4 bytes is in access register 0, and the second 4 bytes are in access register 1. It might be the address of a system-wide control block for that exit routine. The exit supplier can pass these 8 bytes with the PARAM keyword on CSVDYNEX REQUEST=ADD.

The parameter lists have no provision for an exit to pass information, such as the address of a work area, to another exit invocation. Your exit can pass information to another of your exits by using the name/token service. Call IEANTCR to create a name/token pair. In another exit, call IEANTRT to retrieve the information and IEANTDL to delete the information. For more information, see *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG*.

Registers on entry to the OPEN/CLOSE/EOV and access method dynamic exits

The registers are as follows:

Register
Contents

0
Not applicable.

1
Address of the OPEN/CLOSE/EOV and access method dynamic exits parameter list. See Table 14 on page 67Table 14 on page 67.

2-12
Not applicable.

13
Address of a 144-byte register save area.

14
Return address to the system.

15
Address of the entry point.

Registers on exit from the OPEN/CLOSE/EOV and access method dynamic exits

The registers are as follows:

Register
Contents

0,1
Unpredictable.

2-14
Same as at entry.

15
Return code. A return code of 0 from the exit routine means to continue the OPEN, STOW, or CLOSE. A return code of 8 means to fail the call to OPEN, STOW, or CLOSE. For OPEN, this results in a specific abend code 013-C1. For CLOSE, this results in a specific abend code 414-18. For STOW, it results in a return code 32 from the STOW macro. STOW has return codes that are smaller and larger than 32. The STOW reason code is the exit routine return code, but the user program cannot know which exit routine gave the return code. The user that submitted the job is aware because of the IEC997I message that the system issued to identify the exit routine.

The system might support new return codes in a future release.

If an abend occurs in the exit routine, it has no effect on the application program, and the system calls other exit routines that are defined for the exit point.

If an exit routine is disabled, a SETPROG command can be used to enable or replace it. The exit routines are not expected to change anything that is visible to the system or the user such as the DCB, OPEN parameter list, or JFCB.

General programming considerations

The system starts these exits with the FASTPATH=YES option. If your exit routine receives an abend, the ESTAE recovery routine of the system writes message IEC997I, retry the abend and start CSVDYNEX REQUEST=RECOVER. This is because the system can call any exit routines that the system has not yet called for this instance of OPEN, STOW, or CLOSE.

The default value for ABENDNUM for these exits is 2. The system tolerates this number of abends in a particular exit routine since IPL before the system changes its state to inactive. Therefore, that the system stops calling it. The system programmer might want to use message automation to handle this condition. When an exit routine is added to the system, the supplier can override ABENDNUM with ADDABENDNUM.

An abend in an exit routine does not affect the other exit routines that are defined for that exit or affect the application program.

The only fetch-protected area that the parameter list points to is the DCB of the user. Use the key in the IFGSVCPLCallerKey field.

The exit routine must be reentrant.

Environment during IFG_OPEN_START

The system compared the device class of the data set to the SERVICEMASK specification on the CSVDYNEX macro. This indicates if the exit routine is interested in this type of data set, which is DASD.

Before OPEN calls the exit, the following occurs:

- Verified the OPEN parameter list.
- Enqueued shared on SYSZTIOT. This is to serialize access to the DSAB chain, TIOT, and XTIOT. This also prevents invocation of dynamic allocation and unallocation in the address space. If the exit dequeues this resource, it endangers the information that the OPEN SVC gathered. For example, the data set being opened might be unallocated or even scratched and system control blocks might get overlaid.
- Found the control blocks for the specified DD name.
- Read the DSCBs.

If the system is not able to perform all of the above actions, the system does not call the exit routines for that DCB. If multiple DCBs are being opened with one call, the system calls the exit for other DCBs.

The SAF interface is not called to check security.

The system did not merge data set attributes to or from the data set label (DSCB) and did not call the DCB open exit routine of the program. Therefore, the DCB fields are not complete and there is no DEB (data extent block).

The system calls the exits only for DASD data sets that are being opened with a DCB. If multiple DCBs are being opened or closed, the exit is called independently for each.

It is possible to open a single-volume VSAM data set with an EXCP DCB.

Effects of concatenation

When the user opens a sequential concatenation, the close and open exits are called for each data set as the user makes the transition to the next data set. Therefore, the application program did not issue a CLOSE or OPEN macro for this transition.

Application programs cannot close a DCB under a task that differs from the task that opened the DCB. However, the application program can read the data set in a task that differs from the task that opened the DCB. When making the transition to the next data set, the system can close and reopen the DCB. Therefore, the system calls the close and open exits under the reading task. The application program issued an access method macro (CHECK, GET, or FEOV) that caused the transition to the next data set or the application program issued an EOV macro for an EXCP DCB.

The DEB always contains the address of the TCB that opened the data set.

When opening a partitioned concatenation, the exit receives a list of information about the data sets. The exit routines are not called separately for each data set.

Environment during STOW

The type of the data set (PDS or PDSE) is compared to the SERVICEMASK specification on the CSVDYNEX macro. This indicates if the exit routine is interested in this type of data set.

The parameter list is checked and the DEB is verified and locked. However, not much action is taken except to verify that the data set is partitioned.

Since SYSZTIOT is not held, do not examine other DSABs, TIOT, or XTIOT entries unless you issue ENQ for SYSZTIOT to prevent deletion of those control blocks.

Environment during CLOSE

The data set is verified to be open and the DEB is locked. Its type is compared to the SERVICEMASK specification on the CSVDYNEX macro. This indicates if the exit routine is interested in this type of data set.

Effects of concatenation

For information about the close exit being called under a task that differs from the task that opened the DCB, see "Effects of concatenation" on page 64 for the open exit.

During this transition for a sequential concatenation, the system might have performed the following for the DCB before calling IFG_CLOSE_START:

- If the data set is subject to QSAM HiperBatch, the caching backend processing is done.
- If the user coded FREE=CLOSE on the DD statement, the system is called dynamic unallocation even though the DEB remains.

Parameter list for OPEN/CLOSE/EOV and access method SVC exit routines

Table 14 on page 67 is the parameter list for OPEN, STOW, and CLOSE SVC installation exits:

Table 14. Parameter list for OPEN/CLOSE/EOV and access method SVC exit routines

Offset	Length, Bit Pattern, or Value	Name	Description
00 (X'00')		IFGSVCParmL	DSECT name.
	4	IFGSVCId	Identifier.
		IFGSVCID_Const	Constant 'SVCP' for IFGSVCID.
04 (X'04')	1	IFGSVCVER	Version (1).
05 (X'05')	1	IFGSVCPL_CallType IFGSVCPL_EarlyOpen IFGSVCPL_EarlyClose IFGSVCPL_EarlySTOW	Type of exit being called 1 Early open 2 Early close 3 Early stow
06 (X'06')	2	IFGSVCLen	Length of parameter list.
08 (X'08')	1	IFGSVCPLDataSetType	Data set type.
	.... .1..	IFGSVCPL_TYPE_DASD	(OPEN and CLOSE exits) DASD opened with DCB.
	.... ..1.	IFGSVCPL_TYPE_PDSE	(STOW Exit) PDSE data set.
	1	IFGSVCPL_TYPE_PDS	(STOWExit) PDS data set.

Table 14. Parameter list for OPEN/CLOSE/EOV and access method SVC exit routines (continued)

Offset	Length, Bit Pattern, or Value	Name	Description
09 (X'09')	1	IFGSVCPLCallerKey	Protection key of the caller of OPEN, STOW, or CLOSE in the high order four bits. It might differ from the task protection key that is in TCBPKF, but it should match the key of the caller of OPEN. Do not use key 0 to examine the DCB unless that is the value in IFGSVCPLCallerKey. This is to protect system integrity because an attacker can examine the registers at any time with a timer exit to learn what the exit picked up. The key of the user is in this parameter list. The key might differ from the task key, which is in TCBPKF. Therefore, the exit should not get storage in a user subpool, such as 0 to 127. An exception is that a request for subpool 0 while running in key 0 results in key 0 storage and therefore, does not violate system integrity rules.
10 (X'10')	1	IFGSVCPL_OPEN	OPEN or CLOSE macro options.
	 xxxx	IFGSVCPL_OPENBITS	Type of I/O accessing being done. The low order four bits correspond to four bits in the OPEN parameter list except that EXTEND or OUTINX are changed to OUTPUT or OUTIN respectively. Therefore, OPEN also changes the first two bits of JFCBIND2 to 10 (JFCMOD) to signify DISP=MOD (unless those two bits already are set that way). Make sure to test all four bits. Ignore the high order four bits.
	 0000	IFGSVCPL_INPUT	INPUT.
	 1111	IFGSVCPL_OUTPUT	OUTPUT or EXTEND.
	 0011	IFGSVCPL_INOUT	INOUT.
	 0111	IFGSVCPL_OUTIN	OUTIN or OUTINX.
	 0001	IFGSVCPL_RDBACK	RDBACK (read backwards) Valid on tape only.
	 0100	IFGSVCPL_UPDATE	UPDAT.
	.xxx		(OPEN and CLOSE exits) These bits represent OPEN and CLOSE macro options.
	.000		DISP.
	.001	IFGSVCPL_REREAD	REREAD.
	.010	IFGSVCPL_FREE	FREE (close only).
	.011	IFGSVCPL_LEAVE	LEAVE.
	.100	IFGSVCPL_REWIND	REWIND (meaningful only on close for tape).
	0....		Always zero.
11 (X'B')	1		Reserved.

Table 14. Parameter list for OPEN/CLOSE/EOV and access method SVC exit routines (continued)

Offset	Length, Bit Pattern, or Value	Name	Description
12 (X'C')	4	IFGSVCPL_DCB_ADDR	DCB address. Accessible in the protection key in IFGSVCPLCallerKey to protect system integrity. Do not use a different key even to test a bit. Do not use key 0 unless that is the value in IFGSVCPLCallerKey. Do not modify the DCB. It has no effect.
16 (X'10')	2	IFGSVCPL_DCB_ORIGIN	DCB origin (undefined bytes at start). This is the number of undefined bytes at the start of the DCB. Do not test DCB bytes that precede the origin. Normally, this is zero. It is nonzero if the programmer coded a value for the DEVD (device dependence) keyword that is not DA, the default. A non-DA value means that the DCB is not built for certain classes of device. For BSAM, BPAM, and QSAM, this always is zero. For BDAM, it is 16 unless the DCBH0 and DCBH1 bits are on. For EXCP information, see *z/OS DFSMSdfp Advanced Services*. A DCB origin of zero means that the first word in the DCB is valid. That word might point to the DCBE.
18 (X'12')	2	IFGSVCPL_DCB_LENGTH	DCB length (from offset 0, DCBDCBE). For QSAM, this is 96. For BSAM and BPAM it is 88. For EXCP, it depends on if the DCB has the OPTCD fields and the fields that identify appendages.
20 (X'14')	4	IFGSVCPL_DEB_ADDR	(STOW and CLOSE) DEB Address. The DEB is locked with the DEBCHK macro to ensure that it remains valid in the exit.
24 (X'18')	4	IFGSVCPL_UCB_ADDR	Captured or actual (31-bit) UCB address. A captured UCB has a 24-bit address of a 31-bit actual UCB. If the UCBVRDEV bit is on, that data set is a VIO data set, therefore, some UCB fields are missing.
28 (X'1C')	4	IFGSVCPL_DSAB_ADDR	DSAB address. The DSABTIOX bit tells whether the DSAB and this parameter list point to an XTIOT entry and not to a TIOT entry. The DSABCATM bit tells whether this data set is part of a concatenation. It might be a sequential or a partitioned concatenation. If the DD name in the TIOT entry or XTIOT is not blank, then this is the first data set in the concatenation. If the DSABLCAT bit is on, this is the last data set in the concatenation.

Table 14. Parameter list for OPEN/CLOSE/EOV and access method SVC exit routines (continued)

Offset	Length, Bit Pattern, or Value	Name	Description
32 (X'20')	4	IFGSVCPL_TIOT_ADDR	TIOT or XTIOT entry address as mapped by IEFTIOT1. If it is a TIOT entry, it contains UCB addresses. They might be captured (24-bit versions of actual 31-bit addresses). If it is an XTIOT entry, it does not contain UCB addresses. To find UCB addresses after the first one, the exit can use the IEFDDSRV macro. The last UCB addresses in the TIOT or XTIOT list for a multivolume data set might be for the dummy SMS UCB that represents slots that SMS completes if this program extends the data set.
36 (X'24')	4	IFGSVCPL_JFCB_ADDR IFGSVCPL_JFCB_ADDR	(OPEN and CLOSE) Address of a copy of the JFCB. See Notes® after this table. (STOW) Data set name address.
40 (X'28')	4	IFGSVCPL_DSCB_ADDR	(OPEN and CLOSE) DSCB Address. Points to an area mapped by the DPL DSECT to contain DSCBs. See Notes after this table.
44 (X'2C')	4	IFGSVCPL_WORKAREA_ADDR	Address of 256 bytes of work area for use by the exit routine. Contents are unpredictable.
48 (X'30')	8		Reserved.
56 (X'38')	8	IFGSVCPL_JOBNAME	Job name.
64 (X'40')	8	IFGSVCPL_STEPNAME	Job step name.
72 (X'48')	8	IFGSVCPL_PGMNM	Job step program name as obtained from the SCT, step control table.
80 (X'50')	8	IFGSVCPL_JOBID	Unique job identifier as obtained by the IAZXJSAB macro. This is for consistency with SMFJOBID in the SMF type 14/15 record.
88 (X'58')	1	IFGSVCPL_STOWREQ	STOW request.
	1	IFGSVCPL_STOWRA	STOW A (Add).
	2	IFGSVCPL_STOWRC	STOW C (Change).
	3	IFGSVCPL_STOWRD	STOW D (Delete).
	4	IFGSVCPL_STOWRR	STOW R (Replace).
	5	IFGSVCPL_STOWRI	STOW I (Initialize).
	6	IFGSVCPL_STOWRDISC	STOW DISC (Disconnect).
	7	IFGSVCPL_STOWRIFF	STOW IFF (IF and only if).
	8	IFGSVCPL_STOWRDG	STOW DG (Delete Generation).
	9	IFGSVCPL_STOWRRG	STOW RG (Replace Generation).
	10	IFGSVCPL_STOWRRECoverG	STOW RECOVERG (Recover Generation).
89 (X'59')	15		Reserved.
104 (X'68')	8	IFGSVCPL_STOWMName	STOW Member Name.
112 (X'70')	8	IFGSVCPL_STOWNMName	STOW New Member Name.
	120	IFGSVCPLLen	Length of parameter list.

Notes:

- The JFCB that IFGSVCPL_JFCB_ADDR points to starts with the data set name and includes up to five volume serial numbers for the data set. The JFCBNVOL byte contains the volume serial count, which might exceed the number of volumes that are identified in the JFCBVOLS field. This allows for SMS to extend this data set while it is allocated and being written by this program. It does not include volumes that might be added by a concurrently running program job. If the data set is open for output and is SMS-managed and the user program writes a large amount of data, the system might be able to extend JFCBNVOL and JFCBVOLS to add volumes.

  If the JFCBVLSQ field contains a value greater than 1, it is the volume sequence number that identifies the volume (such a 2, 3 or 4) that the user wants the access method to start with. It might be reading or writing.

  If the user read the JFCB and turned on the JFCNWRIT bit and issued the OPEN macro with TYPE=J, it means that the user does not want the system to update the system's copy of the JFCB. If the user also changed the data set name in the JFCB, this can be a system integrity problem because the system did not serialize on the new data set name. (It also must exist.) Therefore, if the application program is not authorized, open calls dynamic allocation for the new data set name and close later unallocates it. Consequently, the data set that is being opened might be different from the data set identified in the original JFCB.

  In these cases where the application program updated the data set name in the JFCB, it is the real name being opened but in the case of an authorized program, it might not be the real name for the close call. The warning about this is documented in the RDJFCB macro documentation in z/OS DFSMSdfp Advanced Services.

  If the data set that is being opened is the VTOC (volume table of contents) for the volume, the data set name is 44 bytes of X'04'. This is not a real data set name.
- In the STOW exit, do not examine other DSABs or TIOT or XTIOT entries unless you issue ENQ for SYSZTIOT to prevent deletion of those control blocks.
- IFGSVCPL_DSCB. In the open exit, this has the address of a chain of one or more DSCBs as mapped by the DPL DSECT defined by the IFGSVCPL macro. If it is a format 4 DSCB, then it is for a VTOC, volume table of contents, and it is the only DSCB. Otherwise, it is a format 1 or 8 DSCB, as mapped by the IECSDSL1 macro.

Table 15. Parameter list for DPL, DSCB

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	44	DPLDSName	Data set name. Do not test this field in the first DSCB. The field is not allocated. The next field still is at offset 44.
44 (X'2C')	96	DPLData	DSCB beginning with DS1FMTID (For more information about IECSDSL1, see *z/OS DFSMSdfp Advanced Services*).
140 (X'8C')	4	DPLNext	Address of next DPL or zero.
144 (X'90')	4	DPLUCB	Address of UCB.

In the close exit, this has the address of only one DSCB unless the data set is striped. If it is striped, the chain contains only format 1 or 8 DSCBs. The DSCB might be a format 4 DSCB, for the VTOC.

The DSCB address might be zero.

In the first entry in the chain mapped by the DPL DSECT, the content of the data set name is unpredictable and possibly unaddressable. Only the last 96 bytes of the first DSCB are present. The name of the first or only data set being opened or closed is in the JFCB. In the subsequent DPL entries, the full 140 bytes of the DSCB are present. In each DSCB, you can test the DS1FMTID byte to learn the type of DSCB.

For the OPEN call and not for CLOSE, there might be more DSCBs:

- If the first DSCB is a format 8, then it is followed by its format 9. An exception is that if the first DSCB is for extended format, the chain has format 1 and 8 DSCBs. It does not have format 9 or format 3 DSCBs.
- If the data set is not a PDSE and not extended format and it has more than three extents or more than two extents with a user label extent, then there is a format 3 DSCB.
- If the access method in the DCB is BDAM and the data set has multiple volumes, then all of the DSCBs for the data set on all of its volumes is on the chain.
- If the DSORG field in the DCB has PO (for partitioned organization), then the data set might be a concatenation. In that case all DSCBs for all data sets in the concatenation are read with these exceptions:
  - For each PDSE, any format 3 DSCBs is not read.
  - For any z/OS UNIX directory, the system builds a pseudo DSCB. If all of the DDs in the partitioned concatenation are for z/OS UNIX directories, then the exit is not called.

If the DS1DSGPS bit in the DS1DSORG field is on, it is a sequential data set. If the DS1LARGE bit also is on, it is a large format-sequential data set. If the DS1STRP bit is on (in addition to DS1DSGPS), it is an extended format-sequential data set. If neither of those two bits (DS1LARGE and DS1STRP) is on, it is a basic format-sequential data set.

If the DS1DSGPO bit in DS1DSORG is on, it is a partitioned data set (PDS or PDSE). If the DS1PDSE bit is off, it is a PDS. If the DS1PDSE bit is on but the DS1PDSEXbit is off, it is a PDSE. If both DS1PDSE and DS1PDSEX are on, it is an HFS data set (not an HFS or zFS file). These exits are not called in that HFS case.

If the DS1ENCRP bit is on, it is an encrypted data set. To learn the encryption information, you can call CSI, catalog search interface, to retrieve the ENCRYPTA field.

Passing information from an OPEN exit routine to a later routine

An exit routine cannot return something that the system passes to another exit point for that DCB because there is no correlation between types of exits. Therefore, if you provide exit routines for OPEN and CLOSE and another provider provides exit routines for OPEN and CLOSE, the system cannot make a connection between the two exits or exit routines.

The exit routine might use the name/token service (IEANTCR) to save information that is associated with the current task, address space, or system for later use.

Chapter 3. Tape Label Processing Installation Exits

This topic discusses installation exits for specialized tape processing. With these 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 Version 3 and Version 4 volumes.
- Selectively convert pre-Version 3 volumes to Version 3 or Version 4 volumes.

Table 16 on page 73 lists the replaceable modules available for tape label processing.

Table 16. Tape Label Processing Modules

Module Name	Description	When Available
IFG019LA	Label anomaly	Called from open and end-of-volume (EOV)
IFG019VM	Volume mount	Called from open and EOV
IFG019FV	File validation	Called from open and EOV
IFG019FS	File start on volume	Called from open and EOV
IFG055FE	File end on volume	Called from close and EOV
NSLOHDRI NSLEHDRI	Nonstandard label processing routines for input headers	Called from open and EOV
NSLOHDRO NSLEHDRO	Nonstandard label processing routines for output headers	Called from open and EOV
NSLETRLI	Nonstandard label processing routine for input trailers	Called from open
NSLETRLO NSLCTRLO	Nonstandard label processing routines for output trailers	Called from EOV and close
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
IFG0193C IFG0553C	Volume label editor routines for open and EOV	At open and EOV
IFG0193G	ISO/ANSI Version 3 and Version 4 label installation exits for volume access, file access, label validation, and label validation suppression	At open or EOV file access: after positioning to a requested data set
IEECVXIT	Replaced by IEAVMXIT	No longer available
IEAVMXIT	WTO, WTOR message processing installation exit	Label version conflict

Using dynamic versions of the Open/Close/End of Volume Tape exits

Previous to z/OS V2R2, any changes to Open/Close/End of Volume tape installation exit routines required an IPL before taking effect. Starting in V2R2, z/OS provides dynamic versions of these tape installation exits: Volume Mount, File Start, File Validate, File End, and Label Anomaly. You can modify the dynamic versions of these exits and implement them without an IPL. Appropriate sections of the 'IFGTEP OCE Tape exit parameter list' macro are used to map the dynamic exits' parameter lists. Each dynamic exit

receives the same parameter list as the corresponding non-dynamic exit. The dynamic tape installation exits are as follows:

1. OCE_VOLUMEMOUNT
2. OCE_FILESTART
3. OCE_FILEVALIDATE
4. OCE_FILEEND
5. OCE_LABELANOMALY.

The z/OS system defines the dynamic exits as AMODE 31, reentrant, and persistent until the next IPL. If an ABEND occurs in one of the exit routines associated with a dynamic exit, that exit routine becomes inactive while others stay active.

By default, each dynamic tape installation exit is associated with one IBM-supplied dummy exit routine:

- IFG019VM – Volume mount dummy exit routine
- IFG019FS – File start dummy exit routine
- IFG019FV – File validation dummy exit routine
- IFG055FE – File end dummy exit routine
- IFG019LA – Label anomaly dummy exit routine.

After the system is IPLed you can see the dynamic Open/Close/End of Volume exits by issuing an MVS DISPLAY command such as the following: D PROG,EXIT,EX=OCE_*

You can also see an exit's diagnostic data with an MVS DISPLAY command, such as the following example for the volume mount exit: D PROG,EXIT,EX=OCE_VOLUMEMOUNT,DIAG

You can keep or delete the IBM-supplied dummy exit routines, and can add your own exit routines, using MVS system commands. For more information, see Using Dynamic Exits in *z/OS MVS System Commands*. You can also use a PROG*xx* member of SYS1.PARMLIB. If an exit routine for a tape dynamic exit is defined in PROGxx, then it is added to the exit during IPL and is visible using a DISPLAY command thereafter. If no exit routines are defined in PROGxx, then the IBM-supplied dummy exit routine is added to the exit.

You can associate multiple exit routines with one dynamic installation exit. Each defined exit routine must be a separate load module in AMODE 31 and reentrant. Changes to the exit routines become available without IPL. If multiple exit routines are defined for one installation exit, they are called one at a time.

Note: The mapping macro for each exit defines which fields are read-write, and the other fields must not be changed.

Exit-specific details for the dynamic exits, including return code handling for multiple exit routines, are given in the sections for each exit in this chapter.

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 and in the LPALST concatenation.

Open, Close, End-of-Volume Tape Management Exits

The open, close, and end-of-volume (EOV) tape management exits allow tape management systems to avoid changing system control blocks or issuing channel programs against tapes in order to maintain a tape inventory.

During the course of opening, processing and closing a tape data set, the system always calls the File Validation, File Start on volume and File End on Volume. Open or EOV calls the file validation exit for each volume regardless of whether it already was mounted or verified.

Many conditions affect whether various exits are called and their sequence. During one call to Open or EOV, the following is a typical sequence:

1. File end on volume exit (if EOV)
2. Volume mount exit for volume verification (if volume was not previously mounted and verified)
3. Volume mount exit for volume security (if volume was not previously mounted and verified)
4. File validation exit (if labels exist)
5. Volume mount exit for volume write function (if the tape has standard labels, the user is writing and the file is the first file on the volume)
6. File start on volume exit

Installing the Open, Close, and EOV Exits

The open, close, and EOV exits are always present and link edited with open, close, and EOV modules. IBM supplies dummy versions of these exits. They have no effect and you can replace them by using the SMP/E USERMOD function. For the volume mount exit, the dummy version always gives the return code that means to continue.

For additional information, see “Replacing an Existing Exit” on page 3.

Characteristics of the Open, Close and EOV Exits

These exits run under the following conditions:

- Supervisor state
- Protection key five
- TCB mode
- 31-bit addressing mode

You must save and restore registers other than register 15.

Registers on Entry to the Open, Close and EOV Exits

Register
Contents

1
Address of Open, Close, and EOV main parameter list which includes a pointer to the appropriate function-specific parameter list

13
Standard 18 word save area

14
Address of exit caller

15
Address of exit entry point

Open, Close, and EOV Main Parameter List

IFGTEP (tape exit parameters) maps the main parameter list for all of the open, close, EOV tape management exits. It also maps the function-specific parameter lists. The parameter lists are in protection key five.

From time to time IBM might add a field to the end of any of the parameter lists for these exits. IBM might also retrofit those changes to earlier releases via a PTF. You can compare the parameter list description

in this book with the actual mapping in the IFGTEP macro to learn which is at the higher level. Your exit routine can also test the content of the length field for the parameter list to see the actual length when executing. This allows you to have one level of your code be able to run on multiple levels of the system with different lengths of the actual parameter list. In other words this allows your code to detect whether the last field in the mapping macro actually exists. Your exit should also test the contents of the version field. If the content is not 1, it means that IBM has made an incompatible change to the parameter list. Therefore your exit routine should not be running on that system. You can change your code to handle multiple versions of the parameter list or to handle only the new version.

The IFGTEP macro supports positional parameters in any order that specify which parameter list or lists to expand. These values can be coded in any combination to get the indicated parameter list:

- MAIN - Main parameter list (see Table 17 on page 76)
- LABAN - Label anomaly parameter list (see Table 18 on page 85)
- VOLM - Volume mount parameter list (see Table 19 on page 91)
- FILEV - File validation parameter list (see Table 20 on page 95)
- FILES - File start on volume parameter list (see Table 21 on page 98)
- FILEE - File end of Volume parameter list (see Table 22 on page 100)

The format of the main parameter list is:

Table 17. Open, Close, EOV Main Parameter List.

Offset	Length or Bit Pattern	Name	Description
00 (X'00')		TEPM	Beginning of the parameter list
00 (X'00')	8	TEPMID	Control block identifier ("TEPMAIN")
8 (X'08')	4	TEPMLEN	Length of main parameter list
12 (X'0C')	1	TEPMVER	Version of parameter list (1)
13 (X'0D')	3		Reserved
16 (X'10')	1	TEPMFUNC	Calling function (binary numeric): 1. OPEN. The call can be during like or unlike concatenation. 2. EOV or FEOV (Note that for QSAM output the caller of EOV might be CLOSE because it reached the end of the tape when writing out the last buffers. EOV will soon continue on a new volume with the file start on volume exit and CLOSE will resume with the file end on volume exit.) The call might be for the first volume during like concatenation. 3. CLOSE with TYPE=T. 4. CLOSE without TYPE=T.

Table 17. Open, Close, EOV Main Parameter List. (continued)

Offset	Length or Bit Pattern	Name	Description
17 (X'11')	1	TEPMOPENOP	First option for the DCB in the OPEN parameter list. Valid during all calls to the exits with one exception. If the application program used OPEN TYPE=J and suppressed the JFCB update, these bits will show EXTEND and OUTINX as OUTPUT and OUTIN during EOV and CLOSE. Note that all options except UPDAT are valid on tape.
	 0000		INPUT
	 0001		RDBACK
	 0011		INOUT
	 0110		OUTINX
	 0111		OUTIN
	 1110		EXTEND
	 1111		OUTPUT
18 (X'12')	2	TEPMCONC	Concatenation number. First or only data set is 0.
20 (X'14')	2	TEPMVSEQ	Volume sequence number. The first volume is 1.
22 (X'16')	6	TEPMVOL	Requested volume serial number, blank if output nonspecific or output VOL=REF and it is still not resolved. Not necessarily in JFCB, might be in JFCBX. First byte will not be X'FF', which would mean it is not resolved. Bit TEPMSYNV means that the system synthesized the volume serial number from other than a volume label. The volume mount exit can set bit TEPMNEWLAB to mean that the exit has changed this field to substitute the volume serial.

Table 17. Open, Close, EOV Main Parameter List. (continued)

Offset	Length or Bit Pattern	Name	Description
28 (X'1C')	1	TEPMFLAG1	Flags
	1...	TEPMSMS	SMS-managed volume.
	.1..	TEPMTLDS	Tape Library Dataserver.
	..1.	TEPMCHKPT	Data set is a checkpoint data set.
	...1	TEPMASCII	The system translated the label from ASCII to EBCDIC or will translate from EBCDIC to ASCII. On input the system translates only if the label begins with ASCII HDR, EOV or EOF.
	.... 1...	TEPMSAB	System issued an ABEND for this data set. Its ABEND completion code is in TEPMABCODE. Before the system calls the exit, this bit means that the system issued an abend for this data set. This might happen when either recovery failed during Open, Close or End of Volume, or force close is in process. After the exit returns, this bit also can mean that the exit issued return code 16 for this DCB. In either case the completion code is in TEPMABCODE and the reason code is in TEPMRSNCODE.
	.... .1..	TEPMAB	The current task is abending apparently because of a reason that is not related to this tape. Its code is shown in field TEPMABCODE.
	.... ..1.	TEPMLWRIT	Current operation. "1" means that the caller will be writing or rewriting labels. If the caller is OPEN, then this bit will be zero if the OPEN option is INPUT, INOUT, RDBACK, EXTEND or OUTINX or if the OPEN option is is OUTPUT or OUTIN with DSP=MOD. If the caller is OPEN, then this bit will be one for OUTPUT or OUTIN without DISP=MOD. If the caller is EOV, then a zero means that the user's last I/O was to read. Either it read a tape mark or the user issued an FEOV or EOV macro. If the caller is EOV, then a one means that the user's last I/O was to write. Either it encountered the end of the tape or the user issued an FEOV or EOV macro.
	1	TEPMACCESS	Access intent. This will be 0 to indicate "read-only" if the OPEN option was for INPUT or RDBACK and it will be 1 to indicate "write" for any other OPEN option.
29 (X'1D')	1	TEPMFLAG2	Flags
	1...	TEPMSYNV	Volume serial number in TEPMLABEL was synthesized by the system, possibly from sense bytes.
	.1..	TEPMVFRY	Volume has been verified.
	..1.	TEPMLBS	The block size value is contained in TEPMBLKS

Table 17. Open, Close, EOV Main Parameter List. (continued)

Offset	Length or Bit Pattern	Name	Description
30 (X'1E')	1	TEPMFLAG3	Flags
	1...	TEPMLABAN	Label anomaly exit being called
	.1..	TEPMVOLM	Volume mount exit being called
	..1.	TEPMFILEV	File verification exit being called
	...1	TEPMFILES	File start of volume exit being called
	.... 1...	TEPMFILEE	File end on volume exit being called
	.... .1..	TEPMATL	Allocated device is in the automated tape library
	.... ..1.	TEPMMTL	Allocated device is in the manual tape library
	1	TEPMWRIT	Write channel program issued to tape in OPEN, EOV or CLOSE.
31 (X'1F')	1	TEPMRECTK	Recording technology
	0000 0000		Unknown
	0000 0001		18-track
	0000 0010		36-track
	0000 0011		128-track
	0000 0100		256-track
	0000 0101		384-track
	0000 0110		EFMT1
	0000 0111		EFMT2
	0000 1000		EEFMT2
	0000 0101		EFMT3
	0000 0110		EEFMT3
	0000 1011		EFMT4
	0000 1100		EEFMT4
32 (X'20')	1	TEPMFLAG5	Flags
	1...	TEPMWORM	WORM (write once, read multiple) tape mounted.
	.1..	TEPMCRYP	On - the key-related fields have data. Off - the key-related fields contain binary zeroes.
	..xx xxxx	Reserved	
33 (X'21')	1	TEPMFLAG6	Flags set by the exit
	1...	TEPMNEWLAB	Exit supplied a volume label. Valid only for label anomaly and volume mount exits.
	.1..	TEPMSCRTCH	Original volume request was for a nonspecific volume
	..xx xxxx	Reserved	Set to zeros

Table 17. Open, Close, EOV Main Parameter List. (continued)

Offset	Length or Bit Pattern	Name	Description
34 (X'22')	1	TEPMMEDT	Media type
	0000 0001		Media1
	0000 0010		Media2
	0000 0011		Media3
	0000 0100		Media4
	0000 0101		Media5
	0000 0110		Media6
	0000 0111		Media7
	0000 1000		Media8
	0000 1001		Media9
	0000 1010		Media10
	0000 1011		Media11
	0000 1100		Media12
	0000 1101		Media13
35(X'23')	1		Reserved
36 (X'24')	4	TEPMABCODE	Task is abending with this code, in form *xxyyyzzz*, where *yyy* is system ABEND code that is normally displayed in hex and *zzz* is user ABEND code. Valid only if TEPMSAB or TEPMAB is on. Same format as TCBCMP.
	1	TEPMCMPF	Flags
	1...	TEPMCREQ	A DUMP was requested.
	.1..	TEPMCSTEP	STEP coded on ABEND macro.
	..1.		Reserved.
	...1		Reserved.
	.... 1...		Reserved.
	.... .1..		Reserved.
	.... ..1.		Reserved.
	1		Reserved.
	3	TEPMCMP	System completion code in first 12 bits. User completion code in last 12 bits.

Table 17. Open, Close, EOV Main Parameter List. (continued)

Offset	Length or Bit Pattern	Name	Description
40 (X'28')	4	TEPMDCB	Address of DCB copy in protection key 5 storage. Can include DEN, TRTCH, OPTCD=B, block count on current volume, MACRF (indicates BSAM, QSAM or EXCP, length of EXCP DCB and whether DCBBLKCT is valid), BUFNO, NCP, address of DCBE, BLKSIZE, LRECL and RECFM. The block count normally is zero at file start on volume except when the data set is open for read backwards or is being extended. For EXCP, DCBMACRF includes an indicator as to whether the user program is keeping a block count in DCBBLKCT (REPOS=Y). If it is off, DCBBLKCT is not valid. It is invalid for any of these reasons: • The EXCP user did not code REPOS=Y on the DCB macro. • The BSAM user issued the CNTRL FSM macro and the system was not able to reestablish the block count later. • While open for read backward, the user issued FEOV or CLOSE before reading the tape mark. • Extending (DISP=MOD or OPEN EXTEND or OUTINX) an unlabeled or NSL tape. During OPEN, this DCB does not have a pointer to the DEB. In the current release, the copy of the DCB points to the user's DCBE if one was supplied. For system integrity reasons refer only to the DCBE in the user's key.
44 (X'2C')	4	TEPMUCB	Address of the captured UCB. At the call to the file-end-on-volume exit, the UCB should have volume error statistics because the tape was mounted or SVC 91 cleared them. These statistics differ in format between reels and cartridges. The UCB also tells whether a cartridge stack loader is attached.
48 (X'30')	4	TEPMJFCB	Address of JFCB copy. Includes data set name, required label type, requested expiration date, requested volume serials (any that start with X'FF' have not yet been resolved), approximate limit on the volume count, density, file sequence number in the volume group, TRTCH, FREE=CLOSE. If the OPEN option was EXTEND or OUTINX and at least one volume serial was specified, then DISP in the JFCB has been changed to MOD. If DISP=MOD was coded and the OPEN option was EXTEND or OUTINX, but no volume serial was specified, then OPEN changes DISP=MOD to DISP=OLD. The exit should not modify the JFCB.
52 (X'34')	4	TEPMDSAB	Address of DSAB. To get the DDname the exit can get the TIOT entry address from the DSAB. The name might be blank, which means concatenation.

Table 17. Open, Close, EOV Main Parameter List. (continued)

Offset	Length or Bit Pattern	Name	Description
56 (X'38')	4	TEPMLABEL	Address of label (80 bytes). If the address is 0 and the tape is SL or AL, then the data set was being read and the user issued FEOV or CLOSE without reading the tape mark. As much of the header label 1 or trailer label 1 for read backwards as the system could read up to 80 bytes. **Note:** The creation date and expiration date in the header label might differ from that in the trailer label. This might be because of DISP=MOD or OPEN EXTEND. The System code field (13 bytes) is available at beginning of data set. It tells whether an AL tape meets IBM's definition of certain label 2 fields. For the volume mount exit and label anomaly exit this is the first block on the volume no matter what label type was requested except for BLP. The label anomaly exit might build a label in this area. For the file validation exit the label is HDR1, EOV1 or EOF1. For the file start on volume exit and the file end on volume exit no label is available because OPEN, CLOSE, EOV sometimes does not have one. For example this might be because it processed user labels or it wrote a message to the operator.
60 (X'3C')	4	TEPMLLEN	Length of the block that was read. Zero means either that no data was read because of an I/O error or it means that a tape mark was read.
64 (X'40')	4	TEPMTEP	Address of function-specific parameter list
68 (X'44')	4	TEPMSENSE	Address of sense bytes 0 and 1 from an IOB. If X'10FE', then the system was not able to obtain sense bytes.
72 (X'48')	6	TEPMMTVOL	Mounted volume serial
78 (X'4E')	6	TEPMEXVOL	External volume serial
84 (X'54')	2	TEPMTDSI	TDSI (tape dataset info) from JFCB
86 (X'56')	2		Reserved
88 (X'58')	4	TEPMVOL1	Address of original VOL1 label. Initialized when OPEN or EOV issues a loadpoint read for volume verification.
92 (X'5C')	4	TEPMHDR1	Address of original HDR1 of first data set on the volume. Initialized when OPEN or EOV issues a loadpoint read for volume verification.
96 (X'60')	8	TEPMBLKS	Block size value
104 (X'68')	4	TEPMCAPM	Media capacity (MB) of the mounted volume on device type 3490 and higher tape drive technologies.
108 (X'6C')	4	TEPMCAPP	Reserved for future partition capacity. Currently same value as TEPMCAPM.
112 (X'70')	12	TEPMWWID	WORM World Wide Volume id
124 (X'7C')	2	TEPMWMC	WORM write mount count
126 (X'7E')	4	TEPMRSNCODE	Abend reason code. Valid only if TEPMSAB or TEPMAB is on.

Passing Information Between Exits

The parameter lists have no provision for an exit to pass information, such as the address of a work area, to another exit invocation. Your exit can pass information to another of your exits by calling IEANTCR to

create a name/token pair. In another exit call IEANTRT to retrieve the information and IEANTDL to delete the information. See *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG*.

You choose a 16-byte name for your 16 bytes of information. For example, you might choose a 12-byte constant and the 4-byte UCB address for the name. It is better not to choose a task level name because any of the exits can be called in various tasks. To avoid the possibility of permanently lost storage, use a home address space name.

Remember that the UCB address in TEPMUCB might be duplicated in another address space and represent a different UCB and device in the other address space.

Remember also that the device might be closed, varied offline and become undefined. In that case, a system-level name/token pair would be lost until IPL unless a device is defined and happens to get a UCB at the same virtual address. It is recommended that you delete your name/token pair in the last exit that uses it.

Open, Close, and EOV Return Codes

If an exit gives an unsupported return code, the system writes a message that displays the name of the exit and the invalid return code. The system then abends the job.

If an exit gives return code 16, then the exit routine must put the intended abend code in TEPMABCODE and the reason code in TEPMRSNCODE. If multiple exit routines give return code 16, only the first ABEND will be issued. The system terminates the task without calling the DCB abend exit.

Label Anomaly Exit (IFG019LA)

The system calls this exit when it has found one or more certain unusual conditions on a tape. With the dummy version of this exit most of these conditions cause OPEN or EOV to go to the label editor routine or to issue ABEND (possibly after calling the user's DCB abend exit). The label editor routine is documented in “Volume Label Verification and Volume Label Editor Routines” on page 118. IBM recommends using the label anomaly exit instead of a label editor routine for the reasons stated in the label editor sections.

The label anomaly exit will receive a UCB address from TEPMUCB in IFGTEP, from DEBUCBA and from TIOEFSRT. All three sources will allow a 31–bit UCB address. TEPMICB will sometimes be an uncaptured 31–bit address. If the DEB31UCB bit is on, the UCB address and modeset byte will be different as described in IEZDEB. If they use the DCBTIOT field to find the TIOT entry address, they should be changed to use TEPMDSAB to point to the DSAB, which points to the TIOT entry or XTIOT.

These are the situations under which the system calls a label editor routine and the label anomaly exit:

- **Label type conflict**. User opening for OUTPUT or OUTIN and the requested label type (SL, AL, NL, NSL) differs from the actual label type. If the exit gives return code 4 and turned off this anomaly bit and the requested type is SL or AL, then the exit has built a volume label in EBCDIC in the supplied area. OCE will verify that it begins with VOL1. The requested label type is in the JFCB. The exit can determine the actual label type by examining the first four bytes of the area to which TEPMLABEL points. It is in the main parameter list. The area is from the first block on the tape after load point. If it is not "VOL1", then the tape is not labeled or it has nonstandard labels. If it is "VOL1", then the label is SL or AL. To determine which, the exit can test bit TEPMASCII in the main parameter list.
- **Density conflict**. User opening for OUTPUT, OUTIN, INOUT, OUTINX or EXTEND and the existing density differs from the requested density. The volume is mounted on a dual density drive. Return code 4 with this anomaly bit off means to use the requested density if writing file 1 or the existing density if writing a later file.
- **Volume serial conflict**. The user is opening for OUTPUT or OUTIN and the mounted volume serial differs from the requested volume serial. With return code 4 and this anomaly bit off this exit can supply a new volume label.
- **Label version conflict**. User opened for OUTPUT on an AL tape and the volume label is not for Version 3 or Version 4. With return code 4 and this anomaly bit off this exit can supply a new volume label.
- **Track conflict**. The tape cannot be read because of incompatibility between the tape and the drive, not because of some other I/O error. The exit can test the CSW status bytes and the sense bytes

in the main parameter list. See TEPMSENSE. One case is 36-track tape mounted on 18-track drive. TEPATRKC on indicates this condition. The volume serial number might have come from the sense bytes, in which case TEPMSYNV is on. A PARMLIB option controls what open, close, EOV does. For example, if DEVSUPxx parameter VOLNSNS=YES is specified, the OPEN or EOV label editor routines will rewrite the VOL1 label using the volume serial number obtained from the drive sense bytes. The LABAN exit can give permission to use the tape, which means that the system can destroy the labels without reading the labels. This means the exit is taking responsibility for such things as security and the files being expired. The exit can provide all the information to be written in the new volume label. The system will still call RACF for the volume but without the header label information. The IBM—supplied Label Editor routine, OMODVOL1 or EMODVOL1 will delete the RACF definition for the volume. The exit can bypass this delete of the RACF TAPEVOL profile when the system has set flag bit TEPABYRACF in byte TEPAFLAG3 indicating the bypass option is active. Setting flag bit TEPAEXSKIP in byte TEPAFLAG3 when TEPABYRACF has been set by the system, and returning return code 4 results in bypassing the delete of the RACF TAPEVOL profile.

- **I/O error**. I/O error while reading volume label for output tape and RACF tape protection is not active. If the exit gives return code 4 with this anomaly bit off, the system tries to create a new label.

Those conflicts can also be handled by OMODVOL1 and EMODVOL1. Any conflicts you do not handle in the label anomaly exit are passed to OMODVOL1 or EMODVOL1 to handle. The following are not supported by OMODVOL1 or EMODVOL1:

- **Unsupported cartridge length** Cartridge tape length exceeds IBM's maximum supported length for this drive. This call to the exit is for information only; the tape will be rejected even if the exit turns off the bit. This means to reject the tape if it is a nonspecific request or to issue ABEND if it is a specific volume request.
- **Writing unsecure checkpoint** Writing a checkpoint to an unsecure tape, indicated by bit TEPACHKPT in byte TEPAFLAG1. Return code 8 from the exit means to reject the tape. This means to reject the tape if it is a nonspecific request or to issue ABEND if it is a specific volume request. Return code 8 can also mean that the volume is not secure. This code is like a reply of "NO" to the messages.

  Return code 4 from the exit with the bit still on will result in one of these messages to the operator:

  IEC254D SHOULD *jjj* USE (*ddn*,*uuu*,serial) FOR CREATING A NEW CHECKPOINT DATA SET
  IEC255D jjj IS (*ddn*,*uuu*,serial) A SECURE CHECKPOINT VOLUME

  Return code 4 and the TEPACHKPT bit set off results in the system assuming the tape is secure. This is like a reply of "YES" to the messages. If you write a non-checkpoint data set as the first data set on a checkpoint secure volume, then bit TEPANOCKPT is set in the exit's parameter list. When the exit returns a return code of 4, and when bit TEPANOCKPT is still on and it is not an Automatic Tape Library (ATL), then the system issues a scratch volume WTOR IEC255D. The system writes an ATL scratch volume's HDR2 and bypasses WTOR IEC255D. If the exit resets the bit TEPANOCKPT, then the system assumes that the volume is not checkpoint secure, does not issue message IEC255D, and behaves as if the operator had replied "NO" to message IEC255D.
- **Overwriting ISO/ANSI user volume labels** User volume labels or ISO/ANSI Version 4 volume labels 2-9 are about to be overwritten.

  Return code 4 with the bit on results in message IEC704A.

  Return code 4 with the bit off results in the labels being overwritten.

  Return code 8 means to reject the volume. This means to reject the tape if it is a nonspecific request or to issue ABEND if it is a specific volume request.
- **Writing volume label** Volume label information is needed to write the label.

  Return code 4 with the bit on results in message IEC704A.

  Return code 4 with the bit off means that the exit has supplied the volume label. For volume serial number conflicts between the volume mounted and the volume requested, and for label type conflicts when an NL scratch volume is requested and a labelled tape is mounted, the volume serial number supplied by the exit will be used to either rewrite the volume label or to catalog the NL volume. In the

latter case the system will not generate an Lnumber. In order for the label anomaly exit to supply a volume serial the following steps are required:

1. Reset TEPASERIAL (volume serial conflict) or TEPALTYPE (label type conflict) in byte TEPAFLAG1.
2. Set bit TEPMNEWLAB in byte TEPMFLAG6.
3. Provide valid volume serial number in field TEPMVOL.

See “Label Anomaly Exit (IFG019LA) Return Codes” on page 89.

Return code 8 means to reject the volume. This means to reject the tape if it is a nonspecific request or to issue ABEND if it is a specific volume request.

- **Multivolume tape conditions** When reading an unexpected volume sequence number, a missing final volume, or a missing first volume (reading forward or backward) was encountered during multivolume tape processing. With return code C and the corresponding anomaly bit on, this exit issues a message and abends the job.

  With DFSMSrmm, the missing or out of sequence conditions are normally resolved, which results in a complete volume list for the multivolume being processed. If you want to process only the original incomplete or out of sequence volume list, you can use a label anomaly exit to ensure the original volume list is used. The following example uses IFG019LA to ensure the original volume list is processed unaltered.

```
* Label anomaly exit to allow missing or out-of-order tape volumes.
IFG019LA CSECT ,
IFG019LA AMODE 31
IFG019LA RMODE ANY
R1       EQU   1
R3       EQU   3
         SAVE  (14,12)
         USING TEPM,R1
         L     R3,TEPMTEP
         USING TEPA,R3
         TM    TEPANMLY2,TEPASEQ     Vol list out of seq?
         JNO   CHKFML                No
         OI    TEPAFLAG3,TEPAOUTSEQ  Don't alter vol list
         J     EXITEXIT              Exit
CHKFML   TM    TEPANMLY2,TEPAFML     Missing last volume?
         JNO   CHKBML                No
         OI    TEPAFLAG3,TEPARDFMLV  Don't alter vol list
         J     EXITEXIT              Exit
CHKBML   TM    TEPANMLY2,TEPABML     Rdback missing last
         JNO   CHKFMF                No
         OI    TEPAFLAG3,TEPARDBMLV  Don't alter vol list
         J     EXITEXIT              Exit
CHKFMF   TM    TEPANMLY2,TEPAFMF     Missing first volume?
         JNO   EXITEXIT              No
         OI    TEPAFLAG3,TEPARDFMFV  Don't alter vol list
         DROP  R3
EXITEXIT L     14,12(,13)            Restore return address
         LA    15,4                  Set continue return code
         LM    1,12,24(13)           Restore callers registers
         BR    14                    Return
         IFGTEP MAIN,LABAN
         END
```

For a particular OPEN or EOV this exit can be called more than once for various reasons.

Label Anomaly Exit (IFG019LA) Function-Specific Parameter List

The following figure is the format of the function-specific parameter list for the Label Anomaly Exit. See Table 17 on page 76 for a description of the main parameter list.

Table 18. Label Anomaly Exit Parameter List

Offset	Length or Bit Pattern	Name	Description
00 (X'00')		TEPA	Label anomaly exit parameter list
00 (X'00')	8	TEPAID	Control block identifier

Table 18. Label Anomaly Exit Parameter List (continued)

Offset	Length or Bit Pattern	Name	Description
08 (X'08')	4	TEPALEN	Length of parameter list
12 (X'0C')	1	TEPAVER	Version of parameter list (1)
13 (X'0D')	3		Reserved
16 (X'10')	1	TEPAFLAG1	Flags. Bits that tell the reason for the problem on the tape. More than one bit can be on. If the exit does not find any bit on that represents a problem it was designed to handle, then it should return with the return code that causes no action.
	1...	TEPALTYPE	Label type conflict
	.1..	TEPADENS	Density conflict
	..1.	TEPASERIAL	Volume serial conflict
	...1	TEPAVERS	Label version conflict
	.... 1...	TEPAIOE	I/O error while reading first block and RACF protection is not active
	.... .1..	TEPANCAP	Drive not capable, such as a 36-track tape on an 18-track drive
	.... ..1.	TEPACLEN	Unsupported cartridge length
	1	TEPACHKPT	Writing a checkpoint to an unsecure volume
17 (X'11')	1	TEPAFLAG2	Flags. Bits that tell the reason for the problem on the tape. More than one bit can be on.
	1...	TEPAUSRVLAB	ISO/ANSI user volume labels or VOL 2-9 labels are about to be overwritten.
	.1..	TEPAVINFO	Volume label information is needed to write the volume label.
	..1.	TEPANOCKPT	Writing a non-checkpoint data set to checkpoint volume
	...1	TEPANINIT	Volume on a 3590 device is not formatted
	.... 1...	TEPAUCKPS	When on, instructs O/C/EOV to unconditionally resolve checkpoint conflicts on this scratch volume
	.... .1..	TEPAUCKPP	When on, instructs O/C/EOV to unconditionally resolve checkpoint conflicts on this private volume
	.... ..1.	TEPATRKC	Track conflict. A 36-track tape volume mounted on a 18-track only tape drive.

Table 18. Label Anomaly Exit Parameter List (continued)

Offset	Length or Bit Pattern	Name	Description
18 (X'12')	1	TEPAFLAG3	Flags. Bits that tell the reason for the problem on the tape. More than one bit can be on.
	1...	TEPABYRACF	Bypass RACF RACDELETE option is active.
	.1..	TEPAEXSKIP	When on, instructs O/C/EOV to not do RACDELETE.
	..1.	TEPAABEND	Job abends when return code is 8.
	...1	TEPAOUTSEQ	Unexpected tape volume sequence number – next volume to process has a sequence number that is not as expected based on the current volume's sequence number. Job abends with message IEC709I when return code is C.
	.... 1...	TEPARDFMLV	Last tape volume missing – processed last volume specified but it had an EOV label, not an EOF label. Job abends with message IEC710I when return code is C.
	.... .1..	TEPARDBMLV	When reading backward, the last specified volume for the standard labeled data set is not the actual last volume because it does not end with an EOF label. Job abends with message IEC711I when return code is C.
	.... ..1.	TEPARDFMFV	First volume missing when reading forward – the first volume in the list does not begin with logical volume sequence number 1. Job abends with message IEC712I when return code is C.
	1		Reserved
19 (X'13')	1	TEPALVR	Current ISO/ANSI version of the tape
20 (X'14')	2	TEPACSWST	"CSW" status bytes that the system received when reading the label. Unit exception will show if the system read a tape mark,and the length of the label will be zero. The I/O errors will show one of these: • Drive incapable. (For example if it is a 36-track tape on an 18-track drive, then the system might still be able to move the actual volume serial from the sense bytes to the place in the input area, which will otherwise be blank.) If the system was not able to read the volume label because it is a 36-track tape on an 18-track drive, then the exit can give permission to rewrite the labels without getting all the volume label contents or reading the header labels. This means the exit is taking responsibility for the data sets being expired. The exit will provide the volume serial and the owner information. The system will still call RACF for the volume but without the header label information. • Data check • Other
22 (X'16')	2	TEPANMLY	Preserved initial TEPAFLAG1, 2, and 3 values.
	1	TEPANMLY2	
	...1	TEPASEQ	IEC709I vols out of sequence
	.... 1...	TEPAFML	IEC710I read forward missing last volume
	.... .1..	TEPABML	IEC711I read backward missing last volume
	.... ..1.	TEPAFMF	IEC712I read forward missing first volume
	5	TEPANMLY3	

Table 18. Label Anomaly Exit Parameter List (continued)

Offset	Length or Bit Pattern	Name	Description
30 (X'1E')	4	TEPAVOLLST	Pointer to an optional volume list initialized by the exit. This is a list of the remaining volumes after the volume passed back from the exit in TEPMVOL, to the end of the multivolume data set being read. TEPMVOL contains the next volume in sequence to read in order to recover from an IEC709I, IEC710I, IEC711I or IEC712I condition. The exit selects a recovery option by returning a 04 return code, turning off the anomaly indicator in the parameter list, and supplying the next volume to read in TEPMVOL. This optional volume list ensures that all the remaining volumes will be processed in the same sequence order that they were created. This is an optional pointer to storage containing all zeros. The LABAN exit can initialize this storage with a list of the remaining volumes after the volume passed back from the exit in TEPMVOL, to the end or last volume of the multivolume data set being read. The exit does not need to indicate the end of the volume list. Instead a null volser (that is, with all zeros for six bytes) marks the end of the list.
34 (X'22')	6	TEPAPREVL	Volume serial of previous volume. Defined only if caller is EOV and the current volume is for the same DD statement as the previous volume.

Controlling the label anomaly exit routine through the dynamic exits facility

IBM has defined the Label Anomaly installation exit to the dynamic exits facility. You can refer to the dynamic exit by the name OCE_LABELANOMALY. You can use the EXIT statement of the PROGxx parmlib member, the SETPROG EXIT operator command, or the CSVDYNEX macro to control this exit and the exit routines of the exit.

The system attempts to add exit routine OCE_LABELANOMALY, unless an exit routine has been associated with OCE_LABELANOMALY by PROGxx or SETPROG.

If you have associated exit routines with OCE_LABELANOMALY the system does not use the default exit routine. In this case, if you require an exit routine of the default name, you must explicitly add the default exit routine to PROGxx.

To limit the number of times the exit routine ends with an abend, before the exit routine becomes inactive, you can use the ADDABENDNUM and ABENDCONSEC parameters on the CSVDYNEX REQUEST=ADD macro, or the ABENDNUM parameter of the SETPROG EXIT operator command or the ABENDNUM parameter of the PROGxx EXIT statement. By default, the system disables the exit routine if the exit routine ends with an abend on two successive calls. An abend is counted when both of the following conditions exist:

- The exit routine does not provide recovery, or the exit routine does provide recovery but percolates the error.
- The system allows a retry (the recovery routine is entered with bit SDWACLUP off).

Return codes for multiple exit routines: If you have associated multiple exit routines with the OCE_LABELANOMALY exit, and those exit routines return different valid return codes (as listed in the following section), then the final return code is determined as follows

1. If an exit routine completes with return code 16, an ABEND will be issued using the abend code in TEPMABCODE and the reason code in TEPMRSNCODE. The exit routine must set those fields. If multiple exit routines give return code 16, only the first ABEND will be issued. The system terminates the task without calling the DCB abend exit.
2. If any exit routine has a return code of 8 then the final return code is 8, and the volume is rejected.

3. Else if at least one exit routine has a return code of 4, then the final return code is 4, and processing continues.
4. Else if any exit routine has a return code of 12, no other exit routines are called, and an ABEND is issued.

If any exit routine returns a non-valid return code, an ABEND is issued. If one exists, the DCB abend exit will be called.

Label Anomaly Exit (IFG019LA) Return Codes

Before OPEN or EOV goes to the label editor routine it calls the label anomaly exit. The exit's return code causes one of these actions:

Return Code
Description

4

The exit might have turned off any of the anomaly bits for the conditions that it has taken care of. If they now all are off for an input tape, then the system will accept the tape. If the system cannot overcome the anomaly in an output tape, then the system will reject the tape. This means to reject the tape if it is a nonspecific request or to issue ABEND if it is a specific volume request.

If the exit returns with all the anomaly bits off and it is an output tape, then the system changes the tape to try to override the problem.

If the problem is that volume label information is needed to write the label, then the exit must supply the complete label in EBCDIC in the area that TEPMLABEL points to and set bit TEPMNEWLAB in the main parameter list. The caller will do any necessary conversion to ASCII. If the exit sets TEMPMNEWLAB on but it is not a valid volume label, then the system will write a message and not call the exit until IPL.

If the exit supplies the label or authorizes destruction of labels, the exit takes responsibility for security and expiration date processing.

If any anomaly bit remains on, then the system will call the label editor (OMODVOL1 or EMODVOL1) if the condition is one that the label editor is designed to handle. If the condition is one that the label editor is not designed to handle, then the result will be as stated in the description.

8

Reject the volume and do not call the label editor. This means to reject the tape if it is a nonspecific request or the conflict is "volume serial conflict" or to issue ABEND if it is a specific volume request.

If the job requests a specific volume and bit TEPAABEND is also set, the system abends the job.

C

The exit requests to abend the job for anomalies indicated by TEPAOUTSEQ, TEPARDFMLV, TEPARDBMLV, or TEPARDFMFV.

Note that return code 0 for this exit is not supported.

If neither the exit nor the label editor rejects the volume, then the system can call the exit more than once for the same volume, but for a different reason.

Volume Mount Exit

The volume mount exit has three functions. During a particular OPEN or EOV, this exit can be called for any or all of the following reasons:

- Verifying the volume (Verify that the mounted volume is acceptable.)
- Writing the volume label
- Processing volume security.

Each invocation of the exit is for only one of these reasons.

- Volume verification.

  During the course of opening and processing a tape data set, OPEN and EOV calls the volume mount exit when OPEN or EOV is verifying that the correct volume was mounted. The OPEN option can be any valid option and the label type can be any (SL, SUL, AL, AUL, NL, LTM, BLP, NSL). The system has just mounted the tape. When the system is planning to write file one header labels, this exit is called once for label verification and once prior to rewriting the volume label. UCBTFL1 indicates the type of volume mounted.

  For volume verification the exit can change the user's request to be for a different volume or to be a scratch request. Independent of this the exit can cause the mounted SL or AL volume to appear to have a different volume label of the same type or to be unlabeled. See the return code 0 description below.

  When the exit is called, open or EOV is processing a tape volume mount that has been satisfied by an operator, an automated cartridge loader or a tape library dataserver. The system has not previously verified the volume.

  The label anomaly exit or OMODVOL1 or EMODVOL1 handles any input or output error or other problems when reading the label. The error might have been that the tape is 36-track and was read on an 18-track drive. In this case, the sense bytes or label anomaly exit provided the volume serial number, since the real label is not available.

  This call is before RACF is called (for RACHECK or RACDEF).

The volume mount exit uses the block ID that is supplied to it by the tape management system. When DFSMSrmm issues an OPEN, the exit writes the block ID that is associated with the data set into JFCB field, JFCRBIDO, and turns on JFCB bit, JFCPOSID. When DFSMSrmm issues a CLOSE, the exit writes the block ID that is associated with the data set into JFCB field, JFCRBIDC. The tape management system can be DFSMSrmm or a functionally equivalent product. The block ID is the one associated with the data set's HDR1 or EOF2 label. For information about how information is written into the JFCB see *z/OS DFSMS Using Magnetic Tapes*.

The following table shows results from the volume mount exit and OPEN or CLOSE processing. The results depend on the disposition that is specified by the program.

OPEN or CLOSE Processing Method	**block ID**	**Volume Mount Exit**	**OPEN or CLOSE Processing**
OPEN OUTPUT NEW	Associated with data set's HDR1 label	– Writes block ID into JFCB field, JFCRBIDO – Turns on JFCB bit, JFCPOSID	CLOSE fast positions to block identified by block ID
OPEN OUTPUT MOD	Associated with data set's EOF2 label	– Writes block ID into JFCB field, JFCPOSID – Turns on JFCB bit, JFCPOSID	– OPEN fast positions to block identified by block ID – OPEN positions tape in front of tape mark that preceeds the data set's EOF1 label
OPEN OUTPUT OLD	Associated with data set's HDR1 label	– Writes block ID into JFCB field, JFCRBIDO – Turns on JFCB bit, JFCPOSID	OPEN fast positions to data set's HDR1 label

OPEN or CLOSE Processing Method	block ID	Volume Mount Exit	OPEN or CLOSE Processing
OPEN INPUT	Associated with data set's HDR1 label	– Writes block ID into JFCB field, JFCRBIDO – Turns on JFCB bit, JFCPOSID	OPEN fast positions to data set's HDR1 label
OPEN RDBACK	Associated with data set's EOF2 label	– Writes block ID into JFCB field, JFCRBIDO – Turns on JFCB bit, JFCPOSID	OPEN fast positions tape in front of tape mark that preceeds data set's EOF1 label
CLOSE OUTPUT	Associated with data set's EOF2 label	Writes block ID into JFCB field, JFCRBIDC	CLOSE passes control to DFSMSrmm File End on Volume Exit

- Volume write function.

  The system is about to rewrite the volume label because the user is open for output to the first file. When the system is writing file one header labels, this exit will be called once prior to label verification and once prior to rewriting the volume label. UCBTFL1 indicates the type of volume mounted.

  If the request was for a scratch tape, the system has determined that the mounted tape is acceptable. If the request was for a specific tape, the system has determined that the mounted tape is the correct one.

- Volume security function.

  The system has just issued the RACROUTE macro to check the user's authority to the volume. The caller of this exit has not tested the SAF return code which is in the exit's parameter list. The exit can change the results of the security processing. You can return details for the first tape data set on the volume. Subject to the DEVSUPxx TAPEAUTHF1 installation option, the tape data set details can be used through RACROUTE as an additional authorization check.

  Note: Certain authorized programs can bypass issuing the RACROUTE macro which also bypasses calling this security function.

The volume mount exit will receive a UCB address from TEPMUCB in IFGTEP, from DEBUCBA and from TIOEFSRT. All three sources will allow a 31–bit UCB address. TEPMICB will sometimes be an uncaptured 31–bit address. If the DEB31UCB bit is on, the UCB address and modeset byte will be different as described in IEZDEB. If they use the DCBTIOT field to find the TIOT entry address, they should be changed to use TEPMDSAB to point to the DSAB, which points to the TIOT entry or XTIOT.

This exit does not replace any checking done by open or EOV but it can replace the SAF return code before open or EOV tests it.

Volume Mount Exit Function-Specific Parameter List

The following figure is the format of the function-specific parameter list for the Volume Mount Exit. See Table 17 on page 76 for a description of the main parameter list.

Table 19. Volume Mount Exit Parameter List

Offset	Length or Bit Pattern	Name	Description
00 (X'00')		TEPO	Open, close, EOV tape exits volume mount parameter list
00 (X'00')	8	TEPOID	Control block identifier
08 (X'08')	4	TEPOLEN	Length of parameter list
12 (X'0C')	1	TEPOVER	Version of parameter list (1)

Table 19. Volume Mount Exit Parameter List (continued)

Offset	Length or Bit Pattern	Name	Description
13 (X'0D')	3		Reserved
16 (X'10')	1	TEPOFLAG1	Flags
	00..		Volume verification. Can be any open option.
	10..	TEPORDWRIT	Volume write function. Open option is not INPUT or RDBACK and the current application operation is a write.
	01..	TEPOSAF	Volume security function. After call to SAF.
	001.	TEPOLBLED	Label anomaly and possibly tape label editor (OMODVOL1, EMODVOL1) has been called for this volume. Volume verification only. Can be any open option.
	0001	TEPOIOE	I/O error during load point read for volume verification.
	11..		Does not occur in current release
	.... 1...	TEPPSEUDO	Pseudo-volume label during the label read process
17 (X'11')	1	TEPOFLAG2	Flags
	1...	TEPORACF	Data set being created is protected by a RACF discrete profile.
	.1..	TEPOALFRC	ON - Exit can override the AL version level that will be used to write the label (only on if file 1, volume 1 and output). OFF - Version level cannot be overridden.
	..1.	TEPOEXFRC	ON - Exit has provided an AL version level to use (only valid if TEPOALFRC is on). OFF - exit has not provided an AL version level to use. Tape will be written in the version selected by OPEN.
	...1	TEPOALVER	ON - ISO/ANSI Version 4. OFF - ISO/ANSI Version 3.
	.... 1...	TEPOIGNORE	Expiration date is ignored
	.... .1..	TEPOHONOR	Expiration date is honored
	.... ..1.	TEPOINDEX	No cartridge loader indexing on the next scratch mount request in the OPEN or EOV routines
	1	TEPOABEND	Job abend when the return code is 8
18 (X'12')	2	TEPOSAFRC	SAF/RACF return code
20 (X'14')	2	TEPOSAFRS	SAF/RACF reason code
22 (X'16')	44	TEPODSNF1	File 1 dsname to be returned by Volume Mount Exit and Volume Security Function. Used by O/C/EOV to authorize access to the first file on a volume when TAPEAUTHF1=YES is set.
66 (X'42')	2	TEPODS1FS	File 1 data set sequence number; corresponds to the JCL LABEL keyword data set sequence number.

Controlling the volume mount exit routine through the dynamic exits facility

IBM has defined the Volume Mount installation exit to the dynamic exits facility. You can refer to the dynamic exit by the name OCE_VOLUMEMOUNT. You can use the EXIT statement of the PROGxx parmlib member, the SETPROG EXIT operator command, or the CSVDYNEX macro to control this exit and the exit routines of the exit.

The system attempts to add exit routine OCE_VOLUMEMOUNT, unless an exit routine has been associated with OCE_VOLUMEMOUNT by PROGxx or SETPROG.

If you have associated exit routines with OCE_VOLUMEMOUNT the system does not use the default exit routine. In this case, if you require an exit routine of the default name, you must explicitly add the default exit routine to PROGxx.

To limit the number of times the exit routine ends with an abend, before the exit routine becomes inactive, you can use the ADDABENDNUM and ABENDCONSEC parameters on the CSVDYNEX REQUEST=ADD macro, or the ABENDNUM parameter of the SETPROG EXIT operator command or the ABENDNUM parameter of the PROGxx EXIT statement. By default, the system disables the exit routine if the exit routine ends with an abend on two successive calls. An abend is counted when both of the following conditions exist:

- The exit routine does not provide recovery, or the exit routine does provide recovery but percolates the error.
- The system allows a retry (the recovery routine is entered with bit SDWACLUP off).

Return codes for multiple exit routines: If you have associated multiple exit routines with the OCE_VOLUMEMOUT exit, and those exit routines return different valid return codes (as listed in the following section), then the final return code is determined as follows:

1. If an exit routine completes with return code 16, an ABEND will be issued using the abend code in TEPMABCODE and the reason code in TEPMRSNCODE. The exit routine must set those fields. If multiple exit routines give return code 16, only the first ABEND will be issued. The system terminates the task without calling the DCB abend exit.
2. If at least one exit routine has a return code of 8, then the final return code is 8, and the volume is rejected.
3. Else if any exit routine has a return code of 0 then the final return code is 0, and volume is accepted.
4. Otherwise, the final return code is 4, and the system continues processing.

If any exit routine returns a non-valid return code, an ABEND is issued. If one exists, the DCB abend exit will be called.

Volume Mount Exit Return Codes

Return Code
Description

0

Accept the volume except as otherwise described here. The following points apply if the exit is called for volume verification and gives return code 0:

- The exit can change the volume serial number in TEPMVOL in the main parameter list to a new volume serial. The system will update the JFCB or JFCB extension used during OPEN or EOV; the system also will update the SWA copy of the JFCB or JFCB extension unless the application program prevented it by turning on bit JFCNWRIT. This bit is honored during OPEN, not during EOV. The system will update the catalog if the catalog retrieved it or the user specified CATLG for the DD DISP=parameter.

  The system will enqueue on the new volume serial. If the enqueue is successful, then the system will dequeue the original volume serial. If the enqueue fails, even if it is being owned by the same job, the system will reject the tape. The system then will reissue the original mount request. It can be a specific request or a nonspecific request.
- The exit can change the volume serial number in TEPMVOL in the main parameter list to be blanks. This converts it to a nonspecific request. This results in the system taking the same actions it would have taken if the exit had not been called except that if the request had been for a specific volume, the system will dequeue that volume serial.
- The exit can change the contents of the label that TEPMLABEL points to. We don't advise changing a reserved field in a label because IBM or a standards organization might define that field. Except for

the checkpoint/restart and DDR functions, the system will treat the volume as having the changed volume label. (For those two functions the system currently reads the original labels.) If the volume serial in this changed label for this exit does not match TEPMVOL for a specific volume request, then the system will reject the volume.

- The exit can change the version of an AL volume to Version 3 or Version 4 if the system sets on bit TEPOALFRC in byte TEPOFLAG2. The system sets this bit on only if all of the following are true:

  1. it is during OPEN
  2. the OPEN option is OUTPUT or OUTIN
  3. the DD does not have DISP=MOD
  4. the file sequence number is 1
  5. the DD statement has AL coded on the LABEL parameter

  To change the version, the exit sets on TEPOEXFRC in TEPOFLAG2 and changes the eightieth byte in the area that TEPMLABEL points to be the desired version level(3 or 4) .
- The exit can change the label type of the mounted volume to unlabeled by changing the first four bytes of the area to which TEPMLABEL points. If the exit changes the area from standard labeled or non-standard labeled to unlabeled, then the system will not process existing labels. This prevents testing the expiration date, testing for data set password protection, testing the ISO/ANSI access code (ACCODE) and testing whether the volume is a checkpoint volume.

  Do not change this area to indicate the tape has SL or AL labels unless the tape actually does have that type of label.
- The exit can change TEPMVSEQ in the main parameter list. When a valid volume sequence is returned (for example, not exceeding the specific volume count in the JFCB), the system demounts the current volume and issues a mount request for the volume serial associated with the exit-supplied volume sequence.
- The exit can supply a volume serial number when processing non-SMS or SMS MTL (Manual Tape Library), nonspecific scratch NL/BLP output. The exit can also supply a volume serial number when processing SMS MTL nonspecific scratch SL/SL output when an NL volume is mounted. This will result in the system using the exit supplied volume serial number instead of generating an LNUMBER, or instead of issuing IEC519I during SMS MTL processing for a nonspecific scratch SL/AL volume when an NL volume is mounted. Prior to returning with RC00, the exit must supply a volume serial number in TEPMVOL in the main parameter list and set bit TEPMNEWLAB in byte TEPMFLAG6. Just as when an LNUMBER is generated, the system will continue to process as a nonspecific volume request.

  Note: For SMS MTL, when the VOLM (Volume Mount Exit) Tape Installation Exit does not supply a volume serial, IEC519A is issued requesting the operator to reply with the external volume serial.
- If the exit was called for volume security, then TEPOSAFRC is to replace the SAF return code. The exit might have changed it to a valid SAF return code.
- When TAPEAUTHF1=YES is set, TEPODSNF1 and TEPODS1FS are used by O/C/EOV to check authority to the first file on the volume. Refer to Table 19 on page 91 for field details.

4

Continue usual processing; the exit is not modifying anything.

8

Reject the volume. The exit should write a message with route code 3 and 11 to explain the reason for the rejection. This return code means to reject the tape whether it is a nonspecific volume request or a specific volume request. This return code is not valid when writing the volume label.

If the call was for the volume security function and the bit TEPOABEND is also set, the system abends the job.

File Validation Exit

If OPEN or EOV mounted a tape, then it has done all processing of an existing volume label before calling the file validation exit. If the call is for OPEN, the forward DCB merge has not been done and the DCB OPEN exit has not been called. If it is an SL or AL tape, OPEN or EOV has read the first 80 bytes of the header or trailer label (if reading backward) but has not examined it yet except to translate it from ASCII if the first four characters are ASCII, HDR1, EOV1, or EOF1. If open for output, then no writing has been done yet on the volume. If it is not SL or AL, then OPEN has not yet positioned the tape. This is so the exit can reject the tape if the user is trying to read past the last file or writing more than one file past the last file. This exit can verify that the specified 44-character data set name matches the tape management system's database.

This call is before RACDEF is called and after RACHECK.

This exit does not replace any checking done by OPEN or EOV except when the exit specifies that the operator is not to be asked about expiration dates on the volume and/or that password security processing should be bypassed during tape SL output processing to file sequence one.

The file validation exit will receive a UCB address from TEPMUCB in IFGTEP, from DEBUCBA and from TIOEFSRT. All three sources will allow a 31–bit UCB address but TIOEFSRT will contain zero if the actual UCB is above the line and allocation did not capture the UCB address. TEPMUCB will sometimes be an uncaptured 31–bit address. If the DEB31UCB bit is on, the UCB address and modeset byte will be different as described in IEZDEB. It is best not to use the DCBTIOT field to find the TIOT entry address. This is because DCBTIOT will contain zeroes if the user program caused an XTIOT to be created instead of a TIOT entry. You always can use TEPMDSAB to point to the DSAB, which points to the TIOT entry or XTIOT.

File Validation Exit Function-Specific Parameter List

The following figure is the format of the function-specific parameter list for the File Validation Exit. See Table 17 on page 76 for a description of the main parameter list.

Table 20. File Validation Exit Parameter List

Offset	Length or Bit Pattern	Name	Description
00 (X'00')		TEPV	Open, close, EOV tape exits file validation parameter list
00 (X'00')	8	TEPVID	Control block identifier
08 (X'08')	4	TEPVLEN	Length of parameter list
12 (X'0C')	1	TEPVVER	Version of parameter list (1)
13 (X'0D')	3		Reserved

Table 20. File Validation Exit Parameter List (continued)

Offset	Length or Bit Pattern	Name	Description
16 (X'10')	1	TEPVFLAG1	Flags
	1...	TEPVIGNORE	Set to 0 by caller. If exit sets it on, then OPEN and EOV will ignore the expiration date for the file on the volume. Takes precedence over the next bit.
	.1..	TEPVHONOR	Set to 0 by caller. If exit sets it on, then OPEN and EOV will honor the expiration dates for the file on the volume and not ask the operator if a date has not passed. This means that OPEN or EOV rejects the volume or issue ABEND.
	..1.	TEPVSCRTCH	Original volume requirement was nonspecific
	...1	TEPVBYSEC	Set to 0 by the caller. If all the following are true, then the caller will ignore password protection of the existing file: • OPEN option allows writing. If the caller is EOV, the last user operation was a write. • IBM standard label tape mounted • Either the caller is open and the file sequence number is one, or the caller is EOV. • The exit turned this bit on. This bit is intended for use by an exit that can determine that password protection no longer applies. It does not affect SAF processing.
	.... .1..	TEPVNINDEX	No loader indexing on the next scratch mount request in the OPEN or EOV routines
	.... ..1.	TEPVABEND	Job abends when return code is 8
17 (X'11')	3		Reserved, set to zero

Controlling the file validate exit routine through the dynamic exits facility

IBM has defined the File Validate installation exit to the dynamic exits facility. You can refer to the dynamic exit by the name OCE_FILEVALIDATE. You can use the EXIT statement of the PROGxx parmlib member, the SETPROG EXIT operator command, or the CSVDYNEX macro to control this exit and the exit routines of the exit.

The system attempts to add exit routine OCE_FILEVALIDATE, unless an exit routine has been associated with OCE_FILEVALIDATE by PROGxx or SETPROG.

If you have associated exit routines with OCE_FILEVALIDATE the system does not use the default exit routine. In this case, if you require an exit routine of the default name, you must explicitly add the default exit routine to PROGxx.

To limit the number of times the exit routine ends with an abend, before the exit routine becomes inactive, you can use the ADDABENDNUM and ABENDCONSEC parameters on the CSVDYNEX REQUEST=ADD macro, or the ABENDNUM parameter of the SETPROG EXIT operator command or the ABENDNUM parameter of the PROGxx EXIT statement. By default, the system disables the exit routine if the exit routine ends with an abend on two successive calls. An abend is counted when both of the following conditions exist:

- The exit routine does not provide recovery, or the exit routine does provide recovery but percolates the error.
- The system allows a retry (the recovery routine is entered with bit SDWACLUP off).

Return codes for multiple exit routines: If you have associated multiple exit routines with the OCE_FILEVALIDATE exit, and those exit routines return different valid return codes (as listed in the following section), then the final return code is determined as follows:

1. If an exit routine completes with return code 16, an ABEND will be issued using the abend code in TEPMABCODE and the reason code in TEPMRSNCODE. The exit routine must set those fields. If multiple exit routines give return code 16, only the first ABEND will be issued. The system terminates the task without calling the DCB abend exit.
2. If at least one exit routine has a return code of 8, then the final return code is 8, and the volume is rejected.
3. Else if any exit routine has a return code of 0 then the final return code is 0, and volume is accepted.
4. Otherwise, the final return code is 4, and the system continues processing.

If any exit routine returns a non-valid return code, an ABEND is issued. If one exists, the DCB abend exit will be called.

File Validation Exit Return Codes

Return Code
Description

0

Accept the volume unless the system finds a problem. The exit can set indicators in the parameter list (see Table 20 on page 95) that mean that the system is not to ask the operator about expiration dates.

4

The exit takes no responsibility and the system continues usual processing.

8

Volume is not acceptable for this request. This should be used only when the system would otherwise accept the volume. For example Open, close, EOV checks security. This condition can also mean that the volume is not acceptable on this system. The system rejects the volume with a standard message that does not indicate the reason. This means to reject the tape if it is a nonspecific request or to issue ABEND if it is a specific volume request. The exit should write an appropriate message.

If bit TEPVABEND is also set, the system abends the job regardless of whether the volume request is specific or nonspecific.

File Start on Volume Exit

The system has read or written all file labels (as appropriate), has called all user and installation exits, issued RACDEF if needed and has positioned the tape for reading or writing user data. It is too late to reject the volume. OPEN or EOV has checked for most or all of the expected ABEND conditions.

No tape label is reliably available to this exit because user labels might have been processed. The exit is expected to gather data from other places.

The caller is OPEN or EOV. The exit is called for the beginning of the file on each volume.

The file start on volume exit will receive a UCB address from TEPMUCB in IFGTEP, from DEBUCBA and from TIOEFSRT. All three sources will allow a 31–bit UCB address. TEPMICB will sometimes be an uncaptured 31–bit address. If the DEB31UCB bit is on, the UCB address and modeset byte will be different as described in IEZDEB. If they use the DCBTIOT field to find the TIOT entry address, they should be changed to use TEPMDSAB to point to the DSAB, which points to the TIOT entry or XTIOT.

File Start on Volume Exit Function-Specific Parameter List

The following figure is the format of the function-specific parameter list for the File Start on Volume Exit. See Table 17 on page 76 for a description of the main parameter list.

Table 21. File Start on Volume Exit Parameter List

Offset	Length or Bit Pattern	Name	Description
00 (X'00')		TEPS	File-start-on-volume exit parameter list
00 (X'00')	8	TEPSID	Control block identifier
08 (X'08')	4	TEPSLEN	Length of parameter list
12 (X'0C')	1	TEPSVER	Version of parameter list (1)
13 (X'0D')	2		Reserved
15 (X'0E')	1	TEPSSGTY	SMS storage group pool type. The value 1 means “general”.
16 (X'10')	6	TEPSFIRST	Volume serial of the first volume for the data set if available
22 (X'16')	1	TEPSFLAG1	Flags
	1...	TEPSSCRTCH	Original volume was nonspecific
23 (X'17')	1	TEPSPARP	Media position (n). Represents the distance from beginning of the volume to start of the data set.
The following four fields are available only with DISP=NEW.			
24 (X'18')	4	TEPSSTORG	Address of two-byte length of storage group name followed by name
28 (X'1C')	4	TEPSMGMTC	Address of two-byte length of management class name followed by name
32 (X'20')	4	TEPSSTORC	Address of two-byte length of storage class name followed by name
36 (X'24')	4	TEPSDATAC	Address of two-byte length of data class name followed by name
40 (X'28')	6	TEPSPREVL	Volume serial of previous volume. Defined only if the caller is EOV and the current volume is from the same DD statement as the previous volume.
46 (X'2E')	2		Reserved
48 (X'30')	4	TEPSBLKID	Address of the four-byte block identifier of the HDR1 label
52(X'34')	6	TEPSKBTRV	KB traversed from BOT.
58(X'3A')	2	TEPSMPOS	Media position decimal value *n*, representing the numerator in the fraction *n*/65535. This fraction represents the ratio of the distance from the beginning of the volume to the current position rounded down to the nearest 1/65535th.
60(X'3C')	3	TEPS4KBYT	Bytes written to the volume in 4K increments since volume mounted.

Controlling the file start exit routine through the dynamic exits facility

IBM has defined the File Start installation exit to the dynamic exits facility. You can refer to the dynamic exit by the name OCE_FILESTART. You can use the EXIT statement of the PROGxx parmlib member, the

SETPROG EXIT operator command, or the CSVDYNEX macro to control this exit and the exit routines of the exit.

The system attempts to add exit routine OCE_FILESTART, unless an exit routine has been associated with OCE_FILESTART by PROGxx or SETPROG.

If you have associated exit routines with OCE_FILESTART the system does not use the default exit routine. In this case, if you require an exit routine of the default name, you must explicitly add the default exit routine to PROGxx.

To limit the number of times the exit routine ends with an abend, before the exit routine becomes inactive, you can use the ADDABENDNUM and ABENDCONSEC parameters on the CSVDYNEX REQUEST=ADD macro, or the ABENDNUM parameter of the SETPROG EXIT operator command or the ABENDNUM parameter of the PROGxx EXIT statement. By default, the system disables the exit routine if the exit routine ends with an abend on two successive calls. An abend is counted when both of the following conditions exist:

- The exit routine does not provide recovery, or the exit routine does provide recovery but percolates the error.
- The system allows a retry (the recovery routine is entered with bit SDWACLUP off).

Return codes for multiple exit routines: If you have associated multiple exit routines with the OCE_FILESTART exit, and those exit routines return different valid return codes (as listed in the following section), then the final return code is determined as follows:

1. If an exit routine completes with return code 16, an ABEND will be issued using the abend code in TEPMABCODE and the reason code in TEPMRSNCODE. The exit routine must set those fields. If multiple exit routines give return code 16, only the first ABEND will be issued. The system terminates the task without calling the DCB abend exit.
2. If at least one exit routine has a return code of 0, processing continues.

If any exit routine returns a non-valid return code, an ABEND is issued. If one exists, the DCB abend exit will be called.

File Start on Volume Exit Return Code

Return Code
Description

0
Take usual action.

File End on Volume Exit

When a user is finishing using a data set on a volume, EOV or CLOSE calls the file end on volume exit routine.

For input this exit is called from EOV unless it is end of data or the labels are NSL and the user has specified in the DCB exit list that trailer label processing is to be deferred from EOV to CLOSE. For output this exit is called from EOV except for the last volume in which case it is called from CLOSE. CLOSE TYPE=T results in this exit being called if the last user operation was a write. This causes CLOSE to write trailer labels but not close the DCB.

If the caller is EOV, then there might be an exit call later for the next volume for input or output or the next data set in the concatenation.

The exit caller has checked or written any trailer labels and disposed of the volume or is about to do so. The return code will not affect the system action. The call is to inform the tape management system which can gather statistics.

The file end on volume exit will receive a UCB address from TEPMUCB in IFGTEP, from DEBUCBA and from TIOEFSRT. All three sources will allow a 31–bit UCB address. TEPMICB will sometimes be an uncaptured 31–bit address. If the DEB31UCB bit is on, the UCB address and modeset byte will be different as

described in IEZDEB. If they use the DCBTIOT field to find the TIOT entry address, they should be changed to use TEPMDSAB to point to the DSAB, which points to the TIOT entry or XTIOT.

File End on Volume Exit Function-Specific Parameter List

The following figure is the format of the function-specific parameter list for the File End on Volume Exit. See Table 17 on page 76 for a description of the main parameter list.

Table 22. File End on Volume Exit Function-Specific Parameter List

Offset	Length or Bit Pattern	Name	Description
00 (X'00')		TEPE	File-end-on-volume exit parameter list
00 (X'00')	8	TEPEID	Control block identifier
08 (X'08')	4	TEPELEN	Length of parameter list
12 (X'0C')	1	TEPEVER	Version of parameter list (1)
13 (X'0D')	3		Reserved
16 (X' 10')	4	TEPEBLKID	Address of the 4-byte block identifier of the next HDR1 label after the trailer labels; it is initialized during the output process.
20 (X' 14')	1	TEPEPARP	Media position (n). Represents the distance from beginning of the volume to start of the data set.
21 (X' 15')	1	TEPEFLAG1	Flags
	1...	TEPELEFT	CLOSE disposition: End of file UCBFSCT/UCBFSEQ incremented
	.1..	TEPEFILES	File start on volume exit previously received control.
22 (X'16')	2		Reserved
24 (X'18')	4	TEPETBLK	Total block count across all volumes; initialized at EOV and CLOSE output
28 (X'1C')	4	TEPELSTB	Total of unsynchronized blocks that were written to the storage controller but an I/O error prevented them from being written to the medium. It includes tape marks.
32(X'20')	6	TEPEPREVL	Volume serial of previous volume. Defined only if caller is EOV and the current volume is for the same DD statement as the previous volume.
38(X'26')	2	TEPEFSCT	Data set sequence number. This is the file number relative to the beginning of the current volume.
40(X'28')	2	TEPEFSEQ	Data set sequence count. This is the file number relative to the beginning of the aggregate of volumes. It matches the value in JFCBFLSQ in the JFCB. Field TEPMJFCB in the main parameter list points to the JFCB.
42(X'2A')	64	TEPEKEK1	KEK label 1.
106(X'6A')	64	TEPEKEK2	KEK label 2.
170(X'AA')	1	TEPEKCD1	Key encoding mechanism associated with KEK label 1: L - Label, H - Hash

Table 22. File End on Volume Exit Function-Specific Parameter List (continued)

Offset	Length or Bit Pattern	Name	Description
171(X'AB')	1	TEPEKCD2	Key encoding mechanism associated with KEK label 2: L - Label, H - Hash
172(X'AC')	6	TEPEKBTRV	KB traversed from BOT.
180(X'B4')	3	TEPE4KBYT	Bytes written to the volume in 4K increments since volume mounted.
178(X'B2')	2	TEPEMPOS	Media position decimal value *n*, representing the numerator in the fraction *n*/65535. This fraction represents the ratio of the distance from the beginning of the volume to the current position rounded down to the nearest 1/65535th.

Controlling the file end exit routine through the dynamic exits facility

IBM has defined the File End installation exit to the dynamic exits facility. You can refer to the dynamic exit by the name OCE_FILEEND. You can use the EXIT statement of the PROGxx parmlib member, the SETPROG EXIT operator command, or the CSVDYNEX macro to control this exit and the exit routines of the exit.

The system attempts to add exit routine OCE_FILEEND, unless an exit routine has been associated with OCE_FILEEND by PROGxx or SETPROG.

If you have associated exit routines with OCE_FILEEND the system does not use the default exit routine. In this case, if you require an exit routine of the default name, you must explicitly add the default exit routine to PROGxx.

To limit the number of times the exit routine ends with an abend, before the exit routine becomes inactive, you can use the ADDABENDNUM and ABENDCONSEC parameters on the CSVDYNEX REQUEST=ADD macro, or the ABENDNUM parameter of the SETPROG EXIT operator command or the ABENDNUM parameter of the PROGxx EXIT statement. By default, the system disables the exit routine if the exit routine ends with an abend on two successive calls. An abend is counted when both of the following conditions exist:

- The exit routine does not provide recovery, or the exit routine does provide recovery but percolates the error.
- The system allows a retry (the recovery routine is entered with bit SDWACLUP off).

Return codes for multiple exit routines: If you have associated multiple exit routines with the OCE_FILEEND exit, and those exit routines return different valid return codes (as listed in the following section), then the final return code is determined as follows:

1. If an exit routine completes with return code 16, an ABEND will be issued using the abend code in TEPMABCODE and the reason code in TEPMRSNCODE. The exit routine must set those fields. If multiple exit routines give return code 16, only the first ABEND will be issued. The system terminates the task without calling the DCB abend exit.
2. If at least one exit routine has a return code of 0, processing continues.

If any exit routine returns a non-valid return code, an ABEND is issued. If one exists, the DCB abend exit will be called.

File End on Volume Return Code

Return Code
Description

0
Take usual action.

Nonstandard Labels

Nonstandard labels do not conform to IBM or ISO/ANSI standard label formats. You can design them and provide routines to write and process them. There are no requirements regarding their length, format, contents, and number, except that the first record on a BCD, EBCDIC, or ASCII tape cannot be a standard volume label.

Figure 29 on page 102 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. Your routines write all labels and tape marks. If an operating system access method is used to retrieve the data, tape marks should precede and follow the data set to indicate the end-of-data-set condition for forward and backward read operations.

Nonstandard labeled tapes are not supported in a library (either by the Tape Library Dataserver or by a manual library).

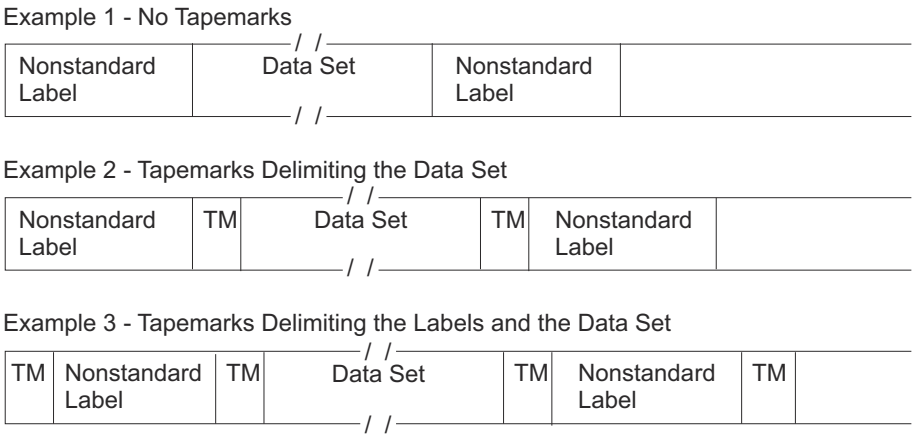


Figure 29. Examples of Tape Organizations with Nonstandard Labels

- No Tape marks: This type of organization can be created by your nonstandard label processing routines, and read with the EXCP technique or BSAM. Do not use with QSAM because there is no tape mark to signal end-of-data.
- Tape marks Delimiting the Data Set: This is the recommended organization. The tape marks are written by your nonstandard label processing routine. When the tape is read by an operating system access method, the tape mark following the data set signals end-of-data for forward read operations, and the tape mark preceding the data set signals end-of-data for backward read operations.
- Tape marks Delimiting the Labels and the Data Set: This is an expansion of the preceding organization. The operating system does not handle the additional tape marks that precede and follow the labels. Instead, your nonstandard label processing routine writes and uses them.

Processing Nonstandard Labels

Generally, the 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.
- Are entered in the addressing mode that is determined by the AMODE option when they are link-edited.
- 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.

- 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 in Table 23 on page 103 for each SKC SKC control program routine and for each type of label processing routine.

Table 23. Control Program and Label Processing Routine Modules

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

Use the GETMAIN or STORAGE macro instruction to obtain virtual storage for all your work areas, including areas used to read in or create a label. Use the FREEMAIN or STORAGE macro instruction to release this virtual storage.

Support for RACF protection of tape volumes might be a part of nonstandard label processing routines.

Input Header Label Routines (NSLOHDRI, NSLEHDRI)

When nonstandard labels are specified, the control program checks the input tape to ensure 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 ASCII and is 80 bytes long or more, a message from the control program rejects the tape and directs the operator to remove the current volume and mount the correct volume. The various error conditions that can occur during verification of the first record are explained under "Label Anomaly Exit (IFG019LA)" on page 83.

If 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.) This is not the same as unlike attributes concatenation that is described in *z/OS DFSMS Using Data Sets*. 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 and is positioned at the inter-record 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 inter-record 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.

The modeset byte is moved from a 24–bit UCB address to a 31–bit UCB address if you request the XTIOT UCB NOCAPTURE option and the NON-VSAM_XTIOT option is set to yes. OPEN, EOV, and CLOSE will

always turn DEB31UCB on in the work area to signify that DXDEBUCB is a 31–bit UCB address. Although the UCB address field will be four bytes, it typically will contain a three-byte address.

Input Trailer Label Routines (NSLETRLI)

When a tapemark is encountered on an input tape, data management's EOV routine gives control to your routine for processing input trailer labels, except when:

- The force end-of-volume (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 is positioned immediately after the tapemark at the end of the data set.
- For backward read operations, the tape is positioned immediately before the tapemark at the beginning of the data set.

Your routine does not need to 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 the user's 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 *z/OS DFSMS Using Data Sets*.

The modeset byte is moved from a 24–bit UCB address to a 31–bit UCB address if you request the XTIOT UCB NOCAPTURE option and the NON-VSAM_XTIOT option is set to yes. OPEN, EOV, and CLOSE will always turn DEB31UCB on in the work area to signify that DXDEBUCB is a 31–bit UCB address. Although the UCB address field will be four bytes, it typically will contain a three-byte address.

Output Header Label Routines (NSLOHDRO, NSLEHDRO)

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 long 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 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 “Label Anomaly Exit (IFG019LA)” on page 83.

If 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 inter-record gap preceding the nonstandard label (the tape has been rewound). However, the tape is not 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 made for a nonspecific volume. The control program recognizes a nonspecific request if the volume serial number requested in the JFCB is blank or SCRTCH. Then, UCBVOLI is set to Lxxxxx, where xxxxx is a 5-digit decimal number. The control program generates this volume serial number. It can 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.

If NSLOHDRO or NSLEHDRO returns control to the control program for a nonspecific request and the UCBVOLI field of the UCB has been modified the control ENQ's on the volume serial number to affect volume integrity and places the volume serial number in the JFCB or JFCB extension. If some other technique is used to support nonspecific request 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, after the control program's ENQ,the resourse is unavailable (either the current task previously obtained the resource or some other task holds the resource), the volume is rejected.

Your routine does not need to reposition the tape before returning control to the control program.

When tapes are first received, 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.

OPEN, EOV, and CLOSE will always turn the DEB31UCB on in the work area to signify that DXDEBUCB is a 31–bit UCB address. Although the UCB address field will be four bytes, it typically will contain a three-byte address. The modeset byte is moved from the DEBSDVM field to the DEBSDVMX field to enable 31–bit UCB addresses to be accommodated.

Output Trailer Label Routines (NSLETRLO, NSLCTRLO)

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 inter-record gap following the last written data set record.

Your routine does not need to reposition 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.

OPEN, EOV, and CLOSE will always turn the DEB31UCB on in the work area to signify that DXDEBUCB is a 31–bit UCB address. Although the UCB address field will be four bytes, it typically will contain a three-byte address. The modeset byte is moved from the DEBSDVM field to the DEBSDVMX field to enable 31 bit UCB addresses to be accommodated.

Data Recovery

Recovery routines are given control when an error occurs during open, and close and end-of-volume processing. They provide data recoverability in case of an error that results in abnormal termination of your task. Data recoverability works in conjunction with your output trailer label routines by writing a tapemark after the last data written to the tape. This indicates the end of the output data set, so that you can save the records written before the error occurred. The tapemark is only 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 is written.

Installing Nonstandard Label Routines

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 when the system is built. (You can also insert the routines after system build by link-editing them into LPALIB). See “Adding a New Exit” on page 3.

Member names for the first load module of each type of label processing routine are listed in Table 24 on page 106. Member names for additional load modules must begin with the letters NSL or IGC.

Table 24. First Load Modules

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

Writing Nonstandard Label Processing Routines

To use nonstandard labels for tape volumes you must:

- Create nonstandard label processing routines for input header trailer labels, input trailer labels, output header labels, and output trailer labels.
- Insert your routines into the link pack area (LPA) library (SYS1.LPALIB).
- Write NSL in the LABEL parameter of the DD statement at execution time.

Processing Tapes with Nonstandard Labels

Your appropriate nonstandard label processing routine is called when a data set is opened or closed or when an EOV or end-of-data-set condition occurs. When your routine is finished, it must return to data management's open, close, EOV, or restart routine, which then continues its normal processing. For input, the EOV routine handles both EOV and end-of-data-set conditions. For output, the EOV routine handles the EOV 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.

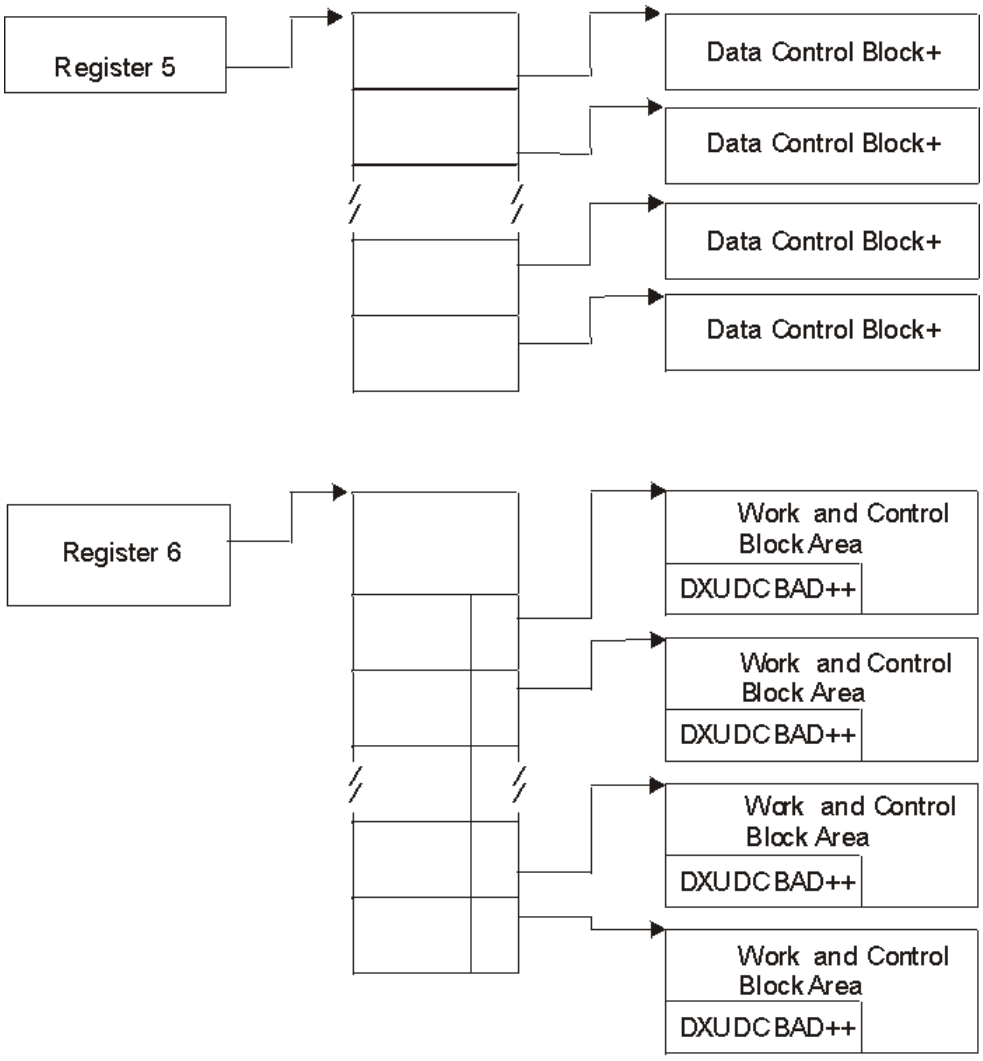
If you want the control program to maintain a block count, your header label routines that receive control from open or EOV must properly initialize the block count field of the DCB. Set DCBBLKCT to zero unless the data set is being extended when the OPEN option is EXTEND or OUTINX or when the OPEN option is OUTPUT or OUTIN and DISP=MOD is coded on the DD statement. If the data set is being extended, then set DCBBLKCT to the number of blocks currently in the file on the volume. On output, your trailer label routine can save the DCBBLKCT value in the label. On input, your trailer label routine can compare the DCBBLKCT with the block count in the label. To compare when using BSAM or QSAM without chained scheduling, your routines must adjust DCBBLKCT. If the access method is BSAM or QSAM and bit DCBCNCHS is off, then DCBIOBA points to an area containing a signed halfword to subtract from DCBBLKCT. The halfword is at offset +36 in the area.

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. To process a data set more than once in a job, or to handle multiple data set volumes, your routines must control the positioning. If your 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 (for example, changing from NSL to NL with a verified tape).

Program Functions

In processing nonstandard labels, the routines must perform many of the functions that the control program performs in processing standard labels. Use the EXCP (execute channel program) macro instruction to perform all input/output operations as reading labels, writing labels, and positioning volumes. Normally, to use EXCP you are required to 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.

- Figure 30 on page 108 shows the status of control information and pointers when your routine receives control from the open or close routine.
- When your routine receives control from the EOV routine, register 2 contains the address of a DCB, and register 4 contains the the address of a combined work and control block area. The format of this area is shown in Figure 31 on page 109.
- 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.

Figure 30. Status of Control Information and Pointers

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 copies in protection key 5 storage. The list might also include one or more ACB addresses. The list, the DCBs, and any ACBs reside below the 16 MB line.

For each DCB specified in the OPEN or CLOSE macro instruction, a combined work and control block area is built unless OPEN or CLOSE has determined that the DCB or ACB cannot be processed. 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 on the table. The format of the combined work and control block area is shown in Figure 31 on page 109.

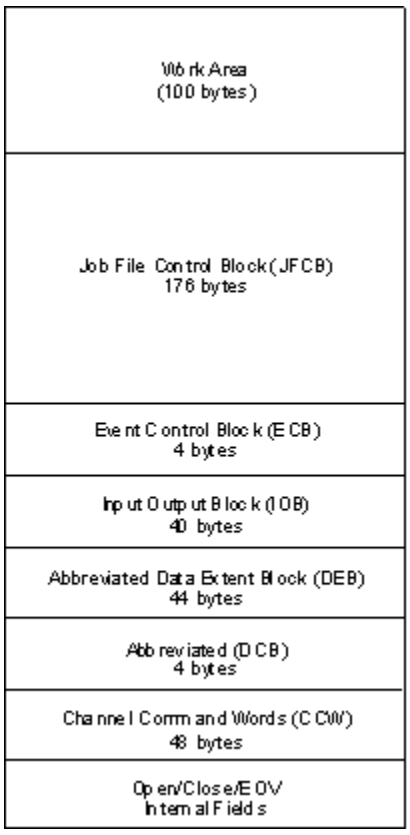


Figure 31. Format of Combined Work and Control Block Area

Your nonstandard label processing routines can address each of the fields within the work and control block area. The IECDSECT macro instruction defines the symbolic names of all these fields. Write 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.

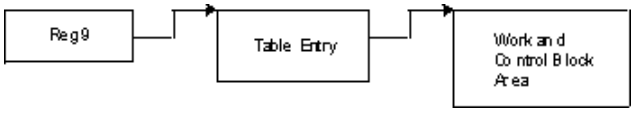


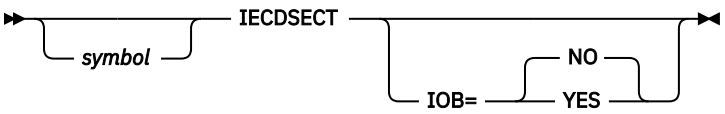
Figure 32. Status of Control Information and Pointers from the Control Program's Restart Routine

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.

Mapping the Common Open, Close, EOV Work Area

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 31 on page 109.

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:



NO specifies that you do not want symbols for a 32-byte basic IOB generated in the work area. We recommend no.

YES specifies that you want symbols for a 32-byte basic IOB generated in the work area.

Flowcharts for Sample Routines

The following information illustrates flowcharts and logic of nonstandard label processing 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 each of the flowcharts is numbered; each number corresponds to an item in the lists of explanations that follow the figures.

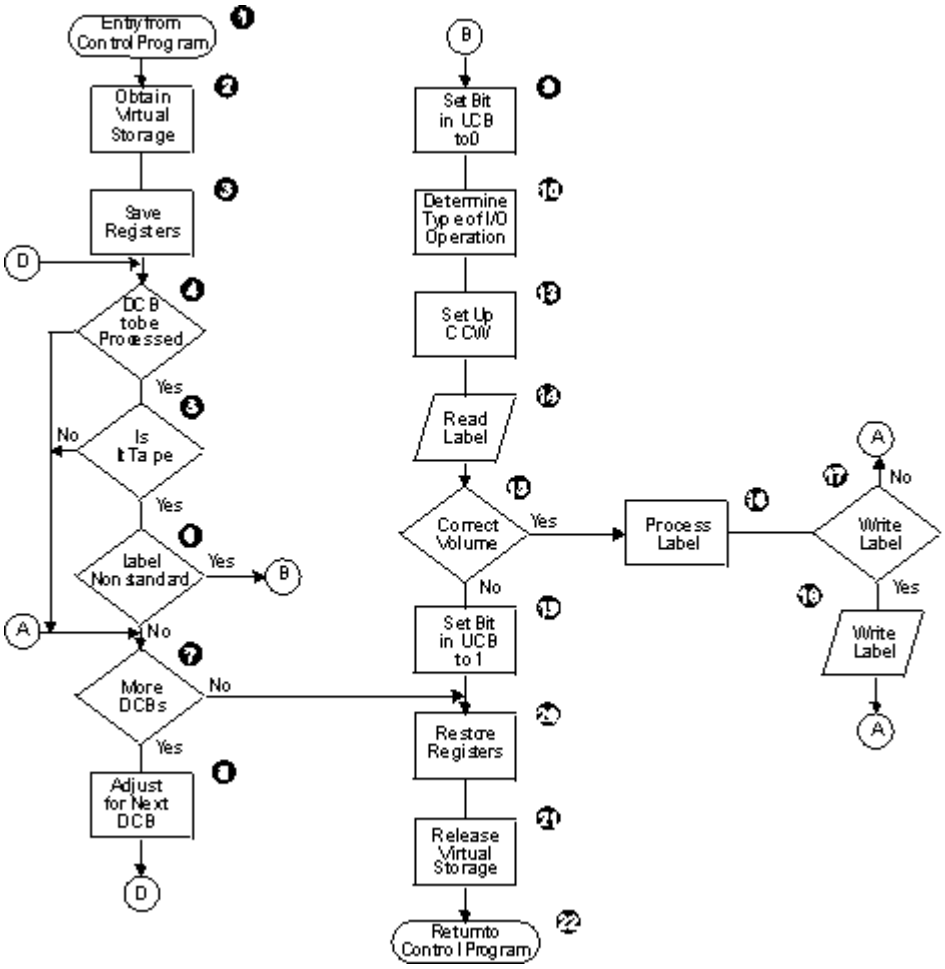


Figure 33. General Flow of a Nonstandard Label Processing Routine After Receiving Control from the Open Routine.

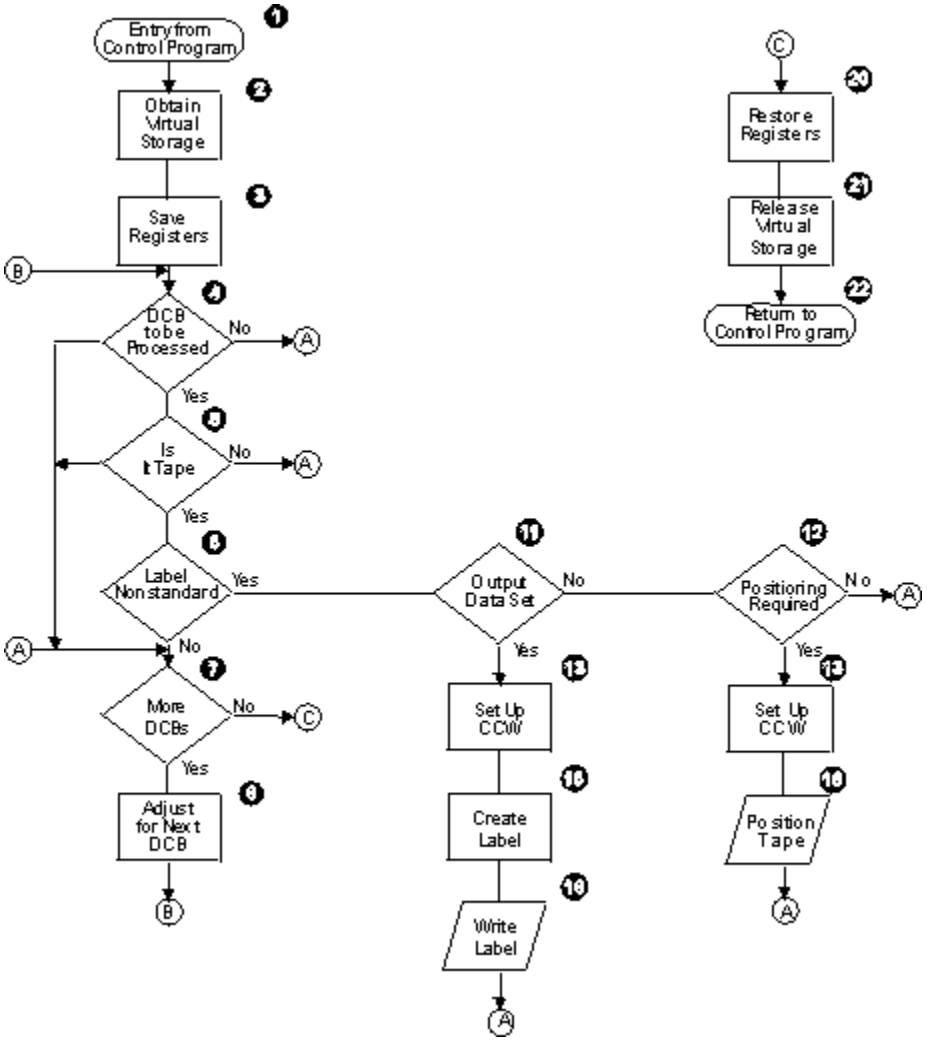


Figure 34. General Flow of a Nonstandard Label Processing Routine After Receiving Control from the Close Routine

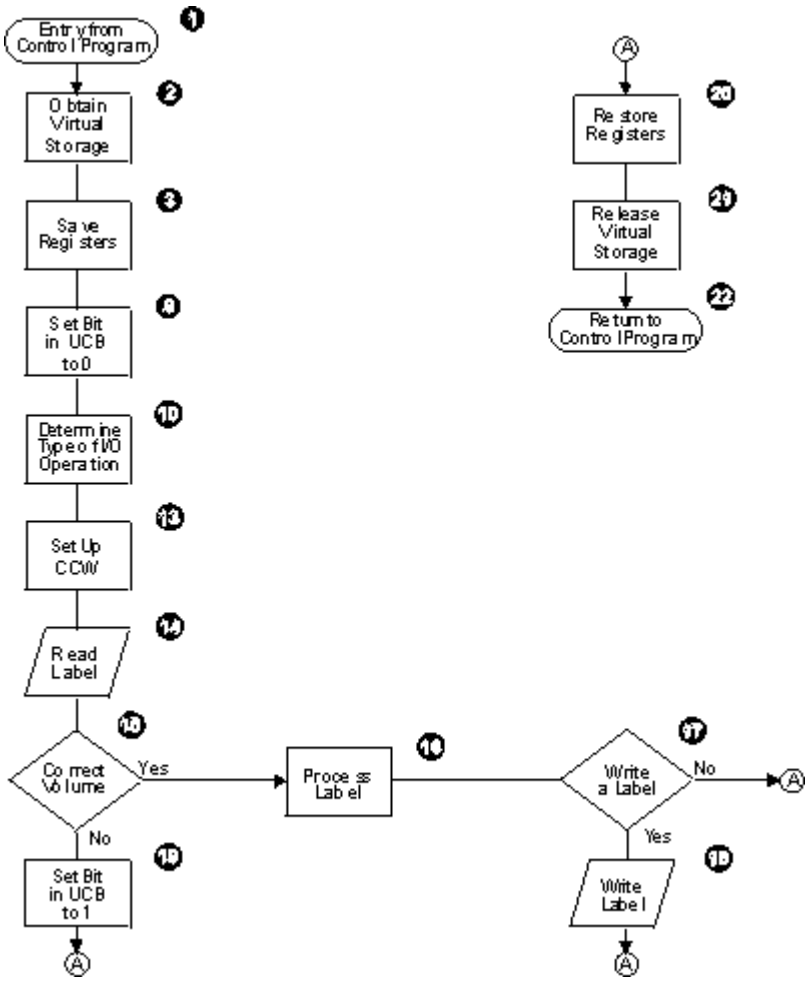


Figure 35. General Flow of a Nonstandard Label Processing Routine After Receiving Control from the EOV Routine

Explanation of Logic Blocks-Figures Figure 33 on page 110, Figure 34 on page 111, and Figure 35 on page 112

1

The entry is in the form of an XCTL macro instruction issued by the control program.

2

Use the GETMAIN or STORAGE macro instruction to obtain virtual storage.

3

Use the store multiple (STM) instruction and then set a register to point 32 bytes past register 6 as described in Figure 30 on page 108.

4

To locate the address of the DCB, use the contents of register 5 (Figure 30 on page 108). Set register 4 to the address of the work area pointed to by the table entry described in Figure 30 on page 108. (For EOV, this register is already set.)

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 is not processed, and its entry is all zeros in the table pointed to by register 6 at entry.

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. Register 4 points to this IECDSECT area. (The IECDSECT macro is described in "Mapping the Common Open, Close, EOV Work Area" on page 109.)

6

If nonstandard labels have been specified, the JFCBLTYP field of the JFCB has a hexadecimal 04.

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 the register set in Step 3.

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 the DCB entry in the list of DCB addresses. The bit configuration for each type of I/O operation is shown as follows. (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 or EXTEND
x	x	x	x	0	1	1	1	OUTIN or 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

If the high-order bit of the DCBOFLGS field of the DCB 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 can 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
Use this block for a write operation.

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, use the compare and swap instructions to set the rewind-issued bit in the UCB (UCBREW) before issuing the EXCP.

If the command is a rewind-unload, use the compare and swap instructions to set the unit-not-ready bit in the UCB (UCBNRY) 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.

This coding sequence illustrates an exit from your routine. At entry to the module, register 4 contains the address of the control program's work area for a DCB. The code following represents a generic 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/EOV area
         LA        15,DXCCW12           use open work area
         XCTL      EPLOC=(6),SF=(E,(15))
MODNAME  DC        C'IGGxxxxx'
```

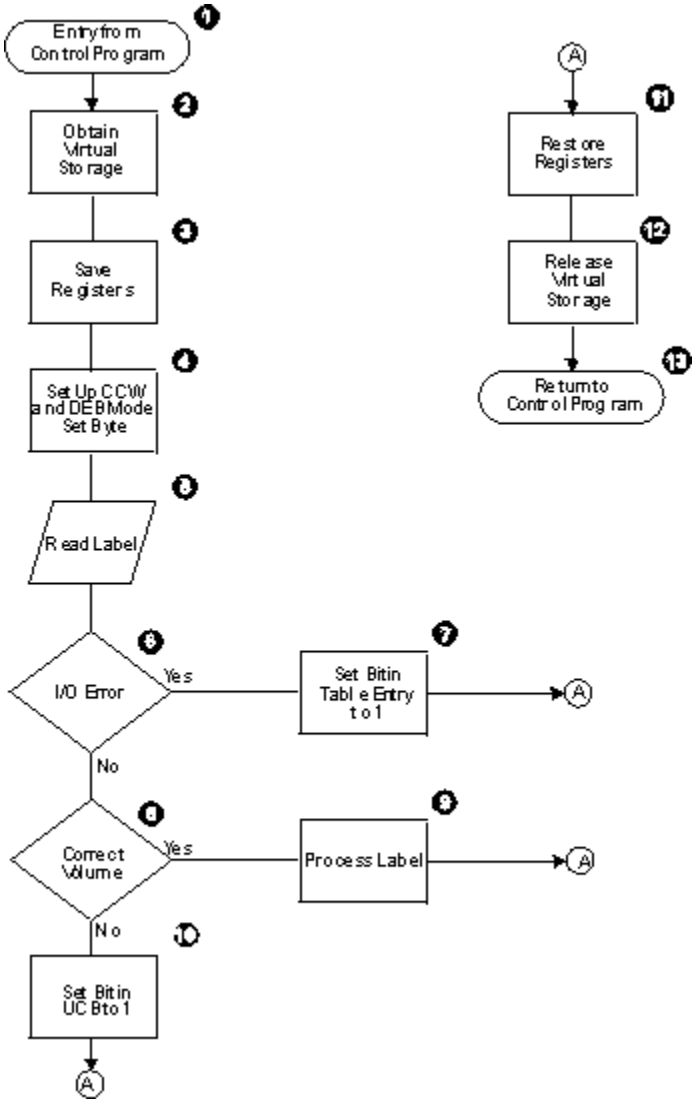


Figure 36. General Flow of a Nonstandard Label Processing Routine After Receiving Control from the Restart Routine

Explanation of Logic Blocks- Figure 36 on page 115

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 control program provides the device modifier byte, RSDEBMOD, in the DEB portion of the restart work area. RSDEBMOD contains the mode-set command for the data portion of the tape. If your nonstandard labels are recorded in a mode different from 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 *z/OS DFSMS Using Magnetic Tapes*.)

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 to indicate the I/O error.
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
         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'
```

Automatic Volume Recognition (AVR) Nonstandard Label Processing Routine (IEFXVNSL)

To allow the AVR option to process nonstandard magnetic tape labels, you must write a routine to supply AVR with information about them. Insert it into the control program replacing the IBM-supplied routine that causes AVR to reject tape volumes with non-standard labels. Your routine returns information to AVR that consists of a validity indication and the location within the nonstandard label of the volume serial number field.

OPEN, EOV, and CLOSE will always turn the DEB31UCB on in the work area to signify that DXDEBUCB is a 31-bit UCB address. Although the UCB address field will be four bytes, it typically will contain a three-byte address. The modeset byte is moved from the DEBSDVM field to the DEBSDVMX field to enable 31 bit UCB addresses to be accommodated.

Installing the AVR Nonstandard Label Routine

See “Replacing an Existing Exit” on page 3. Your routine must have an entry point name that matches the exit name. Your CSECT is linked together with certain system CSECTs into a single load module.

Writing the AVR Nonstandard Label Processing Routine

Your routine must:

- Determine if the label under consideration is a valid, nonstandard label as defined by your installation.
- 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.)

  Place the location of the volume serial number field within the label in an area provided by AVR when a valid label is detected. (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 inter-record gap after the nonstandard label.)
- Return control to AVR. Register 14 contains the return address. (The SAVE and RETURN macro instructions can 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 118.

The format of your nonstandard labels must provide for a 6-byte volume serial number field within the first 80 bytes of the label. Otherwise, the volume serial number is 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.

Your routine is called in 24-bit mode and must return in that mode. The name of your routine must be IEFXVNSL.

Registers on Entry to the AVR Exit Routine

Register
Contents

1
Address of AVR parameter list

2-12
Unpredictable

13
Address of a save area

14
Return address to the calling program

15
Address of the entry point to IEFXVNSL

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:

AVR Parameter List

Table 25. AVR-Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	Address of 80-byte work area that contains the nonstandard label.
04 (X'04')	4	Address of 4-byte area where you place the address of the volume serial number field within the nonstandard label.

NSL Volume Verification with Dynamic Device Reconfiguration (NSLREPOS)

If you use nonstandard tape labels and you want to use the dynamic device reconfiguration (DDR) option, you need an exit routine to perform volume verification.

When your NSLREPOS routine receives control from the DDR tape reposition routine, register 2 contains a pointer to an XCTL list. This list contains the module name to which you transfer control when you return control to DDR. Register 5 points to a buffer 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 UCBVOLI field of the UCB. Register 7 contains the UCB address.

If NSLREPOS gives return code 0, 12, or 16, the system's reposition routine repositions the tape. With a reel tape, this requires a count of the blocks from the labels or tape mark and this may depend on application program logic. With a cartridge tape, repositioning does not require a block count; the system uses a block identifier which is faster and more reliable.

Your NSLREPOS routine is not link-edited with any IBM-supplied routine. It resides in LPA and can have any valid combination of AMODE and RMODE. All the supplied areas reside below the 16MB line.

OPEN, EOV, and CLOSE will always turn the DEB31UCB on in the work area to signify that DXDEBUCB is a 31–bit UCB address. Although the UCB address field will be four bytes, it typically will contain a three-byte address. The modeset byte is moved from the DEBSDVM field to the DEBSDVMX field to enable 31–bit UCB addresses to be accommodated.

Before returning control, your routine should put one of these codes into register 0:

Code
Description

0 (X'00')
Volume verification is complete and a tape mark follows this label. Tape reposition routine positions the tape to that tape mark and clears 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 and no tape mark follows this label. The tape reposition routine repositions the volume, using the block count it has accumulated.

16 (X'10')
Volume verification is complete and the tape mark 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, use this sequence:

```
LR     15,2
XCTL   SF=(E,(15))
```

Volume Label Verification and Volume Label Editor Routines

(OMODVOL1, EMODVOL1)

If you state 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 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 label anomaly exit and under certain conditions to a volume label editor routine. Your installation can use routines supplied by IBM or you can supply your own routines. The system calls the label anomaly exit, IFG019LA, before a possible call to a volume label editor routine. The type of anomaly and the actions taken by the label anomaly exit determine whether the system calls the volume label editor. IBM recommends using the label anomaly exit instead of the volume label editor because the label anomaly exit can handle more situations, its interface is cleaner and it is expected to be easier to write. IBM does not plan to enhance the label editor function.

The IBM-supplied volume label editor routines find 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 under the following conditions:

- If you use a single-density 9-track tape unit, 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 you use a dual-density 9-track tape unit, the record is read in its existing density, provided that density is available on the unit. If the density is not available, or if the record is a 7-track record, a unit check occurs.
- If you use a 7-track tape unit, 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 you use a cartridge unit, the record is read in the density of the unit. If the record cannot be read, a unit check occurs.

Various error conditions can occur when the system verifies the initial record on a tape. The following figures illustrate system actions (when standard labels are specified, when nonstandard labels are specified, and when no labels are specified), resulting from these error conditions.

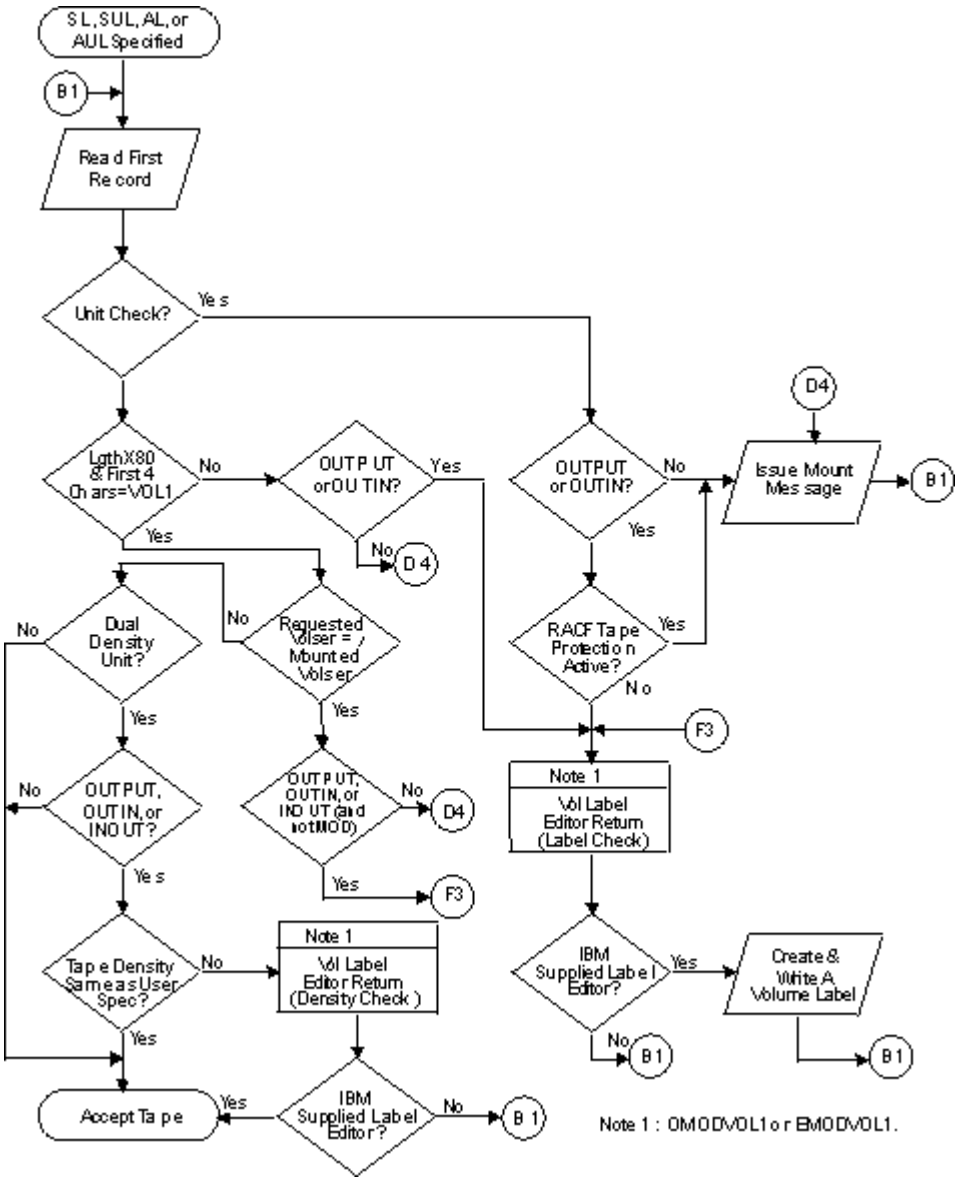


Figure 37. Verification of First Record When Standard Labels Are Specified

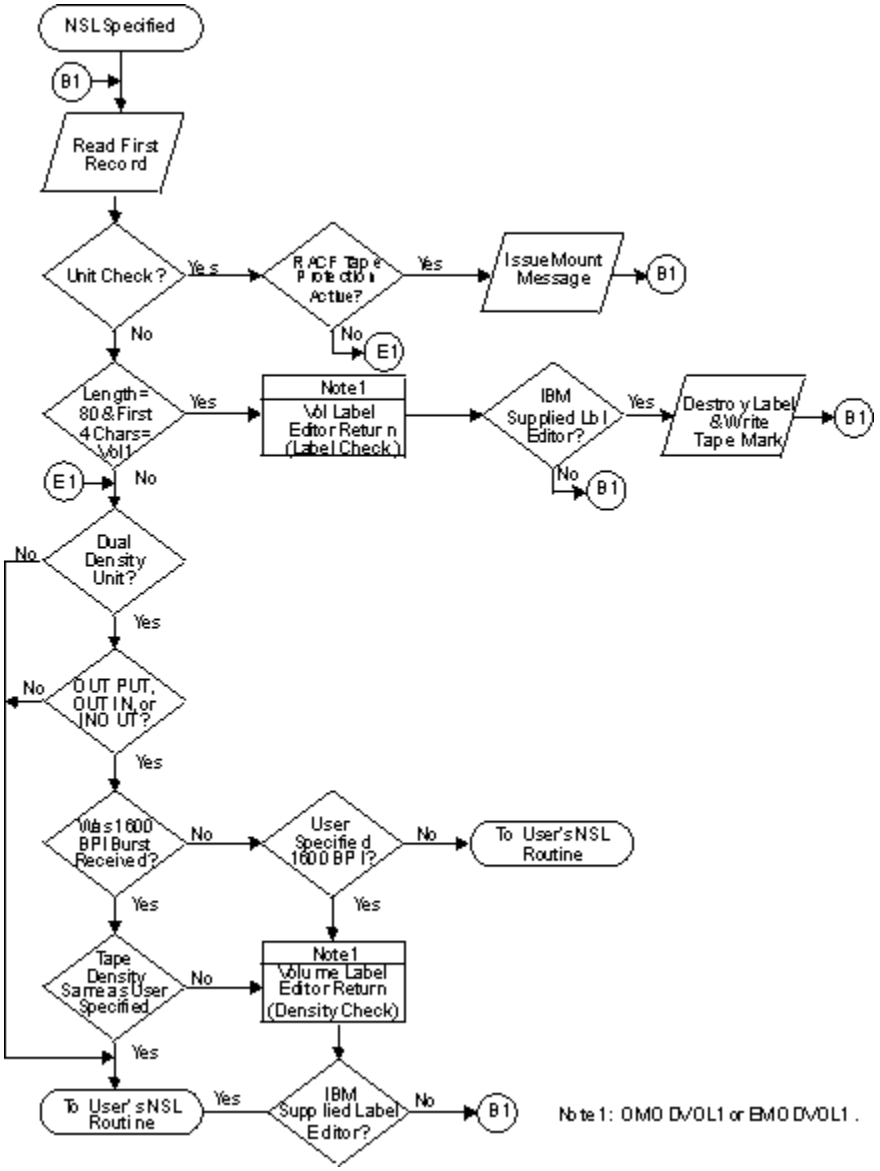


Figure 38. Verification of First Record When Nonstandard Labels Are Specified

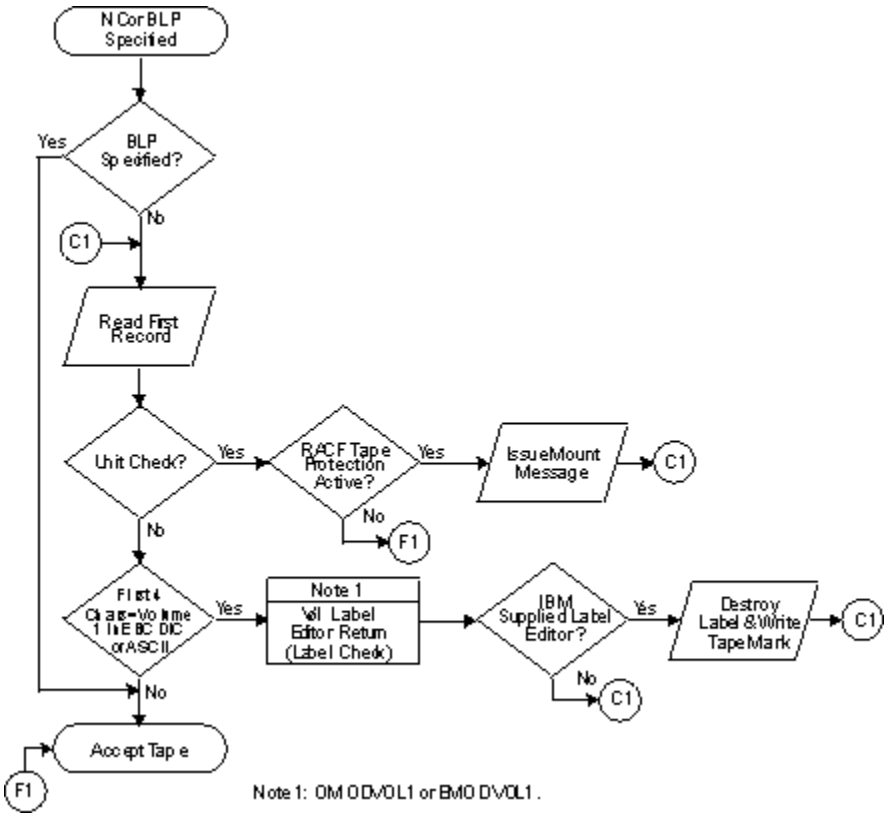


Figure 39. Verification of First Record When Unlabeled Tape Is Specified

Volume Label Editor Routines

When data sets are written on tape, data management's open or EOV routine might detect these:

- **Label—Type Conflict.** Conflict between the label type specified by the user and the actual label type on the mounted output volume (OUTPUT or OUTIN).
- **Density Conflict.** Conflict between 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.
- **Volume Serial Conflict.** Conflict between the volume serial number specified by the user and the actual volume serial number on the mounted output volume (OUTPUT or OUTIN).
- **Label Version Conflict.** Conflict between the existing label version on the mounted output volume and ISO/ANSI Version 3 or Version 4 level that open or EOV want to use either by default or by using the force option in the VOLM (Volume Mount Tape Installation exit) or by the FORCE3 or FORCE4 parameter specified in the ALVERSION keyword of the DEVSUPxx Parmlib member.
- **Mode Conflict.** Conflict between the compaction mode specified by the user and the actual compaction mode of the mounted output volume (OUTPUT or OUTIN).
- **Track Conflict.** Conflict between the cartridge type of the mounted output volume and that of the drive (OUTPUT or OUTIN). The first block of the tape cartridge cannot be read. The system has reconstructed an SL or AL volume label from sense bytes. The first 11 bytes match the volume label. The rest of the 80 bytes are EBCDIC or ASCII blanks according to whether the first four bytes are EBCDIC or ASCII. If the label is in ASCII, OPEN or EOV has translated it to EBCDIC.

If the internal volume serial number does not equal the external (optical) volume serial number, create the internal equal to the external in the volume label editor routines for nonspecific volume requests. 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 Version 3 or Version 4 volume, the system enters RACF, checks the expiration date, and enters 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—type 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 either the volume serial number and owner information be supplied or, optionally, that the use of this tape volume be refused. 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 rejects the tape and issues 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 passed, or the operator allowed the use of the tape.
- The volume is neither password-protected nor RACF-protected, and the accessor is ALTER authorized.

If these 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. It deletes the RACF definition of the volume if it was found to be RACF defined and the user is ALTER authorized. 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.

If the existing ISO/ANSI label is not Version 3 or Version 4, during an output request to the first data set on the volume, the volume label editor routines provide an option allowing the label to be rewritten to conform to Version 3 or Version 4 standards. The WTOR message processing installation exit can be used to provide label information for the new Version 3 or Version 4 label instead of requiring the operator to supply it through a WTOR message (see “WTO/WTOR Message Processing Facility Installation Exit (IEAVMXIT)” on page 133). 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 your routines to resolve the conflict. Your editor routines can:

- Resolve label and density type 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.
- 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.
- Provide a combination of these actions, including removing the volume under certain conditions.

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.

Installing Your Own Label Editor Routines

See “Replacing an Existing Exit” on page 3. For OMODVOL1, code IFG0193C on the ++MOD statement. For EMODVOL1, code IFG0553C on the ++MOD statement.

Note: OMODVOL1 and EMODVOL1 are aliases.

Your label editor routines can reside above or below the 16MB line.

Writing Volume Label Editor Routines

Your editor routines must conform to the same general programming conventions as the nonstandard label processing routines discussed under "Processing Nonstandard Labels" on page 102, for size, design, register usage, entry points, and work areas. As discussed under "Nonstandard Labels" on page 102, 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:

IFG0193C
The editor routine associated with open

IFG0553C
The editor routine associated with EOV

If your editor routines have 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.

DFSMSrmm makes use of the OMODVOL1 and EMODVOL1 routines. DFSMSrmm changes the flow of control so that DFSMSrmm code is entered before OMODVOL1 and EMODVOL1 are entered.

With tape cartridges, 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 run. This simulation improves performance.

If your routine does I/O, it should use the DCB that is in the work area. Do not substitute or modify the DEB appendage vector table.

Program Functions

Table 26 on page 125 presents the conditions under which the open or EOV routines transfer control to your editor routines. Each condition suggests what your routine can do to permit processing of the current volume to continue. The first two conditions (density—type conflicts) arise only when the tape volume is mounted on a dual-density tape device.

General flowcharts of editor routines are shown in Figure 40 on page 126 and Figure 41 on page 127. The logic is shown separately for routines that receive control from the open or EOV routine of the control program. Each block in each of the flowcharts is numbered; each number corresponds to an item in the lists of explanations that follow the figures. Other points to note are:

- The logic in the flowcharts is oriented toward resolving the label and density—type conflicts by altering the characteristics of the mounted volume.
- Figure 41 on page 127 (the EOV editor routine) does not contain logic blocks corresponding to blocks 5, 18, and 19 in Figure 40 on page 126 (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 you do not support expiration date and protection checking on nonstandard label volumes, or 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, or EOV processing. During open processing, register 7 points to a parameter list that contains the addresses of the DCBs 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.

Table 26. Editor Routine Entry Conditions from the Open and EOV Routine

Program Specification	Mounted Volume Characteristics	Transfer Conditions	Possible Editor Routine Action
SL or AL	NSL or NL"1" on page 126	Label—Type Conflict"2" on page 126	Write a standard volume label. (See Figure 40 on page 126: blocks 15, 15A, and 16. If you support protection and retention date checking on NSL volumes, see block 6.)
NSL or NL	SL"3" on page 126 or AL"4" on page 126	Label—Type Conflict	Overwrite standard label with tapemark, for example, cancel. (See Figure 40 on page 126: blocks 15, 15A, and 16.) Depending on whether NL or NSL is specified by the program, open or EOV either positions tape (NL) or transfers control to your nonstandard label routines (NSL).
AL	SL	Label—Type Conflict	Overwrite an IBM standard label with a Version 3 or Version 4 VOL1 label.
SL	AL"4" on page 126	Label—Type Conflict	Overwrite ISO/ANSI label with an IBM standard label.
AL or SL	AL"4" on page 126 or SL	Density Conflict	Overwrite the existing standard label with the requested standard label. The first write from load point sets the recording density on a dual-density device. (See Figure 40 on page 126 or Figure 41 on page 127: blocks 15B, 16, and explanation.)
NSL	NSL or NL with different density	Density Conflict	Write a tapemark to set density. The program specification NSL gives control to your nonstandard label routines after return to Open or EOV. (See Figure 40 on page 126: blocks 15, 15B, and 16. If your installation supports protection and retention date checking on NSL volumes, see block 6.)
AL or SL	AL or SL	Volume Serial Conflict	Overwrite volume label with requested volume serial number.
AL	AL"4" on page 126	Version Coexistence Conflict	Overwrite an ASCII label with a Version 3 or Version 4 label (first file output only).
Compacted or non-compacted mode	Opposite of mode specified	Mode Conflict	No action taken.
Cartridge	36-track cartridge mounted on an 18-track drive	Track Conflict	Overwrite existing volume label so that it is compatible with drive.

Legend:

AL
ISO/ANSI standard volume label

SL
IBM standard volume label

NSL
Nonstandard volume label

NL
No volume label

Notes:

1. If the volume is mounted on a dual-density device, a density condition might also exist. The write operation corrects this.
2. When SL is specified, a label—type conflict might also indicate that the system could not recognize the first record because of a unit check condition.
3. 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.
4. The open and EOV routines position the tape at load point before transferring control to the editor routines.

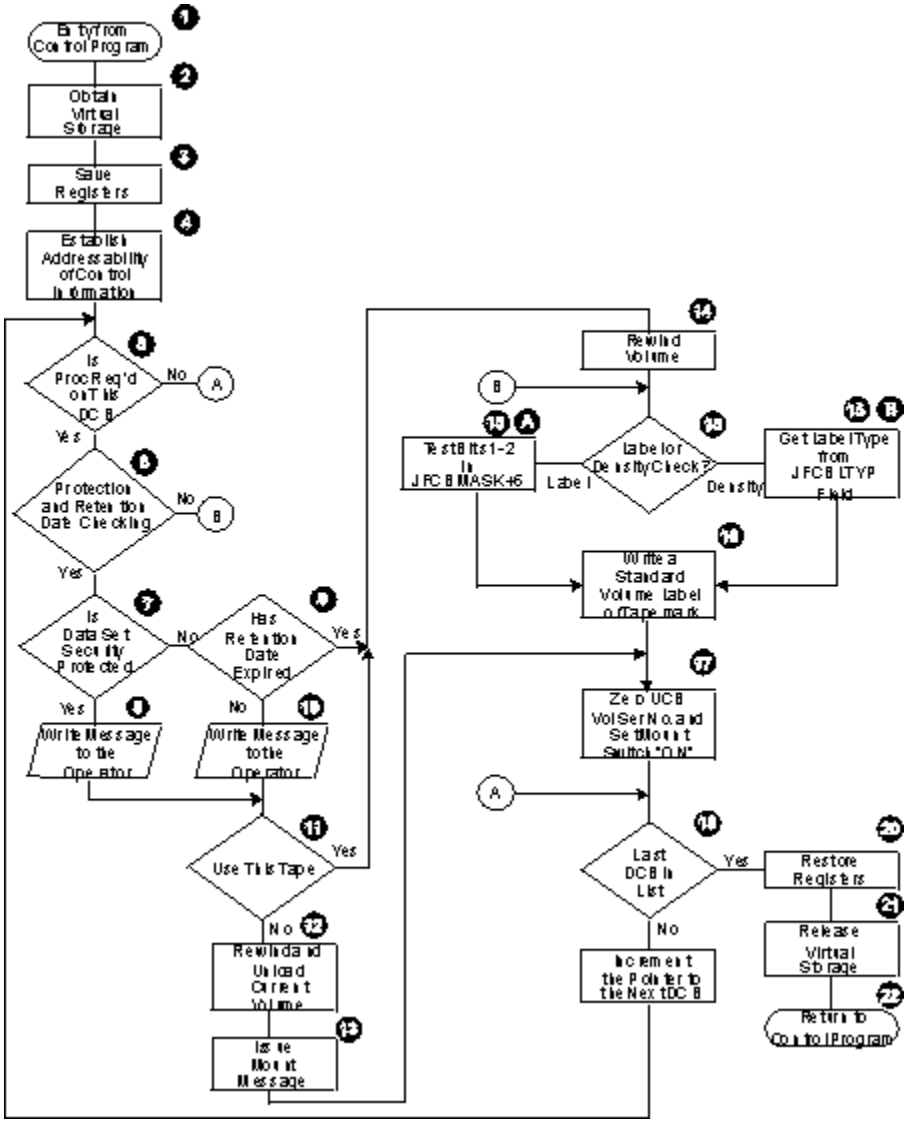


Figure 40. General Flow of an Editor Routine after Receiving Control from the Open Routine

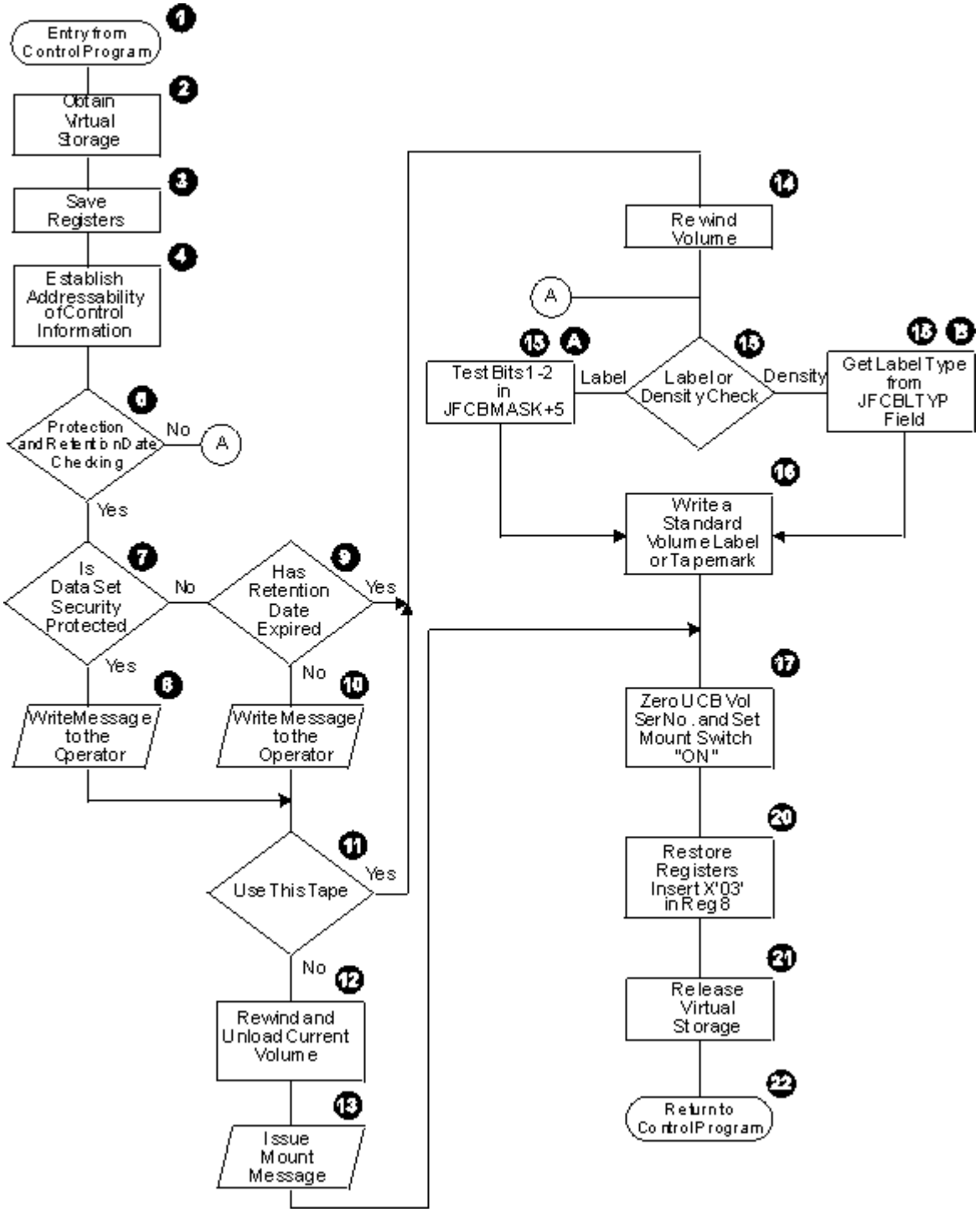


Figure 41. General Flow of an Editor Routine after Receiving Control from the EOV Routine

Explanation of Logic Blocks-Figures Figure 40 on page 126 and Figure 41 on page 127

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 get 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 30 on page 108 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 Open, Close, EOV Work Area" on page 109) symbolically defines the fields of the work and control block area (see Figure 31 on page 109).

Also, address the UCB for the device on which the tape volume is mounted. You can get the address of the UCB from the DXDEBUCB field of the DEB defined by the IECDSECT macro instruction. The IEFUCBOB macro instruction defines the fields of the unit control block.

5

Bit configurations in the byte addressed by JFCBMASK+5 indicate whether label—type conflicts or density conflicts have occurred and, in the case of a label—type conflict, the condition that caused the conflict. Now 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—type conflict has occurred.
1	1	Standard label (SL or AL) specified; no label/nonstandard label on mounted volume. If JFCBAL (AL label requested) is set and UCCBBSTR is set in the UCB (ASCII tape is mounted), an ISO/ANSI version conflict has occurred, and a valid Version 3 or Version 4 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 conflict
4	1	Track conflict. The system has reconstructed a volume label from sense bytes.
5	1	See step 12
6-7		Reserved for future use

6

If your installation supports a protection and retention date scheme involving nonstandard labels, or if 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.

To check, you must read the first record and determine the label type.

For I/O, 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. There are 12 CCW locations in the DXCCW field. There are 12 CCW locations in the DXCCW field. You can only use the first six locations.

7

To check the retention date 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 you define determines how you access those fields in the nonstandard label. Step 6 provides directions for handling the I/O operation.

8
Write a message to the operator stating that the volume is protected and asking if it is to be used.

9
Repeat step 7.

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 or EOV mount verification routines to handle the mounting or removing on volume verification, set bit 4 (X'08') of JFCBMASK+5 in the open or EOV work area and go to block 22 to return to open or EOV. Subsequent volume level errors cause the label editor routines to be reentered.

13
Write a message to the operator requesting removing of the current volume and mounting of a new volume. You can get the 4-digit device number (in binary) from the UCBCHAN 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 conflict.

Test bit 0 of the byte at JFCBMASK+5. If set to 1, control was received as the result of a label—type conflict.

a
If control was received as the result of a label—type conflict, 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 conflict, use the JFCBLTYP field in the JFCB to determine the type of label in the program. A X'04' indicates a NSL has been specified; a X'02' indicates a standard label has been specified.

16
When you correct a density conflict or label—type conflict condition, and the program specifies an NSL, record the tape whether the open or EOV routines interpret 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. If you've specified NSL, OPEN or EOV transfers control to your nonstandard label routines. If you've specified NSL, it positions the tape for writing.

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.

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. 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 causes the mount verification routines to bypass further label processing and reverify the tape without an intervening removal.

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 using the following code:

```
         MVC   0(L'IDRETURN,6),IDRETURN
         XCTL  EPLOC=(6),SF=(E,DXCCW12)
IDRETURN DC    CL8'nnnnnnnn'  Name of return routine
```

nnnnnnnn is the name of the routine to return to. In OMODVOL1 the name is IGG0190A. In EMODVOL1 the name is IGG0550P.

Return From
To Module

OMODVOL1
IGG0190A (Open)

EMODVOL1
IGG0550P (EOV)

Note: Open and EOV rewind the volume upon receiving control from OMODVOL1 or EMODVOL1.

ISO/ANSI Version 3 and Version 4 Installation Exits (IFG0193G)

Four installation exits are provided, as defaults, for ISO/ANSI Version 3 and Version 4 volumes:

1. Label validation
2. Label validation suppression
3. Volume access
4. File access.

A fifth installation exit, MPF for WTO/WTOR, can be written (or modified, if one has already been written) by your installation to convert ISO/ANSI pre-Version 3 to Version 3 or Version 4 labels (see “WTO/WTOR Message Processing Facility Installation Exit (IEAVMXIT)” on page 133).

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 run 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 remove volumes.

The return code from the exits can be modified to request continued processing. However, in cases in which the label—validation exit is entered and the routine has not been modified to correct certain errors,

the results can be unpredictable The prologue 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 uppercase A through Z for ISO/ANSI Version 3 tapes. In addition to uppercase A through Z, ISO/ANSI Version 4 tapes also allows special characters !*"%&'()+,-./:;<=>?_ and numeric 0–9.

Return codes from the Version 3 and Version 4 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 or Version 4 and RACF exits are not the same.

Neither the Version 3 or Version 4 nor RACF installation exits should alter any of the parameter list fields, except IECIEXRC or IECIEUSR.

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 version (UCBCXVER). The address of the appropriate UCB is maintained in the parameter list.

If you replace any of the IBM-supplied exit routines with your own routines, follow the programming conventions described under “Processing Nonstandard Labels” on page 102, except that return must be by way of a BR 14 instruction.

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.

MODESET to key 0 to alter protected control blocks (such as the UCB). Always restore the original key at entry immediately after you make any alterations to key 0 storage; this minimizes risk of inadvertent data destruction.

Label Validation Exit

The label validation exit is entered during open or 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 for Version 3, or a zero generation number), invalid label sequence, asymmetrical labels, invalid expiration date sequence for Version 3, and duplicate data set names for Version 3.

Input to the exit is the address of the exit parameter list containing the type of exit being run, the type and 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. If the same data set is read backward, the invalid label is detected during open or EOV for the new volume, and causes the label validation exit to be entered.

Because modifications to an existing data set can result in non-symmetrical trailer labels, these open options cause the label validation exit to be entered for Version 3:

- Open for OUTPUT or OUTIN with DISP=MOD.
- Open for 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.

If you have generalized library subroutine programs that use 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
Description

X'00'
Continue processing volume

X'04'
Reject volume (set by the IBM-supplied exit)

To identify the condition that is not valid, 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 a condition that is not valid, but the condition causes an abend in subsequent processing. The system does not rewrite labels after return from the label validation exit. To correct a label, you must write a label validation exit. If certain errors are not corrected, they cause unpredictable results when the volume is processed by a return code of zero from the label validation exit. They are:

- Incorrect sequencing
- Unsupported characters
- Incorrect field alignment
- Some unsupported values (RECFM=U, block size greater than 2048 for Version 3, and a zero generation number are processed by the system).

If an error is corrected by a return code of zero from the label validation exit, the resulting volume might not meet the specifications of Version 3 or Version 4 standards, and therefore requires agreement between interchange parties.

Label Validation Suppression Exit

The validation suppression exit lets you suppress label validation. It is entered during open or EOV if:

- volume security checking has been suppressed or
- the volume label accessibility field contains an ASCII space character or
- for Version 3 tapes, RACF accepts a volume and the accessibility field does not contain an uppercase letter from A through Z or
- for Version 4 tapes, RACF accepts a volume and the accessibility field does not contain an uppercase letter from A through Z or one of the special characters !*"%&'()+,-./:;<=>?_ or a numeric 0 through 9.

Label validation can also be suppressed by the volume access exit. If you suppress label validation, the resulting volume might not meet the specifications of Version 3 or Version 4 standards, and therefore would require agreement between interchange parties.

Volume Access Exit

The volume access exit is entered during open or 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 for Version 3 tapes or
- an ASCII uppercase letter from A through Z, a numeric 0 through 9, or special characters !*"%&'() +,-./:;<=>?_ for Version 4 tapes.

The exit is bypassed if 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 these return codes:

Return Code
Description

X'00'
Use volume

X'04'
Reject volume (set by IBM-supplied exit)

Suppress label validation 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.

The volume access exit is used only when the label validation suppression exit is not used.

File Access Exit

The file access exit is entered after positioning to a requested data set if the volume is not RACF protected and if the accessibility field in the HDR1 label contains:

- an ASCII uppercase letter from A through Z for Version 3 tapes or
- an ASCII uppercase letter from A through Z, a numeric 0 through 9, or special characters !*"%&'() +,-./:;<=>?_ for Version 4 tapes.

The exit is also entered when a data set is written to an output volume if the first character of the JCL ACCODE keyword is one of the characters listed in the previous paragraph for a Version 3 or Version 4 tape.

The exit can either accept the data set or reject the volume, issuing one of the following return codes:

Return Code
Description

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.

WTO/WTOR Message Processing Facility Installation Exit (IEAVMXIT)

For ISO/ANSI tape volumes, DFSMSdfp only supports output to ISO/ANSI Version 3 or Version 4 and input from either ISO/ANSI Version 1, Version 3 or Version 4. 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 get information for rewriting the volume label as a Version 3 or Version 4 label. If you do not want the operator to provide the label information (volume serial number, owner identification, and volume access code), you can use the WTO/WTOR message processing facility to intercept message IEC704A C and provide this information.

The name of the general purpose WTO/WTOR message processing installation exit is IEAVMXIT. You can use this installation exit or a message-specific installation exit routine (one that you specify on the USEREXIT parameter in the MPFLSTxx member of SYS1.PARMLIB) to process WTOR message IEC704A.

IECIEPRM Parameter List

The parameters passed to a Version 3 or Version 4 installation exit during label processing vary slightly among different types of exits. These differences, are noted in the "Description" column in Table 27 on page 134. 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. Parameter fields not available to a particular exit are set to zero. The only fields allowed to be altered by an exit are the return code (IECIEXRC) and the user area (IECIEUSR); changing any other field has an unpredictable effect on system processing. A flag in the parameter list indicates which type of exit was entered.

Table 27. ISO/ANSI Version 3 or Version 4 Exit Parameter List. All fields apply to all four exits unless otherwise stated in description.

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	4	IECIEID	Parameter list identifier (APRM)
04 (X'04')	4	IECIESIZ	Length of IECIEPRM
08 (X'08')	4		Reserved
12 (X'12')	1	IECIEFL1	Exit flags
EXACTLY ONE OF THE FOLLOWING FOUR BITS IS ON FOR EACH CALL			
	1...	IECIEVAL	Entry is Validity Check
	.1..	IECIEVAE	Entry is Volume Access
	..1. ...	IECIEFAE	Entry is File Access
	...1	IECIEVSP	Entry is Validation Suppression
	.... 1...	IECIEWRT	Label will be written (WRITE)
	.... .1..	IECIEEOV	EOV in process
13 (X'13')	1	IECIEERR	Validation error type (valid only for label validation exit)
	1...	IECIEVRS	Version coexistence conflict"1" on page 135
	.1..	IECIEUNK	Unsupported or unknown value
	..1.	IECIEADJ	Invalid field alignment
	...1	IECIESEQ	Label sequence error
	.... 1...	IECIEDUP	Duplicate file name
	.... .1..	IECIECHR	Invalid character type
	.... ..1.	IECIEXPR	Invalid expiration date
	1	IECIESYM	Symmetry conflict"2" on page 135
14 (X'14')	1	IECIEPOS	Starting character position in label examined"3" on page 135 (Valid only for label validation exit)
15 (X'15')	1	IECIEXRC	Return code from exit processing. Set by exit.
	1...	IECIESUP	Suppress label validation"5" on page 135
	.000 0000	IECIERC0	Accept volume
	.000 0100	IECIERC4	Reject volume (ignored for VSP Exit)
16 (X'16')	1	IECIEJAC	User-requested file accessibility code.
17 (X'17')	2		Reserved
19 (X'19')	1	IECIEDCB	Copy of open parmlist options (4 low order bits)
	.... ..1.	IECIEOUT	Bit on for OUTPUT, OUTIN
	X'0E'	IECIEIN	These are bits off for INPUT, RDBACK
20 (X'20')	4	IECIELBL	Address of label being processed"7" on page 135
24 (X'24')	4	IECIEUCB	Address of UCB for volume will be a 24 bit actual or captured address"8" on page 135
28 (X'28')	4	IECIEUSR	User area. Set to zero by the system before the first call. Set by exit for use on later calls.
32 (X'32')	0	IECIEND	End of exit parameter list

Notes:

1. "Version" error is set for the open, close, EOV message routine for internal use, and the volume is unconditionally rejected.
2. A symmetry conflict results from a condition that produces non-matching or asymmetrical labels framing a file, and/or inconsistent file structure.
3. The first character position is offset 0, the second position is offset 1, and so forth.
4. A return code of 4 is set by the IBM-supplied exits. This causes 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 always unconditionally accepts a volume after execution of the validation suppression exit). IECIEXRC is ignored by open or EOV when control returns from RACF.
5. IECIESUP is 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.
6. The file accessibility code in IECIEJAC is only valid when Write Mode (IECIEWRT) was set on by the caller of 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). If bit IECIEWRT is set and the value in IECIEJAC is valid as a file accessibility code, then IECIEJAC will be written in the file label when the exit returns.
7. 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 is zero. Binary zeros indicate data in the label that is not available to an exit. The volume accessibility code is always available in the UCB tape class extension at UCBCXACC (for ISO/ANSI) when an ISO/ANSI volume has been opened and not removed.
8. The UCB tape class extension for ISO/ANSI volumes 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:

```
LR    R1,xxx                GET UCB ADDRESS
IOSCMXR ,                   GET COMMON EXTENSION ADDRESS
USING UCBCMEXT,R1
L     Rx,UCBCLEXT           GET CLASS EXTENSION ADDRESS
DROP  R1
USING IECUCBCX,Rx           IECUCBCX MAPPING
```

Note that the UCB tape class extension might reside above the 16 MB line.

You can use the base UCB to access the serial number for the mounted volume (in UCBVOLI).

UCB Tape Class Extension-IECUCBCX

The tape class extension area generated for a UCB is addressed by UCBCLEXT in the UCB common extension. It contains zeros at IPL, and is set to zeros whenever the volume label is ready to be verified and processed for accessibility (as in open, or next volume for EOV). The class extension holds volume label data across opens when there is no intervening volume label reverification (as is true after CLOSE LEAVE and another OPEN in the same job step). The UCBCX DSECT in the IECUCBCX macro maps the tape class extension area. If you use this DSECT to map the area, you get the version of the tape class extension area that you need. To see other fields for diagnosis purposes, go to *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

The UCB tape class extension can reside above the 16 MB line, so your code should execute in 31-bit mode.

Table 28. UCB Tape Class Extension Data Area

Offset	Length, Bit Pattern or Value	Name	Description
8 (X'8')	1	UCBCXACC	VOL1 Access code from label
10 (X'A')	1	UCBCXVER	VOL1 Label-standard version
	"3"	UCBCXV3	Version 3

Table 28. UCB Tape Class Extension Data Area (continued)

Offset	Length, Bit Pattern or Value	Name	Description
	"4"	UCBCXV4	Version 4
12 (X'C')	14	UCBCXOWN	VOL1 owner identification

Chapter 4. Pre-ACS Installation Exit IGDACSDX

IBM has defined the pre-ACS routine exit (IGDACSDX) to the dynamic exits facility. The exit enables a tape management system to influence ACS routine construct selection. As SMS starts, the system will add the exit routine IGDACSXT to this exit. As shipped by IBM, this routine is a no-op. Through this interface, four new read-only variables can be set by your tape management system and then used in ACS routine processing:

- Pool name (&MSPOOL)
- Policy name (&MSPOLICY)
- Destination name (&MSPDEST)
- User parameter information (&MSPARM)

Prior to invoking the ACS routines, the exit is called via dynamic exit services, providing an opportunity for the tape management system to set the tape-management related read-only variables.

Note: DFSMSrmm does not use this exit. An equivalent interface exists. See *z/OS DFSMSrmm Application Programming Interface*.

Customers use their tape management system as a repository for movement rules of tape data sets to vaults or backup centers. In the allocation process, with a system-managed tape library, and especially with a VTS, the choice of a storage group needs to be influenced by whether or not the tape stays in the computing center or goes outside. For environments with multiple system-managed tape libraries and complex vaulting requirements, allocation to a particular library or VTS should be matched to the vaulting requirements of the data sets being allocated. Because this information is already available within the tape management system, this exit provides a way to make this information accessible to the ACS routines.

Allowing the tape management system's logic to more directly support tape allocation under SMS will avoid redundancy and duplication of effort. These same variables can also replace most of the filter lists currently used by customers using the IBM Tape Mount Management methodology. Though the pre-ACS routine exit is not limited to the usage described here, this is one of the ways in which this exit may be used by a tape management system.

If any exit routine associated with this exit abends, SMS ignores any information set by that exit routine and continues processing. By default, the system will not stop calling an exit routine even if it abends.

Updating the Pre-ACS Exit Routine

The IBM-provided Pre-ACS Exit Routine is a no-op. If you choose not to provide an exit routine with a name of your choosing, use SMP/E to replace the exit routine with the name IGDACSXT.

Controlling the Pre-ACS Exit Routine through the Dynamic Exits Facility

IBM has defined the IGDACSXT exit to the dynamic exits facility. You can refer to the exit by the name IGDACSDX. You can use the EXIT statement of the PROGxx parmlib member, the SETPROG EXIT operator command, or the CSVDYNEX macro to control this exit and its exit routines.

The exit routine should reside in LPA, the LNKLST concatenation, or the nucleus. Do not use the DSNAME keyword when adding the exit routine in PROGxx.

You can use the ADDABENDNUM and ABENDCONSEC parameters on the CSVDYNEX REQUEST=ADD macro or the ABENDNUM parameter of the SETPROG EXIT operator command to limit the number of times the exit routine abnormally ends before it becomes inactive. An abend is counted when the exit routine does not provide recovery, or the exit routine does provide recovery but percolates the error.

By default, the system does not disable the exit routine.

Characteristics of the Pre-ACS Exit Routine

The pre-ACS exit routine must:

- Handle multiple concurrent requests (be reentrant)
- Have AMODE 31

The exit routine is given control in task mode and PSW key zero with no locks held and in 31-bit addressing mode with PASN=HASN=SASN.

Registers on Entry to the Pre-ACS Exit Routine

When the pre-ACS exit routine gets control, the general-purpose registers have the following content:

Register
Contents

0
Not applicable

1
Address of the parameter list mapped by IGDACERO.

2-12
Not applicable

13
Address of register save area

14
Caller's return address

15
Address of the exit's entry point

Note: Because this routine is called before ACS routines, some variables, such as Data Class, will not have been set. JCL parameter values should be available.

Registers on Return from the Pre-ACS Exit Routine

When you return from the pre-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
No predefined reason codes and return codes are provided.

Chapter 5. Automatic Class Selection (ACS) Installation Exits

Each automatic class selection (ACS) routine, except the one for storage group, has an ACS installation exit that lets you write exit routines for additional capabilities. The exit routines you write are processed when the corresponding ACS routine is called.

The ACS installation exits reside in SYS1.LPALIB and are listed in Table 29 on page 139. ACS routines are part of system-managed storage and are described in *z/OS DFSMSdfp Storage Administration*.

Table 29. ACS Replaceable Modules

Module Name	Description	When Available
IGDACSDC	Data class exit	After the data class ACS routine has executed
IGDACSSC	Storage class exit	After the storage class ACS routine has executed
IGDACSMC	Management class exit	After the management class ACS routine has executed

ACS routines and ACS installation exit routines can perform many of the same functions. Wherever possible, use ACS routines because, compared with ACS installation exit routines, they are relatively easy to write, maintain, and modify. You do not need to re-IPL the system after you create or modify ACS routines; because ACS exit routines reside in SYS1.LPALIB, you must re-IPL after you change them. However, only the ACS installation exit routines can be used to:

- Call other programs.
- Call other subsystems.
- Write SMF records.
- Write GTF trace records.
- Take SVC dumps.
- Maintain large, easily searched tables of information in storage.

ACS services allow 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 recall an ACS routine one time to determine a new value for an SMS class. This new value can be assigned to a data set or be used for comparison with the original value assigned in step 1, Table 30 on page 140. Use the ACS interface routine to invoke an ACS routine. The parameter list that is passed to your exit routine, as shown in Figure 42 on page 143, contains a field (ACSPACS) that points to the ACS interface routine.

Installing the ACS Exit Routine

You can write routines for any or all of the ACS installation exits. You 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 written a routine for that ACS installation exit and ensured that it resides in SYS1.LPALIB. See “Adding a New Exit” on page 3.

Characteristics of the ACS Installation Exits

In general, the routines you write for the ACS installation exits must:

- Handle multiple requests (reentrant)
- Reside in SYS1.LPALIB
- Have AMODE 31
- Have RMODE ANY
- Be written in Assembler H or High-Level Assembler.

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 operate in cross-memory mode.

When writing 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. An ACS routine should not issue a dynamic allocation request, because a dynamic allocation request can invoke an ACS exit. During the course of allocation, the ACS routine or exit can be entered multiple times.

Linkage is with standard MVS linkage conventions.

Understanding the Automatic Class Selection Process

The system processes that result in the invocation of ACS routines are:

- Allocation of new data sets that are eligible to be SMS-managed.

  Restriction: The data class routine can also be executed for data sets that are not eligible to be system-managed. However, if the ACS routine detects a data set that is not eligible to be system-managed, the following should occur:

  – If a data class is specified, it should be assigned to that data set.
  – If a storage class or management class is specified, the allocation should fail followed by an error message.
- Conversion of SMS volumes and data sets
- DFSMShsm recall and recover
- DFSMSdss COPY, RESTORE, and CONVERTV commands
- Access method services ALLOCATE, DEFINE, and IMPORT commands
- Object access method (OAM) STORE, CHANGE, and CTRANS.

Table 30 on page 140 summarizes what happens to each of the SMS constructs when automatic class selection takes place.

Table 30. Automatic Class Selection Process

Step	Action
1	Assign data class: **Step** **Action** **A** Call data class ACS routine, if it exists, to assign a data class. **B** Call data class ACS installation exit. See note. **C** Validate data class name. ACS services ensures that the name assigned is for a defined data class.

Table 30. Automatic Class Selection Process (continued)

Step	Action
2	Assign storage class: **Step** **Action** **A** Call storage class ACS routine to assign a storage class **B** Call storage class ACS installation exit. See note. **C** Validate storage class name. ACS services ensures that the name assigned is for a defined storage class. **D** Check user's authority to determine if she or he is allowed to use the storage class.
If no storage class is assigned in the previous step, the following steps are **not** taken:	
3	Assign management class **Step** **Action** **A** Call management class routine to assign a management class. **B** Call management class ACS installation exit. See note. **C** Validate management class name. ACS services ensures that the name assigned is for a defined management class. **D** Check user's authority to determine if she or he is allowed to use the management class.
4	Assign storage group **Step** **Action** **A** Call storage group ACS routine to assign a storage group. **B** Validate the storage group name. ACS services ensures that the name assigned is for a defined storage group.

Note: The ACS installation exit routines can override the class assignments made in the ACS routine. Also, ACS installation exit routines can alter the input to ACS routines and recall them one time. The installation exit is not recalled when it recalls the ACS routine.

Recovery Environment for ACS Exit Routines

SMS establishes a recovery environment for the ACS installation exits by issuing an ESTAE before invoking them. ACS installation exit errors occur when:

- The SMS ESTAE exit covering the exit is entered.
- The ACS exit returns an invalid return code.
- The ACS exit returns return code 16.

If any of these errors occur,

- An output message describes the error.

- An SVC dump is taken, SYS1.LOGREC error recording is done, and the failing ACS exit is marked not valid.
- The failed ACS exit is not recalled until the SMS address space is restarted The SMS address space can be restarted with an IPL, or by issuing the SET SMS command if the SMS address space has been ended and not automatically restarted.

Registers on Entry to the ACS Exit Routines

When your routine gets control, the general-purpose registers have the following content:

Register
Contents

0
Not applicable

1
Address of a fullword that contains the address of the parameter list mapped by IGDACSPM.

2-12
Not applicable

13
Address of register save area

14
Caller's return address

15
Address of the exit's entry point

Using the ACS Exits Parameter Lists

Figure 42 on page 143 illustrates the parameter structure for the ACS installation exits. A 4 KB 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 exit can reference when selecting an SMS class.

IGDACERW
Maps the read-write variables that the ACS exit can set when selecting an SMS class.

IGDACSPM
Describes the parameter list for an ACS installation exit.

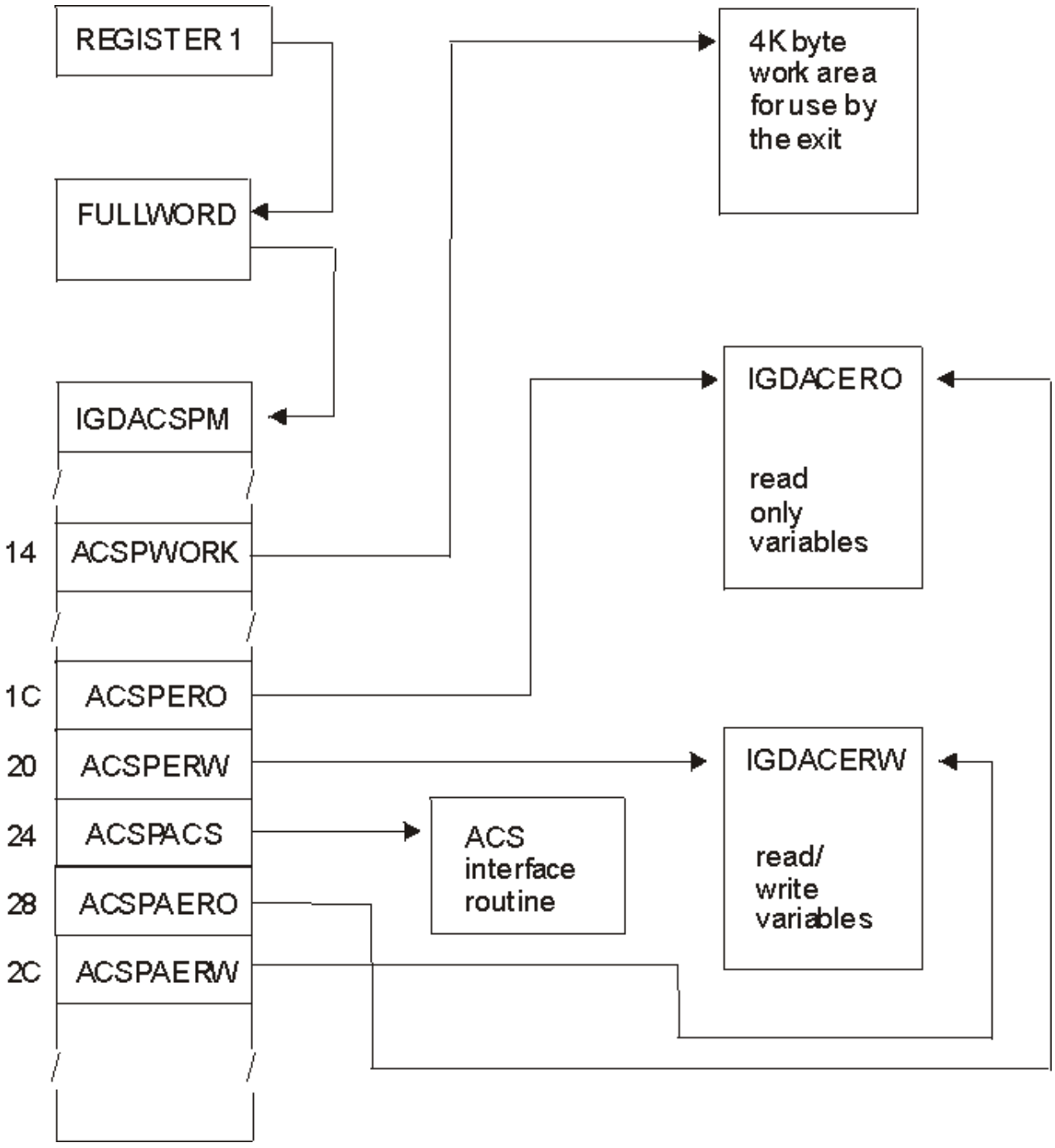


Figure 42. Parameter Structure for the ACS Installation Exits. This figure shows the control block structure upon entry into the exit. All offsets are in hexadecimal.

Assigning Classes

When entering an ACS installation exit, ACSPERW points to a list of read-write variables which are mapped by IGDACERW. Initially the list of read-write variables contains the original value for the SMS class that was assigned in the ACS routine. To assign an SMS class from an ACS installation exit, your routine must, 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 a length value based on the logic in the local ACS routine. If more than one SMS class is returned, only the first is accepted.
- Set the ACERWVLN field to the actual length of the SMS class name. Trailing zeros and blanks should not be included in the length. Initially, this field contains the length value of the SMS class being passed from the local ACS routine.
- Set the ACERWVAL field to the name of the SMS class being assigned. Initially, this field contains the SMS class value passed from the local ACS routine.

To assign a null value to an SMS class, you must set the ACERWNCS field to one and the ACERWVLN field to zero. The ACERWVAL field is then ignored. If you do not want to assign a class, set both the ACERWNCS and ACERWVLN fields to 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 and place the text of the messages in ACERWMSG. ACERWMSG can hold up to six messages that must be 110 bytes long. Pad messages with blanks if needed.

If your ACS installation exit routine invokes the ACS interface routine, then the ACS routine might have created messages by issuing the WRITE statement. The system retains those messages in ACERWMSG and sets ACERWNMG equal to the number of messages. The system has not written them yet. After the ACS installation exit routine ends, the messages will be written along with any messages that the ACS installation exit routine adds. Your routine should not overlay messages that are already in ACERWMSG.

Invoking ACS Interface Routine from an Exit

Your ACS installation exit routine can use the ACS interface routine to call an ACS routine. The ACSPACS field shown in Figure 42 on page 143 contains the address of an ACS interface routine that calls the corresponding ACS routine for the SMS class being selected. If the ACSPACS field contains a zero, then no ACS routine exists for the SMS class. The return code is in register 15. If it is nonzero, the reason code is found in register 0.

Linkage is that of standard MVS linkage conventions. Figure 43 on page 145 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. ACERWVLN and ACERWVAL fields zero.

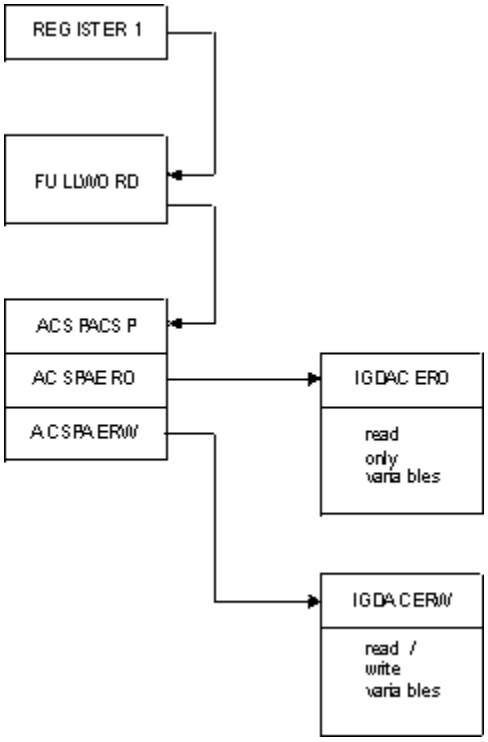


Figure 43. Parameter Structure for the ACS Interface Routine. This figure shows the control block structure for calling the ACS interface routine from the installation exit.

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 call the ACS routine. Or you can write the ACS exit routine to create an entirely new list of read-only variables for the ACS interface routine and point to them with ACSPAERO before calling 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 assigned in the ACS routine.

You can call the ACS routine without changing the value in ACSPAERW. When the ACS routine runs, 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 is the new class that is created when your exit routine calls the ACS routine. However, if the new class is to be used as the input class by the ACS routine, then it must be copied into the corresponding field in ACSPAERO from ACSPAERW.

You can also write your ACS exit routine to create an entirely new list of read-write variables and point to them with ACSPAERW before calling the ACS routine. After the ACS routine runs, 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 are assigned by the ACS routines. By creating a new read-write variable list in your exit routine, and then calling the ACS routine, you can compare the original values, pointed to by ACSPERW, with the ne values, pointed to by ACSPAERW.

If you create a new list of read-write variables and call the ACS routine, and you want 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 are *not* used to assign a class.

A zero in the ACERWNCS field upon return from the ACS routine indicates there is a null value 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.

ACS Installation Exits Parameter List (IGDACSPM)

Table 31 on page 146 shows the parameter list for the ACS installation exits. The parameter list is mapped by the IGDACSPM macro.

Table 31. ACS Installation Exit Parameter List (IGDACSPM)

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	52	ACSPM	ACS parameter list
00 (X'00')	8	ACSPID	Control block ID='IGDACSPM' (character)
08 (X'08')	2	ACSPLEN	ACSP control block length (signed)
10 (X'A')	2	ACSPVER	Control block version (binary)
12(X'C')	4	ACSPACEE	ACEE pointer
16 (X'10')	4		Reserved
20 (X'14')	4	ACSPWORK	Pointer to a work area for the exit
24 (X'18')	4	ACSPWLEN	Length of work area
28 (X'1C')	4	ACSPERO	Pointer to read-only variables mapped by IGDACERO
32 (X'20')	4	ACSPERW	Pointer to read-write variables mapped by IGDACERW
36 (X'24')	4	ACSPACS	Pointer to interface routine for calling the ACS routines. If this field is zero, then no ACS routine exists for the current SMS class.
40 (X'28')	12	ACSPACSP	Parameters for ACS routines
40 (X'28')	4	ACSPAERO	Pointer to read-only variables initially set to ACSPERO
44 (X'2C')	4	ACSPAERW	Pointer to read-write variables, initially set to ACSPERW
48 (X'30')	4	ACSPATOK	Token for use by ACS interface routine (do not modify)

Read-Only Variables Parameter List (IGDACERO)

Table 32 on page 146 shows the parameter list for the read-only variables that is mapped by the IGDACERO macro. For additional information about read-only variables, see *z/OS DFSMSdfp Storage Administration*.

Table 32. Read-Only Variables (IGDACERO)

Offset	Length or Value	Name	Description
0 (X'0')	1524	ACERO	Read-only variables parameter list
0 (X'0')	8	ACEROID	Control block ID contains the value "ACERO"
8 (X'8')	2	ACEROLEN	Length of control block
10 (X'A')	2	ACEROVER	Control block version number
	0	ACEROV	Version 0
12 (X'C')	4	ACEROSIZ	Primary or actual size of data set in KB

Table 32. Read-Only Variables (IGDACERO) (continued)

Offset	Length or Value	Name	Description
16 (X'10')	4	ACEROMSZ	Maximum size of data set in KB
20 (X'14')	8	ACEROUNT	Unit name (character)
28 (X'1C')	8	ACEROMVG	MSS volume group name (character)
36 (X'24')	8	ACEROAPP	Application ID (RACF) (character)
44 (X'2C')	8	ACERODSO	Data set owner (RACF) (character)
52 (X'34')	8	ACEROUSR	User (RACF) (character)
60 (X'3C')	8	ACEROGRP	Group (RACF) (character)
68 (X'44')	4	ACERODSG	Data set organization (unsigned binary). Can be one of the following:
	0	ACERONUL	Null
	1	ACEROPS	PS-physical sequential
	2	ACEROPO	PO-Partitioned
	3	ACEROVS	VS-/VSAM Organization
	4	ACERODA	DA-Direct Organization (BDAM)
	5	ACEROEXC	Extended Data Format, preferred
	6	ACEROEXR	Extended Data Format, required
72 (X'48')	4	ACERORCG	Record organization. Can be one of the following:
	0	ACERONUL	
	1	ACEROKS	VSAM key-sequenced data set
	2	ACEROES	VSAM entry-sequenced data set
	3	ACERORR	VSAM Relative-record data set
	4	ACEROLS	VSAM linear data set
76 (X'4C')	4	ACERODST	Data Set Type. Can be one of the following:
	0	ACERONUL	Null
	1	ACEROGDS	One generation data set of a generation data group
	2	ACEROPRM	Standard permanent data sets
	3	ACEROTMP	Temporary data sets
80 (X'50')	4	ACEROXMD	Execution mode. Can be one of the following:
	0	ACERONUL	Null
	1	ACEROBCH	Batch execution mode
	2	ACEROTSO	TSO execution mode
	3	ACEROTSK	Started task
84 (X'54')	8	ACEROJOB	Job name
92 (X'5C')	8	ACERODD	Ddname
100 (X'64')	8	ACEROPGM	Program name
108 (X'6C')	4	ACEROEXP	Expiration date (YYYYDDDF) in packed decimal
112 (X'70')	4	ACERORTP	Retention period days (binary)
116 (X'74')	1		Reserved

Table 32. Read-Only Variables (IGDACERO) (continued)

Offset	Length or Value	Name	Description
117 (X'75')	3	ACERODIR	Directory blocks from JCL
244 (X'F4')	8	ACEROENV	Environment. One of the following values or you can set your own in the exit. (Values are padded on the right with blanks).
	STORE	ACEROSTE	OSMI store environment
	CHANGE	ACEROCHE	OSMI change environment
	CTRANS	ACEROCTE	OSMC class transition environment
	RECALL	ACERORCL	Data set recall operations
	RECOVER	ACERORCV	Data set recover operations
	CONVERT	ACEROCNV	Data set convert in place operations
	ALLOC	ACEROALC	New data set allocations (default)
	ALLOCTST	ACEROTST	New data set allocations in the ACS Allocation Test environment
252 (X'FC')	32	ACERODDC	Default data class (from RACF)
252 (X'FC')	2	ACERODDL	Length of the default data class name
254 (X'FE')	30	ACERODDV	Name of the default data class
284 (X'11C')	32	ACERODSC	Default storage class (from RACF)
284 (X'11C')	2	ACERODSL	Length of the default storage class name
286 (X'11E')	30	ACERODSV	Name of the default storage class
316 (X'13C')	32	ACERODMC	Default management class (from RACF)
316 (X'13C')	2	ACERODML	Length of the default management class name
318 (X'13E')	30	ACERODMV	Name of the default management class
348 (X'15C')	80		Reserved
428 (X'1AC')	32	ACERODC	Data class input only. Output returned in IGDACERW.
428 (X'1AC')	2	ACERODCL	Length of DATACLAS variable
430 (X'1AE')	30	ACERODCV	Value of DATACLAS variable
460 (X'1CC')	32	ACEROSC	Storage class input only. Output returned in IGDACERW.
460 (X'1CC')	2	ACEROSCL	Length of STORCLAS variable
462 (X'1CE')	30	ACEROSCV	Value of STORCLAS variable
492 (X'1EC')	32	ACEROMC	Management class input only. Output returned in IGDACERW.
492 (X'1EC')	2	ACEROMCL	Length of MGMTCLAS variable
494 (X'1EE')	30	ACEROMCV	Value of MGMTCLAS variable
524 (X'20C')	44	ACERODSN	Data set name
568 (X'238')	8	ACEROGEN	Generation number

Table 32. Read-Only Variables (IGDACERO) (continued)

Offset	Length or Value	Name	Description
576 (X'240')	4	ACERODNT	Data Set name type. Can be one of the following:
	0	ACERONUL	
	1	ACEROLIB	LIBRARY (PDSE)
	2	ACEROPDS	PDS (partitioned data set)
	3	ACEROHFS	HFS (hierarchical file system date set)
	4	ACEROPIP	PIPE (pipe)
	5	ACEROEXR	EXR (Extended format required)
	6	ACEROEXC	EXC (Extended format preferred)
	7	ACEROBAS	BASIC (not extended or large format)
	8	ACEROLRG	LARGE (large format sequential data set)
580 (X'244')	257	ACEROJAC	Job account information
580 (X'244')	1	ACEROJNM	Number of fields in the account data
581 (X'245')	256	ACEROJFL	Job account data. Each field contains the length of the field followed by data.
837 (X'345')	7		Reserved
844 (X'34C')	257	ACEROSAC	Step account information
844 (X'34C')	1	ACEROSNM	Number of fields in the account data
845 (X'34D')	256	ACEROSFL	Step account data. Each field contains the length of the field followed by data.
1101 (X'44D')	7		Reserved
1108 (X'454')	2	ACERONVL	Number of volume serial numbers (up to 59)
1110 (X'456')	6	ACEROVOL	Array of volume serial numbers (up to 59)
1464 (X'5B8')	8		Reserved
1472 (X'5C0')	44	ACEROMEM	Member name
1516 (X'5EC')	44	ACEROCDS	Name of model data set for cluster
1560 (X'44')	44	ACEROIDS	Name of model data set for index
1604 (X'644')	44	ACERODDS	Name of model data set for data
1648 (X'670')	4	ACEROFNO	Tape file sequence number
1652 (X'674')	4	ACEROLBL	Tape label type. Possible values are:
	1	ACERONL	Unlabeled
	2	ACEROSL	IBM standard
	3	ACEROAL	ISO/ANSI standard
	4	ACERONSL	Non-standard
	5	ACEROSUL	IBM standard user
	6	ACEROAUL	ISO/ANSI standard user
	7	ACEROBLP	Bypass label processing
	8	ACEROLTM	Leading tape mark (unlabeled)

Table 32. Read-Only Variables (IGDACERO) (continued)

Offset	Length or Value	Name	Description
1656 (X'678')	8	ACEROTLB	Tape library name
1664 (X'680')	8		Reserved
1672 (X'688')	2	ACEROPOL	Length of pool name (ACEROPOV).
1674 (X'68A')	30	ACEROPOV	Tape pool name — allows your tape management system to pass a tape pool name to the ACS routine.
1704 (X'6A8')	44	ACEROMSD	Follows the rules for TSO data set names although each qualifier is a tape management destination.
1748 (X'6D4')	8	ACEROPCY	Single variable, ISPF naming convention
1756 (X'6DC')	1	ACEROMFN	Number of fields in ACEROMFL
1757 (X'6DD')	256	ACEROMFL	Multi-field variable in external user exit routine — repetitive fields of 1 byte length field in one byte hexadecimal followed by value of the field in character text. (for example: X'05'"value"X'02')
2013 (X'7DD')	15		Reserved

Read-Write Variables (IGDACERW)

Table 33 on page 150 shows the parameter list for read-write variables that is mapped by the IGDACERW macro.

Table 33. Read-Write Variables Parameter List (IGDACERW)

Offset	Length or Bit Pattern	Name	Description
0 (X'0')	1172	ACERW	Read-write variables parameter list
0 (X'0')	8	ACERWID	Control block ID contains the value "ACERW"
8 (X'8')	2	ACERWLEN	Length of control block
10 (X'A')		ACERWVER	Control block version
2	0	ACERWV	Version number (0)
12 (X'C')	4	ACERWNCS	Number of class selection return variables that follow (up to 15)
16 (X'10')	32	ACERWCSV	First class selection variable returned
16 (X'10')	2	ACERWVLN	Length of first value
18 (X'12')	30	ACERWVAL	Value of first returned variable
48 (X'30')	448		Up to 14 more class selection return variables
496 (X'1F0')	2		Reserved
498 (X'1F2')	2	ACERWNMG	Number of messages that follow (up to 6)
500 (X'1F4')	660	ACERWMSG	Area for up to 6 messages generated by execution of ACS routine write statements
500 (X'1F4')	110	ACERWTXT	Text of first message
1160 (X'488')	12		Reserved

Registers on 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

ACS Return and Reason Codes

Your ACS 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.

The ACS exit routine must end 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
Description

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 exit contains at least one error. You want it to be placed in disabled wait until the SMS address space is restarted.

Any other return code represents an error.

If the ACS routine is recalled by an ACS exit routine, the return code from the ACS routine needs to be propagated through the ACS exit if the allocation is to be failed. The exit routine needs to set the return code to X'04'.

Your ACS 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.

Example of the ACS Installation Exit Routine

The ACS installation exit routine in Figure 44 on page 152 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
         USING *,15
         B     PROLOG
         DC    AL1(17)              Length of string that follows
         DC    C'IGDACSSC &SYSDATE' For save area trace in dump
PROLOG   STM   14,12,12(13)         Standard entry linkage
         L     PARMLIST,0(R1)       Establish addressability to
         USING ACSPMD,PARMLIST       exit parameter list
         L     WORKBASE,ACSPWORK    Establish addressability to
         USING WORKAREA,WORKBASE     work area
         LA    11,SAVEAREA          Standard save area chaining
         ST    13,4(11)             Point new save area to caller's
         ST    11,8(13)             Point caller's save area to ours
         LR    13,11                Establish new save area

         LR    12,15                Load base register
         DROP  15                   Drop addressability
         USING IGDACSSC,12
         LA    TOADDR,STARTWK       Set up to clear the work area
         LA    TOLEN,CLEARLEN       Following the save area
         LA    FROMLEN,0            Set length of source (null)
         MVCL  TOADDR,FROMADDR      Clear work area
*
*  Set up access to read-only variables.
         L     ERO,ACSPAERO
         USING ACERO,ERO            Anchor read-only variables
*
*  Set up access to read-write variables.
         L     ERW,ACSPERW
         USING ACERW,ERW            Anchor read/write variables
         MVC   ACERWVAL(L'BLANK8),BLANK8
*
*  Invoke ACS routine.  Note that it may put messages in ACERWTXT.
         LA    SERVICEP,ACSPACSP    Parameter list for ACS routine
         ST    SERVICEP,ACSPARM     Use standard MVS linkage
         LA    R1,ACSPARM            conventions
         L     15,ACSPACS           Load address of ACS interface rtn
         BASR  14,15                Call STORCLAS selection routine
*
*   Return message indicating exit was entered.
```

Figure 44. Sample Storage Class ACS Installation Exit Routine Part 1 of 3

```
         LA    R1,MESSAGE1          Set address of message to write
         BAS   R14,WRITE            Call subroutine to return message
*
* If the ACS routine returned a null value, we need to do some resets.
         MVC   SCNAME(8),ACERWVAL   Prime area for message 2
         CLC   ACERWNCS,=F'0'       Is number of constructs = 0?
         BNE   CONTINUE             If ACS rtn set it, no change
         MVC   ACERWNCS,=F'1'       Set number of constructs = 1
         MVC   ACERWVLN,=H'0'       Set length of value
         MVC   SCNAME,NULLSC        Update area for message 2
*
* Build a message indicating a storage class was assigned.
CONTINUE EQU   *
         MVC   MSG2AOUT,MSG2A
         MVC   MSG2BOUT,MSG2B
         MVC   PAD,BLANK75
         LA    R1,MESSAGE2          Set address of message to write
         BAS   R14,WRITE            Call subroutine to return message
*
RETURN   EQU   *
         L     13,4(13)             Point to caller's save area
         LM    14,12,12(13)         Restore most of caller's registers
         LA    15,0                 Set return code
         BR    14                   Return to caller of exit routine
*
* Subroutine to add a message to those to be printed.  At entry to this
* routine R1 points to a 110-byte message.
WRITE    LH    WORKREG,ACERWNMG       Get number of existing messages
         LA    R0,1(,WORKREG)         Increment counter of messages
         CH    R0,=H'6'               Compare to max number of messages
         BHR   14                     Return if no more messages allowed
         STH   R0,ACERWNMG            Set new number of existing messages
         MH    WORKREG,=Y(L'ACERWTXT) Multiply number by length of each
         LA    WORKREG,ACERWTXT(WORKREG)  Point to appropriate msg area
         MVC   0(L'ACERWTXT,WORKREG),0(R1)  Move new message
         BR    14
*****************************************************************
* The documented size of the area containing the work area is 4096
* bytes.  If this routine were to store even one byte past its end,
* there probably would be a disaster.  The following is a trap to
* cause an assembly error if the length of the DSECT exceeds 4096.
* The value of the expression must not exceed 15.
*****************************************************************

         BR    ((WORKLEN-1)/4096)+15  Asm error if DSECT too long
*
```

Figure 45. Sample Storage Class ACS Installation Exit Routine Part 2 of 3

```
*****************************************************************
*                                   Start of constant 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
BLANK75  DC    CL75' '
BLANK8   EQU   BLANK75,8            For clearing message insert
*
* Map of dynamic storage used by this routine.  Supplied by caller.
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
*
CLEARLEN EQU   *-STARTWK            Length to clear
WORKLEN  EQU   *-WORKAREA           Amount of 4K area that we are using
* Register names.
R0       EQU   0                    Register 0
R1       EQU   1                    Register 1
WORKREG  EQU   2                    Work register
TOADDR   EQU   2                    Register for MVCL destination addr
TOLEN    EQU   3                    Register for MVCL destination 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 parameters for service
WORKBASE EQU   7                    Address of work area
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 46. Sample Storage Class ACS Installation Exit Routine Part 3 of 3

Chapter 6. DFSMShsm Installation Exits

You can use DFSMShsm installation exits to customize DFSMShsm processing.

The DFSMShsm installation exits fall into two categories: exits that support basic DFSMShsm functions and exits that support DFSMShsm ABARS functions. This topic describes only the exits that support the **basic** DFSMShsm functions. For information about the DFSMShsm ABARS installation exits, see Chapter 7, “DFSMShsm ABARS Installation Exits,” on page 203.

Note: If you use the REXX language to code installation exits, you should either code your REXX processing so that it is serially reusable or alternatively establish a separate environment control table for each task. This is because the REXX language has a 'single threaded appearance', which can cause problems if it is used in exits that are taken from multitasked functions.

Table 34 on page 155 lists the installation exits that support the basic DFSMShsm functions.

Table 34. DFSMShsm Installation Exits

Module Name	Description	When Available	See Topic
ARCADEXT	Data set deletion exit	During data set deletion or data set retirement. Not called for SMS-managed data sets.	“ARCADEXT: Data Set Deletion Installation Exit” on page 159
ARCBDEXT	Data set backup exit	During volume backup, when a data set fulfills the selection criteria. Also during command backup of individual data sets.	“ARCBDEXT: Data Set Backup Installation Exit” on page 162
ARCCBEXT	Control data set backup exit	After DFSMShsm creates backup copies of the control data sets.	“ARCCBEXT: Control Data Set Backup Installation Exit” on page 166
ARCCDEXT	Data set reblock exit	During recall or recovery processing	“ARCCDEXT: Data Set Reblock Installation Exit” on page 168
ARCINEXT	Initialization exit	After DFSMShsm startup, before functional subtasks become active.	“ARCINEXT: Initialization Installation Exit” on page 170
ARCMDEXT	Space management exit	When a data set fulfills the selection criteria for the level 0 volume being managed, but before the data set migrates or transitions	“ARCMDEXT: Space Management Exit” on page 172
ARCMMEXT	Second-level migration data set exit	When a command or automatic selection process selects to migrate an already migrated data set.	“ARCMMEXT: Second Level Migration Data Set Installation Exit” on page 179
ARCMVEXT	Space management volume exit	After DFSMShsm completes processing at level 0 volume during volume space management.	“ARCMVEXT: Space Management Volume Installation Exit” on page 181
ARCRDEXT	Recall exit	During recall of non-SMS-managed data sets. Not called for SMS-managed data sets.	“ARCRDEXT: Recall Installation Exit” on page 182

Table 34. DFSMShsm Installation Exits (continued)

Module Name	Description	When Available	See Topic
ARCRPEXT	Return-priority exit	Before a recall, delete, or recover (data set or volume) request is queued.	"ARCRPEXT: Return-Priority Installation Exit" on page 188
ARCSAEXT	Space management and backup exit	Once for each data set processed during volume space management or backup.	"ARCSAEXT: Space Management and Backup Installation Exit" on page 193
ARCSDEXT	Shutdown exit	After DFSMShsm has received a shutdown command, or is shutting itself down because of some unanticipated condition.	"ARCSDEXT: Shutdown Installation Exit" on page 196
ARCTDEXT	Tape data set exit	When an output tape data set is opened during backup, migration, recycle, or tape copy processing.	"ARCTDEXT: Tape Data Set Installation Exit" on page 197
ARCTEEXT	Tape-ejected exit	When an input tape is required for recall, data set or volume recovery, data set or volume restore, or volume recycle processing.	"ARCTEEXT: Tape-Ejected Installation Exit" on page 197
ARCTVEXT	Tape volume exit	When a DFSMShsm-owned tape no longer contains valid data, and therefore becomes empty. This exit also supports tapes used in ABARS processing. See "ARCTVEXT: Tape Volume Installation Exit" on page 213 for more information.	"ARCTVEXT: Tape Volume Installation Exit" on page 200

Using DFSMShsm Installation Exits

This topic contains information that is common to all installation exits supporting the basic DFSMShsm functions. See Chapter 7, "DFSMShsm ABARS Installation Exits," on page 203 for information specific to the installation exits supporting the DFSMShsm ABARS functions.

Installing DFSMShsm Exits

To use a DFSMShsm installation exit, you must link edit the exit into a load library, such as SYS1.LINKLIB. To request that DFSMShsm use the exit, issue the SETSYS EXITON command. If there are different versions of the exits that you want to use with different DFSMShsm hosts in an MVS image, you have at least the following two options:

- Define each startup procedure using a STEPLIB that consists of 1) a library that contains exits that are unique to that procedure, followed by 2) a library that contains DFSMSdss and other modules (including other exits) that are common to all hosts.
- Design each exit such that it determines the ASCB address for the address space in which it is running, and then scans the QCT's linked list of DFSMShsm host entries for the DFSMShsm host that is running under that ASCB. See "ARCINEXT: Initialization Installation Exit" on page 170.

Normally, you can activate DFSMShsm installation exits during DFSMShsm startup by including the SETSYS EXITON command in the DFSMShsm startup PARMLIB member. However, you can also activate

an inactive exit or dynamically refresh an active exit, even while DFSMShsm is running, by link-editing the desired version of the exit into a load library and then issuing the SETSYS EXITON command. The SETSYS EXITON command causes DFSMShsm to reload the exit using the standard search sequence.

The ARCINEXT, ARCSDEXT, and ARCRPEXT exits, whether link-edited in 24-bit or 31-bit mode, are each passed parameters that are in 31-bit storage (above the 16MB line).

See "Adding a New Exit" on page 3.

Replacing DFSMShsm Exits

The sequence for replacing each DFSMShsm exit presented in this topic is always the same:

1. Issue the SETSYS EXITOFF command to deactivate the active exit.
2. Link-edit the replacement exit code into the proper library in the LNKLST concatenation (job or step libraries).
3. Refresh the library lookaside (LLA).
4. Issue the SETSYS EXITON command to activate the replacement exit.

Writing DFSMShsm Exits

With DFSMShsm installation exits, you can tailor DFSMShsm functions and associate them with other space management activities. As you write exits, remember that they run as if they were DFSMShsm code. Therefore, it is recommended that you routinely incorporate the following considerations whenever you write installation exits:

- Keep supervisor services such as I/O to a minimum.
- Schedule for another time any processes that extend the elapsed time of automatic primary space management (for example, defragmentation of a DASD volume).
- Write all exits supporting the DFSMShsm basic functions as reentrant, with the exception of the two following exits. Writing exits as reentrant removes the rewrite requirement should the functions calling the exits become capable of concurrent processing by multiple tasks. You don't need to code the following exits as reentrant:
  - ARCINEXT (the initialization exit)
  - ARCSDEXT (the shutdown exit)
- Be aware that the recall exit is called in a system performing JES3 setup. The exit is called for migrated non-SMS-managed data sets during the JES3 converter-interpreter locate routine and is, therefore, in a critical system performance path.

Also always remember that DFSMShsm installation exits:

- Run enabled for interrupts.
- Run in problem program state.
- Run in either the DFSMShsm primary address space or the ABARS secondary address space (ARCTVEXT exit).
- Have pageable storage.
- Are protected by ESTAE.
- Are entered in the standard address space protection key of 8. Control must return in the same key as at entry.
- Execute in an authorized program facility (APF) authorized address space. As with all subroutines, do *not* link edit installation exits with APF authorization.

Special Considerations

In coding exit routines for DFSMShsm, you might sometimes find it useful to refer to a portion of a DFSMShsm data area. For descriptions and formats of general use DFSMShsm data areas, see *z/OS*

DFSMShsm Diagnosis. That book provides descriptions of fields in the data areas, the meaning and use of those fields, and the offsets and the byte and bit patterns for the fields. Data area cross-reference information follows the description of each record.

DFSMShsm uses subpools, which it shares within its subtasks, through the SHSPL parameter of the ATTACH macro. The subpools are 1, 6, 7, 10, 11, 12, 13, and 78. When you write exit routines or modify DFSMShsm and that code requires GETMAIN processing from one of these subpools, the FREEMAIN request must ask for the amount of storage requested by the GETMAIN and not a complete subpool FREEMAIN.

Registers on Entry to DFSMShsm Installation Exits

Before the exit is called, the contents of the registers are:

Register
Contents

0
Not applicable

1
Address of input parameters

2–12
Not applicable

13
Address of register save area

14
Caller's return address

15
Address of exit entry point

Registers on Return from DFSMShsm Installation Exits

Before the exit routine ends, the contents of the registers are:

Register
Contents

0
Not applicable

1–14
Restored to contents at entry

15
Not applicable

Calling DFSMShsm Installation Exits

The DFSMShsm installation exits must be accessible by the LOAD macro and must communicate with the standard MVS linkage for registers. “Registers on Entry to DFSMShsm Installation Exits” on page 158 and “Registers on Return from DFSMShsm Installation Exits” on page 158 describe standard MVS linkage with which DFSMShsm complies. The exits must save and restore registers. DFSMShsm does not place return codes in the registers; it includes them as parameters in the parameter list for exits that provide return codes. An installation exit cannot issue the DFSMShsm supervisor call instruction (SVC).

DFSMShsm calls each of its exits in the addressing mode in which they were link edited (either 24-bit mode or 31-bit mode), except for the ARCTVEXT exit, which must be in 24-bit mode.

Because DFSMShsm responds differently to abends that occur for different exits, the actions DFSMShsm takes are described for each exit. Generally, when an exit abends, DFSMShsm issues a HOLD on the

function, writes a message, and does not call the exit again until either a DFSMShsm RELEASE command releases the function or a SETSYS command requests the function.

The exit parameter list contains pointers to the information described in the discussion of each individual exit. It points to copies of the information unless specifically stated otherwise.

The exits are ESTAE protected. They are not subject to TRAP commands. If an exit abends and you require a dump for problem determination, see *z/OS DFSMShsm Diagnosis*.

Table 35 on page 159 lists the hex values for exits that do selective processing based on the device type DFSMShsm is processing.

Table 35. Hexadecimal Values for UCB Device Types

Unit Name	UCB Device Type Third and Fourth Bytes (Note 1)
3380	X'200E'
3390	X'200F'
3420	X'8003'
3423	X'8082'
3480	X'8080' (Note 2)
3480X	X'8080' (Note 2)
3490	X'8081'
3590-1	X'8083'
9345	X'2004'

Note:

1. The logical device names for some tape devices are different from their physical device names. 3480XF tape devices are identified with 3480X in the JCL; therefore, specify 3480X in the JCL whenever you identify 3480XF tape devices to DFSMShsm.
2. A tape device supports improved data recording capability whenever the 04 bit is on in the second byte of the four-byte UCB device-type code.

Creating User-Defined Messages

Message numbers ARC9000 through ARC9299 have been set aside for use by DFSMShsm installation exits, either as samples supplied by DFSMShsm or messages written by customers. For an explanation of these messages, customers have to locate the issuing exit or any user-created documentation for that exit.

Message numbers ARC9000 through ARC9199 are intended for use specifically for the ARCRPEXT exit. The ARCRPEXT installation exit can pass messages back to DFSMShsm, and DFSMShsm writes these messages in the migration or backup activity log. User messages issued by other DFSMShsm installation exits are generally write-to-operator (WTO) messages.

ARCADEXT: Data Set Deletion Installation Exit

You can use the data set deletion exit (ARCADEXT)to exclude data sets on level 0 volumes or on migration volumes from being processed during data set deletion (DBA-delete by age processing) or data set retirement (DBU-delete if backed up processing) for non-SMS-managed data sets.

Characteristics of the ARCADEXT Exit

The ARCADEXT installation exit receives control after DFSMShsm confirms that a data set has not been referred to in the number of days specified for the deletion. Checks for other criteria depend on whether the processing is for a level 0 volume or a migration volume.

- If DFSMShsm is retiring data sets on a level 0 volume (DBU processing), ARCADEXT is called *before* DFSMShsm determines that a current backup version exists. If DFSMShsm is deleting data sets on a level 0 volume (DBA processing), ARCADEXT is called regardless of the expiration date. Although the exit is called before all criteria are met, DFSMShsm does not scratch a data set unless all criteria are met.
- If DFSMShsm is retiring data sets on a migration volume (DBU processing), ARCADEXT is called *after* DFSMShsm confirms that the data set has either expired or has a backup version.

The ARCADEXT routine must be reentrant.

Recovering from an Abend of ARCADEXT Processing

If the data set deletion exit abends, DFSMShsm

- Does not delete the data set
- Stops processing the volume
- Holds migration
- Issues messages ARC0004I, ARC0535I, and ARC0734I with a return code of 54 and records them in the migration activity log

DFSMShsm does not disable the exit. Analyze the cause of the abend and determine if it is repeatable. If it is, issue the SETSYS EXITOFF(AD) command to disable the ARCADEXT exit before you release space management.

ARCADEXT Parameter List

Register 1 contains the address of the ARCADEXT parameter list as shown in Table 36 on page 161.

Table 36. ARCADEXT Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of the 140-byte VTOC entry of the data set about to be deleted or retired. If a non-VSAM data set is being deleted or retired from a level 0 volume, this data area contains the real VTOC entry of the data set. If a VSAM data set is being deleted or retired from a level 0 volume, this data area contains the real VTOC entry of the base data object with the base cluster name contained in the first 44 bytes. If a data set is being deleted or retired from a migration volume (DASD or tape), this data area contains a dummy VTOC entry of the data set containing only the following valid fields: Data set name (base cluster name if VSAM) Data set organization Data set indicators (DS1DSIND) Data set creation date Data set expiration date Data set last referred to date Data set block size (non-VSAM) Data set key length (non-VSAM) Data set record format (non-VSAM) The last-referred-to date field can be the same as the creation date if the data set did not have a last-referred-to date when it migrated. The last-referred-to date contains the last update date if a VSAM data set with a last update date is being retired.
04 (X'04')	4	The address of a 6-byte data area. If a data set is being deleted or retired from a level 0 volume, this data area contains the volume serial number of the level 0 volume. If a data set is being deleted or retired from a migration volume, this data area contains the volume serial number of the original level 0 volume that the data set was on before it migrated.
08 (X'08')	4	The address of a fullword binary area containing the approximate size of the data set in units of 1024 bytes. The size is the space available to be allocated for the user's data set. **Note:** Data set size is based on a 2KB blocking factor for any DASD. This field is only an estimate of size. If the data set is blocked with a factor other than 2048 bytes, available space can be larger or smaller.
12 (X'0C')	4	The address of a 4-byte data area with only the first 3 bits defined. The bits are: **0... ....** This data set is being retired. **1... ....** This data set is being deleted. **.0.. ....** The next bit does not contain valid information about the type of volume DFSMShsm is processing. **.1.. ....** The next bit contains valid information about the type of volume DFSMShsm is processing. **..0. ....** A level 0 volume is being processed. **..1. ....** A migration volume is being processed.
16 (X'10')	4	The address of a fullword binary return code.

ARCADEXT Return Codes

The following list contains the ARCADEXT return codes. You must put one of these codes, in binary, into the area pointed to at offset 16 of the parameter list.

Return Code
Description

00 (X'00')
Scratch this data set.

04 (X'04')
Do not scratch this data set.

ARCBDEXT: Data Set Backup Installation Exit

You can use the data set backup exit (ARCBDEXT) to perform the following tasks:

- To prevent DFSMShsm from backing up selected data sets whenever volume backup processes the level 0 volumes on which the data sets reside.
- To exclude non-SMS-managed data sets from backup as an alternative to using the ALTERDS command.

  This technique is effective for excluding large numbers of non-system-managed data sets from backup. For example, you can design the exit to make decisions based on data in the data set VTOC entry by selecting data sets based on part of the data set qualifier.
- To prevent compaction of a data set during volume backup to tape, DASD, or both, whenever you have previously specified one of the following commands:
  - SETSYS COMPACT(TAPEBACKUP)
  - SETSYS COMPACT(DASDBACKUP)
  - SETSYS COMPACT(ALL)
  - SETSYS ZCOMPRESS(TAPEBACKUP(YES))
  - SETSYS ZCOMPRESS(DASDBACKUP(YES))
  - SETSYS ZCOMPRESS(ALL)
- To direct DFSMShsm as to whether serialization should, or should not, be attempted before backing up the current data set, and whether a backup should be performed if serialization has been attempted but fails.
- To specify a RETAINDAYS value for the backup of a given data set

Characteristics of the ARCBDEXT Exit

The ARCBDEXT installation exit, called during volume backup processing, receives control after DFSMShsm determines that a data set should be backed up but before DFSMShsm backs it up. It also receives control during the backup of individual data sets through the BACKDS and HBACKDS commands and during the backup of migrated data sets.

A RETAINDAYS value may be returned by the exit for the data set. If a RETAINDAYS value was also specified on the BACKDS/HBACKDS command, then the value specified on the command will be used. If a value greater than 50000 (other than 99999) is returned by the exit, it will be interpreted as 50000. A decimal value of 99999 will indicate that the backup version should never expire. If a value less than zero is returned by the exit, it will be interpreted as zero.

The data set backup exit is called for both system- and non-system-managed data sets that are eligible for backup. A flag (DS1SMSFG) in the data set VTOC entry indicates whether the data set being processed is system-managed.

Note: Do not use the ARCBDEXT installation exit to override management class parameters for a data set. However, you can use it to change the compaction rules for system-managed data sets.

The ARCBDEXT routine must be reentrant.

Recovering from an Abend of ARCBDEXT Processing

If the data set backup exit abends, DFSMShsm

- Does not back up the data set
- Stops processing the volume
- Holds backup
- Issues messages ARC0734I and ARC0004I and records them in the backup activity log

DFSMShsm does not disable the exit. Analyze the cause of the abend to determine if it is repeatable. If it is, disable the ARCBDEXT exit by issuing the SETSYS EXITOFF(BD) command before releasing backup.

Systems constrained for virtual storage can abend. If your system is abending, consider calling ARCSAEXT, the space management and volume exit, to do some of the work done by the ARCBDEXT exit. You save space because the ARCSAEXT exit does not build work elements for data sets it eliminates from processing and because the exit is taken before DFSMShsm performs all data set eligibility tests.

ARCBDEXT Parameter List

Register 1 contains the address of the ARCBDEXT parameter list as shown in Table 37 on page 163.

Table 37. ARCBDEXT Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of the 140-byte VTOC entry of the data set about to be backed up. The first 44 bytes are the data set name. The contents of the format1 DSCB and how to code the IECSDSL1 mapping macro are described in *z/OS DFSMSdfp Advanced Services*. Exceptions to the data set VTOC entry include the following: • For system-managed VSAM data sets, the data set VTOC entry is that of the data object with the base cluster name contained in the first 44 bytes. • For non-system-managed VSAM data sets cataloged in an integrated category facility catalog, a dummy data set VTOC entry is provided containing the base cluster name, data set organization, the change bit (DS1DSCHA), the date the data set was last referred to, creation date, and expiration date. • For non-system-managed VSAM data sets not cataloged in an integrated catalog facility catalog, a dummy data set VTOC entry is provided containing the base cluster name and the data set organization. • For migrated data sets (being backed up), the data set VTOC entry that is provided contains only the following valid fields: – Data set name (base cluster name for VSAM data sets) – Data set organization – Data set indicators (DS1DSIND) – Data set creation date – Data set expiration date – Data set last-referred-to date – Data set record format (non-VSAM)
04 (X'04')	4	The address of a 6-byte data area containing the volume serial number of the volume being backed up. If DFSMShsm is backing up an individual data set, the data area contains the serial number of the volume the data set currently resides on. If DFSMShsm is backup up a migrated data set, the data area contains the serial number of the original level 0 volume on which the data set resided before it migrated.
08 (X'08')	4	The address of an area with the data set size and other environmental data, as show in Table 38 on page 164.
12 (X'C')	4	The address of a fullword binary return code.

ARCBDEXT—Input Data Structure

The third word of “ARCBDEXT Parameter List” on page 163 points to the address of the area described in Table 38 on page 164 and Table 39 on page 165.

Table 38. ARCBDEXT Input Data Structure -- Input Data Structure for Volume Backup Requests

Offset	Length or Bit Pattern	Description
00 (X'00')	4	Approximate size of the data set, in units of 1024 bytes.
04 (X'04')	8	Identifies the level of parameter list and data structure. If there is a value other than '*EXPAND1' at this offset, then this and subsequent fields are not part of the structure.
12 (X'0C')	1	Data set status: **1... ....** Data set is a backup-while-open (BWO) candidate. **.1.. ....** Reserved. **..1. ....** RETAINDAYS value set by exit. **..0. ....** RETAINDAYS value not set by exit. **0... ....** Data set is not a BWO candidate.
13 (X'0D')	1	Directions for backing up a data set in use (decimal value): 0 = Try to serialize and back up, but fail if data set is in use. This value is established by the command: SETSYS BACKUP - (INUSE(RETRY(N))). 1 (output only) = Try to serialize, but back up even if data set is in use. 2 (output only) = Back up without trying to serialize first. 10 = Try to serialize, but if data set is in use, schedule a retry for which serialization is REQUIRED. This value is established by the command: SETSYS BACKUP(INUSE(RETRY(Y) SERIALIZATION(REQUIRED))). 11 = Try to serialize, but if data set is in use, schedule a retry for which serialization is PREFERRED. This value is established by the command: SETSYS BACKUP(INUSE(RETRY(Y) SERIALIZATION(PREFERRED))). **Note:** Field INUSE serves both as input to and output from ARCBDEXT. The input is derived from SETSYS parameters, but the exit can change the field contents to other valid values. On return from the exit: • If the return code indicates "do not back up," field INUSE is ignored. • If INUSE does not contain one of the defined values, it is treated as 0. • For a BWO candidate, any value other than 0 or 10 is treated as 0.
14 (X'0E')	2	Length of management-class name in the following field. If this field is zero, there is no associated management class.
16 (X'10')	30	Management-class name associated with the data set.
46 (X'2E')	2	Reserved.
48 (X'30')	4	RETAINDAYS value.

Table 39. ARCBDEXT Input Data Structure (continued) -- Input Data Structure for Individual Data Set Backup Command Requests

Offset	Length or Bit Pattern	Description
04 (X'04')	8	Identifies the level of parameter list and data structure. If there is a value other than '*EXPAND2' at this offset, then this and subsequent fields are not part of the structure.
12 (X'0C')	2	Data set status: **1... ....** Data set is a backup-while-open (BWO) candidate. **..1. ....** RETAINDAYS value set by exit. **..0. ....** RETAINDAYS value not set by exit. **0... ....** Data set is not a BWO candidate. **.1.. ....** Backup is the result of data set backup command. **.0.. ....** Backup is not the result of data set backup command. Backup is the result of retry from volume request due to an INUSE failure.
14 (X'0E')	2	Reserved.
16 (X'10')	4	RETAINDAYS value.

Table 40. ARCBDEXT Input Data Structure (continued) -- Input Data Structure for Autobackup Stage 3

Offset	Length or Bit Pattern	Description
04 (X'04')	8	Identifies the level of parameter list and data structure. If there is a value other than '*EXPAND3' at this offset, then this and subsequent fields are not part of the structure.
12 (X'0C')	2	Data set status: **1... ....** Reserved. **.1.. ....** Reserved. **..1. ....** RETAINDAYS value set by exit. **..0. ....** RETAINDAYS value not set by exit.
14 (X'0E')	2	Reserved.
16 (X'10')	4	RETAINDAYS value.

ARCBDEXT Return Codes

The following list contains the ARCBDEXT return codes. You must put one of these codes, in binary, into the area pointed to at offset 12 of the parameter list.

Return Code
Description

00 (X'00')
DFSMShsm backs up and (optionally) compacts the data set. DFSMShsm sets the RETAINDAYS value for the given data set if specified by the exit.

04 (X'04')
For level 0 volumes, DFSMShsm backs up but does not compact the data set even if the data compaction option is active. This return code has no effect when DFSMShsm is backing up migrated volumes. DFSMShsm sets the RETAINDAYS value for the given data set if specified by the exit.

08 (X'08')
For level 0 volumes, DFSMShsm does not back up the data set but does turn off the update flag in the data set's VTOC entry. DFSMShsm does not set a RETAINDAYS value.

For migrated data sets, DFSMShsm does not back up the data set. DFSMShsm does not set a RETAINDAYS value.

12 (X'0C')
For level 0 volumes, DFSMShsm does not back up the data set and does not turn off the update flag in the data set's VTOC entry.

For migrated data sets, DFSMShsm does not back up the data set.

ARCCBEXT: Control Data Set Backup Installation Exit

You can use the control data set backup exit (ARCCBEXT) to perform the following tasks:

- Identify the data mover that backed up the control data sets and the journal.
- Merge all tape backup copies onto fewer tapes.
- Define RACF profiles to protect the tapes that contain the backup versions.
- Ensure control data set integrity by scheduling a job to examine the structural consistency of the control data sets. You can import the backed up data sets to temporary or renamed data sets, and then call the IDCAMS EXAMINE command to ensure that the control data sets are structurally consistent.
- Improve performance by scheduling a job to copy the control data sets and journal from DASD to tape. First, back up your control data sets and journal to DASD in parallel. Later, with this exit, copy your control data sets and journal from DASD to tape.
- Save the list of volume serial numbers that are associated with each backup data set. You can also get a list of volume serial numbers by calling catalog services (outside of the exit) because the backup data sets are cataloged by DFSMShsm when it finishes the backup.

Always perform the batch-type exit functions, such as copying data sets, within a batch environment. To run the exit functions in a batch environment, you must open an internal reader and submit JCL to the internal reader. You can then build the JCL with the data passed by the control data set backup function.

Characteristics of the ARCCBEXT Exit

The ARCCBEXT installation exit receives control after DFSMShsm creates backup copies of the control data sets.

The ARCCBEXT routine should be reentrant.

Recovering from an Abend of ARCCBEXT Processing

If the control data set backup exit abends, DFSMShsm

- Disables the exit
- Issues message ARC0502I and records it in the command activity log.

Analyze the exit to determine if the abend is repeatable. If it is, replace the broken exit with a version that has been fixed. Reenable the exit with the steps described in “Installing DFSMShsm Exits” on page 156.

ARCCBEXT Parameter List

Register 1 contains the address of the ARCCBEXT parameter list as shown in Table 41 on page 167.

Table 41. ARCCBEXT Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of a fullword binary area containing one of the following: **0** The exit was called during automatic backup. **4** The exit was called by the BACKVOL command.
04 (X'04')	4	The address of a 44-byte area containing the name of the journal backup data set(1)
08 (X'08')	4	The address of a list of volume serial numbers indicating which volumes the journal backup data set is on (1 and 2)
12 (X'0C')	4	The address of a 44-byte area containing the name of the migration control data set (MCDS) backup data set (1)
16 (X'10')	4	The address of a list of volume serial numbers indicating which volumes the MCDS backup data set is on (1 and 2)
20 (X'14')	4	The address of a 44-byte area containing the name of the backup control data set (BCDS) backup data set (1)
24 (X'18')	4	The address of a list of volume serial numbers indicating which volumes the BCDS backup data set is on (1 and 2)
28 (X'1C')	4	The address of a 44-byte area containing the name of the offline control data set (OCDS) backup data set (1)
32 (X'20')	4	The address of a list of volume serial numbers indicating which volumes the OCDS backup data set is on (1 and 2)
36 (X'24')	4	The address of a 4-byte area with only bits in the first byte defined. The bits are: **.0.. ....** DFSMShsm backed up the control data sets **.1.. ....** DFSMSdss backed up the control data sets. **..0. ....** The MCDS backup was successful. **..1. ....** The MCDS backup was unsuccessful. **...0 ....** The BCDS backup was successful. **...1 ....** The BCDS backup was unsuccessful. **.... 0...** The OCDS backup was successful. **.... 1...** The OCDS backup was unsuccessful. **.... .0..** The journal backup was successful. **.... .1..** The journal backup was unsuccessful. **.... ..0.** Neither the MCDS nor the BCDS has been split. **.... ..1.** Either, or both, the MCDS and BCDS have been split into a multicluster control data set.

Table 41. ARCCBEXT Parameter List (continued)

Offset	Length or Bit Pattern	Description
40 (X'28')	4	The address of a 44-byte area containing the name of the MCDS2 backup data set (1 and 3)
44 (X'2C')	4	The address of a list of volume serial numbers indicating which volumes the MCDS2 backup data set is on (1, 2, and 3)
48 (X'30')	4	The address of a 44-byte area containing the name of the MCDS3 backup data set (1 and 3)
52 (X'34')	4	The address of a list of volume serial numbers indicating which volumes the MCDS3 backup data set is on (1, 2, and 3)
56 (X'38')	4	The address of a 44-byte area containing the name of the MCDS4 backup data set (1 and 3)
60 (X'3C')	4	The address of a list of volume serial numbers indicating which volumes the MCDS4 backup data set is on (1, 2, and 3)
64 (X'40')	4	The address of a 44-byte area containing the name of the BCDS2 backup data set (1 and 3)
68 (X'44')	4	The address of a list of volume serial numbers indicating which volumes the BCDS2 backup data set is on (1, 2, and 3)
72 (X'48')	4	The address of a 44-byte area containing the name of the BCDS3 backup data set (1 and 3)
76 (X'4C')	4	The address of to a list of volume serial numbers indicating which volumes the BCDS3 backup data set is on (1, 2, and 3)
80 (X'50')	4	The address of a 44-byte area containing the name of the BCDS4 backup data set (1 and 3)
84 (X'54')	4	The address of a list of volume serial numbers indicating which volumes the BCDS4 backup data set is on (1, 2, and 3)

Note:

1. If the exit passes a value of 0 for the address of the backup data set name or the list of volumes, DFSMShsm did not back up that data set.
2. The area pointed to by the address of the list of volume serial numbers is:
   - A halfword binary area containing the number of volume serial numbers in the list.
   - A list of volume serial numbers and generic unit names. Each volume serial number is followed by the generic name of the tape unit on which the volume was written. Each volume serial number is 6 bytes long, and each unit name is 8 bytes long.
3. This parameter is passed only for multicluster control data sets.

ARCCDEXT: Data Set Reblock Installation Exit

The data set reblock exit (ARCCDEXT) can reblock recalled or recovered data sets as they move from a source DASD to a target DASD. You can use the ARCCDEXT exit to allow the system either to specify a block size for these data sets using a DFSMSdss mover or to specify the block size yourself using a DFSMShsm mover. You can only use the ARCCDEXT exit to respond with a yes or no answer to the question: "Should this data set be reblocked?" In most cases, system-determined block size can block data sets and there is no reason to reblock them. You can use the ARCCDEXT exit, however, to prevent exception, block-size dependent data sets from accidentally being reblocked. Reblocking can allow more data to be stored on each DASD track or cause better performance with any device.

Characteristics of the ARCCDEXT Exit

The ARCCDEXT installation exit is called during recall or recovery processing when all of the following are true:

- The recalled or recovered data set is sequential.
- The SETSYS CONVERSION(REBLOCKTOANY) command has been specified.
- The system reblockable indicator in the data set VTOC entry is off.

ARCCDEXT is called for both system- and non-system-managed data sets that are eligible for DFSMShsm-controlled reblocking. A flag (DS1SMSFG) in the data set VTOC entry indicates the type of data set being processed.

After the DFSMShsm data set reblock exit has been called, DFSMShsm checks the parameter list to determine if you requested reblocking. If DFSMSdss is the data mover for recall or recovery and DFSMShsm determines that you have requested reblocking, DFSMShsm tells DFSMSdss to reblock the data set subject to its ADRREBLK exit. The DFSMSdss ADRREBLK exit is preloaded with a default that allows the system to determine the blocksize. You can override the system-determined block size by specifying a different blocksize in the ADRREBLK exit. Although ADRREBLK can be coded to specify an explicit blocksize, it is recommended that you use the default implementation (system-determined block size for the optimal block size value).

Changing the ADRREBLK exit affects all jobs that invoke DFSMSdss restore.

For detailed information about the DFSMSdss ADRREBLK exit, see "Reblock Installation Exit Routine (ADRREBLK)" on page 237.

The ARCCDEXT routine must be reentrant.

Recovering from an Abend of ARCCDEXT Processing

If the data set reblock exit abends:

- No reblocking occurs
- DFSMShsm disables the exit
- DFSMShsm issues message ARC0004I and records it in the migration activity log (for recall) or in the backup activity log (for recovery)
- The recall does not end in error
- DFSMShsm does not hold any functions

Analyze the exit to determine if the abend is repeatable. If it is, replace the broken exit with a version that has been fixed. Reenable the exit with the steps described in "Installing DFSMShsm Exits" on page 156.

ARCCDEXT Parameter List

Register 1 contains the address of the ARCCDEXT parameter list as shown in Table 42 on page 169.

Table 42. ARCCDEXT Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of the 140-byte VTOC entry of the data set about to be reblocked. The first 44 bytes are the data set name.

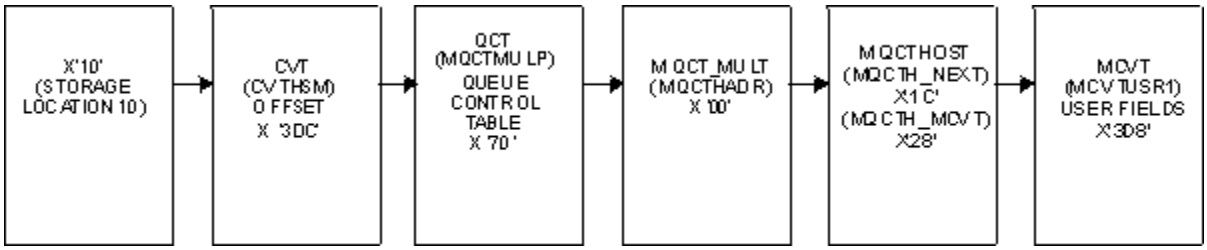
Table 42. ARCCDEXT Parameter List (continued)

Offset	Length or Bit Pattern	Description
04 (X'04')	4	The address of a 7-byte area. The first 6 bytes contain the volume serial number of the volume the data set was migrated from or backed up from. Byte 7 contains a data mover flag in bit 0 that indicates the data mover that was used. **0... ....** If bit 0 of byte 7 is set to 0, DFSMShsm has moved the data. **1... ....** If bit 0 of byte 7 is set to 1, DFSMSdss has moved the data.
08 (X'08')	4	The address of a fullword binary area containing the approximate size of the data set in units of 1024 bytes. The size is that of the space required to recall or recover the data set.
12 (X'0C')	4	The address of a fullword binary area containing one of three values that determine the block size DFSMShsm uses if the data set is reblocked. The DFSMSdss mover always passes a value of zero, which directs the system to calculate the block size for a data set. The DFSMShsm mover passes a value that offers greater track utilization on the target device (the largest acceptable block size nearest to 6144) or it passes the current block size. You can change this variable and use a different value. If you return a value greater than 32 760 or less than the record size, DFSMShsm uses the value passed to the exit. If DFSMSdss is the data mover, the value in this field is ignored.
16 (X'10')	4	The address of a fullword binary return code for the exit to set to zero to cause DFSMShsm to reblock the data set, or nonzero if DFSMShsm should not reblock the data set during recall or recovery.

ARCINEXT: Initialization Installation Exit

The initialization exit (ARCINEXT) can perform tasks of your choosing before returning to DFSMShsm during DFSMShsm startup time. For example, the exit can open and read a data set containing installation control records for building tables. The address of the tables can be stored in the installation-reserved fields of the MCVT control block. These tables can contain information required by other installation exits.

To find an MCVT, the exit uses the field MQCTHADR as a pointer to a linked list of elements, each of which (MQCTHOST) represents one DFSMShsm host in some address space with its own MCVT. (The last element in this list has binary zeros in its MQCTH_NEXT field.) To access the proper MCVT, subsequent installation exits need to search the chain starting at MQCTHADR for the element with MQCTH_ASCB that points to the current address space control block. MQCTH_MCVT in that element points to the proper MCVT with its user fields. Subsequent installation exits can refer to these user fields in the MCVT control block by the following method:



Note: The sample code ARCTPEXT that is found in SYS1.SAMPLIB(ARCTPEXT) scans the linked list of host elements for the proper MCVT pointer.

Attention: Do not attempt to invoke ARCINEXT exit to start DFSMS Optimizer, because Optimizer is already started (depending on the level of DFSMSopt).

Characteristics of the ARCINEXT Exit

The ARCINEXT installation exit receives control after the DFSMShsm startup PARMLIB member (ARCCMD*xx*) is processed, but before the functional subtasks become active.

The installation-reserved fields are the only fields in the MCVT control block that you are allowed to change. All other fields are reserved for IBM's use and are subject to change.

All parameters passed to ARCINEXT are located above the 16 MB line and have 31-bit addresses. If your ARCINEXT runs in AMODE 24, you can use the following example to switch to 31-bit mode before referring to 31-bit parameters from a 24-bit location:

```
MODE31  LA    15,*+10           GET 31-BIT TARGET ADDRESS
        O     15,=X'80000000'    SET AMODE 31 IN ADDRESS
        BSM   0,15              BRANCH INTO 31-BIT AMODE
```

Note: Before issuing the instructions in the preceding example, your program's base register should have a 31-bit address. If the program is running in 24-bit mode, byte 0 must contain zeros. Do not load the register with a BALR instruction while in 24-bit mode, as it sets inappropriate bits in byte 0. Instead, use LR or BASR instructions.

You can use the following example to switch back to 24-bit mode after referring to the 31-bit parameters.

```
*SWITCH TO 24 BIT ADDRESSING
MODE24   LA    15,*+6            GET TARGET ADDRESS
         BSM   0,15              ENTER 24 BIT MODE
```

This code is intended for use in RMODE 24. When used by RMODE 24 code, the AMODE can be either 24 or 31 before running either piece of switching code. The end result is the addressing mode indicated.

If ARCINEXT is link-edited in AMODE 31, then no change is needed.

Recovering from an Abend of ARCINEXT Processing

If the initialization exit abends:

- DFSMShsm issues message ARC0004I along with the abend code from the system diagnostic work area
- DFSMShsm continues processing

ARCINEXT Parameter List

Register 1 contains the address of the ARCINEXT parameter list as shown in Table 43 on page 171.

Table 43. ARCINEXT Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of the MCVT control block.
04 (X'04')	4	The address of a fullword binary return code field. This field is not used by DFSMShsm. There are no return codes for this exit.

MCVT User-Reserved Fields for Use with the ARCINEXT Exit

Table 44 on page 171 describes the MCVT user-reserved fields for use with the ARCINEXT installation exit.

Table 44. MCVT User-Reserved Fields. These areas are repositories for user-provided information to be used by other exits and programs. (for use with the ARCINEXT installation exit)

Offset	Length or Bit Pattern	Description
984 (X'3D8')	4	The address of a fullword binary area (MCVTUSR1) in the MCVT control block.

Table 44. MCVT User-Reserved Fields. These areas are repositories for user-provided information to be used by other exits and programs. (for use with the ARCINEXT installation exit) *(continued)*

Offset	Length or Bit Pattern	Description
988 (X'3DC')	4	The address of a fullword binary area (MCVTUSR2) in the MCVT control block.
992 (X'3E0')	4	The address of a fullword binary area (MCVTUSR3) in the MCVT control block.
996 (X'3E4')	4	The address of a fullword binary area (MCVTUSR4) in the MCVT control block.

ARCMDEXT: Space Management Exit

You can use the space management exit (ARCMDEXT) to perform the following tasks:

- Prevent selected data sets from migration or class transition from level 0 volumes during volume migration processing, even when the data sets fulfill the selection criteria. For example, with this exit you could prevent migration of a large number of non-system-managed data sets instead of having to issue SETMIG command for each of the data sets.

  For system-managed data sets whose management classes fulfill the migration selection criteria, you can use this exit to override the management class. Though this is not recommended, some users find that overriding the management class eliminates the introduction of many more management classes for these exception data sets.

  Table 45 on page 173 shows the rules governing migration versus class transition.
- Convert a class transition to a migration. If MGCBF_MDEXT flag is set ON, and ARCMDEXT exit returns RC=20 through RC=40 return codes, a class transition function is converted to the migration function. In this case, the data set is migrated according to the ARCMDEXT return code.

  Table 45 on page 173 shows the rules governing migration versus class transition.
- Override a data set condition called "in need of backup," which can prevent a data set from migrating to tape. A data set needs to be backed up if no current backup version exists and if the data set resides either on a non-system-managed volume or is a system-managed data set for which auto-backup is requested. Use caution, however, because if you use this exit to migrate the data set to tape, the data set will not be backed up by DFSMShsm.
- Prevent compaction of a data set during volume migration if you have previously specified one of the following:
  - SETSYS COMPACT(TAPEMIGRATE)
  - SETSYS COMPACT(DASDMIGRATE)
  - SETSYS COMPACT(ALL)
  - SETSYS ZCOMPRESS(TAPEMIGRATE(YES))
  - SETSYS ZCOMPRESS(DASDMIGRATE(YES))
  - SETSYS ZCOMPRESS(ALL)
- Override the target device type selection and route the data set to a tape migration level 2 volume. For system-managed data sets, it is easier to define a management class that migrates data sets from level 0 user volumes directly to level 2 migration volumes.
- Control which data sets are eligible for reconnection when you specify either the SETSYS TAPEMIGRATION(RECONNECT(ALL)) command or the SETSYS TAPEMIGRATION(RECONNECT(ML2DIRECTEDONLY)) command. These data sets remain eligible for normal migration when the ARCMDEXT exit does not allow reconnection.
- Indicate that the number of days that a data set is eligible for migration is passed in the input data structure. For system-managed data sets, eligibility age is based on the PRIMARY DAYS NON-USAGE management class attribute. For non-system-managed data sets, eligibility age is based on the number of days that are specified in the MIGRATE subparameter of either the ADDVOL PRIMARY command or the MIGRATE VOLUME command.

Note: Eligibility age fields may contain a negative value for both system-managed and non-system managed data sets that are processed for extent reduction and for system-managed data sets whose management class value is overridden by the DAYS(0) parameter of the MIGRATE VOLUME command. A negative value represents the number of days until the data set becomes eligible for normal migration.

- Delay certain checkpointed data sets (for MVS or IMS) from migrating from level 0 volumes until the need for restart has passed.

  If a system-managed, physical-sequential data set is open during a checkpoint and is moved after that checkpoint, the program cannot restart from that checkpoint. This is true even for data sets that were only being read. A flag (DS1CPOIT) in the data set entry of the VTOC, indicates if a checkpoint was taken while the data set was open.

  When an application program takes a checkpoint, the system records information about the status of that program in a checkpoint data set. This information includes the location on disk or tape where the application is currently reading or writing each open data set. If a data set that is open at the time of the checkpoint is moved to another location before the restart, you cannot restart the application from the checkpoint because the location-dependent information recorded by checkpoint/restart is no longer valid.

  There are several system functions (for example, DFSMShsm) that might automatically move a data set without the owner specifically requesting it. To ensure that all checkpointed data sets remain available for restart, the checkpoint function sets the unmovable attribute for each SMS-managed sequential data set that is open during the checkpoint. An exception is the data set containing the actual recorded checkpoint information (the checkpoint data set), which does not require the unmovable attribute.

  You can move checkpointed data sets when you no longer need them to perform a restart. DFSMShsm and DFSMSdss FORCECP(days) allow you to use operations such as migrate, copy, or defrag to move an SMS-managed sequential data set based on a number of days since the last access. DFSMShsm recall and DFSMSdss restore and copy are operations that turn off the unmovable attribute for the target data set.

- If the DS1CPOIT bit is on (set to 1), the ARCMDEXT exit allows migration only if an installation-determined number of days have passed since the data set was last referred to. You can modify the field name, MGC_CKPOINT_DAYS, by issuing the following patch command:

  ```
  PATCH .MGCB.+70 X'03' VERIFY(.MGCB.+70 X'05')  /* change to 3 */
  ```

  For more information about the checkpointed data set patch command, refer to *z/OS DFSMShsm Implementation and Customization Guide*.

Table 45. Rules for migration versus class transition, based on eligibility, MGCBF_MDEXT flag value, and return code from ARCMDEXT

Eligibility Status	RC=0 (Allow Migration)	RC=8 (Disable Migration)	RC=20 Through RC=40	RC=52 (Disable Migration and Class Transition)
Eligible for migration only	Migration	Data set processing fails with RC=45, reason 92	Migration	Data set processing fails with RC=45, reason 92
Eligible for migration and class transition	Migration	Class transition	Migration	Data set processing fails with RC=45, reason 92
Eligible for class transition only and MGCBF_MDEXT = OFF	Class transition	Class transition	Class transition	Data set processing fails with RC=45, reason 92

Table 45. Rules for migration versus class transition, based on eligibility, MGCBF_MDEXT flag value, and return code from ARCMDEXT (continued)

Eligibility Status	RC=0 (Allow Migration)	RC=8 (Disable Migration)	RC=20 Through RC=40	RC=52 (Disable Migration and Class Transition)
Eligible for class transition only and MGCBF_MDEXT = ON	Class transition	Class transition	Migration according to the return code	Data set processing fails with RC=45, reason 92

Characteristics of the ARCMDEXT Exit

The ARCMDEXT installation exit receives control whenever a data set fulfills the selection criteria for the level 0 volume being managed, but before the data set migrates or transitions. It is called when DFSMShsm processes a level 0 volume or an individual data set through any of the following:

- HMIGRATE command
- MIGRATE command
- Automatic primary space management
- Interval migration
- On-demand migration
- Class transitions

The input data structure provides flags that identify the type of volume migration function under which the exit was invoked.

This exit is called for both system- and non-system-managed data sets that are eligible for migration or extent reduction. A flag (DS1SMSFG) in the data set VTOC entry indicates whether the data set is system managed. For multivolume data sets, the exit is called for only the first volume. For system-managed data sets, the input data structure provides the associated management class.

An indicator in the ARCMDEXT parameter list indicates that this invocation is for a data set migrating by way of a data set command, not by way of a volume migration.

This exit is not called for data sets moving between migration volumes.

You must write ARCMDEXT to be reentrant.

Recovering from an Abend of ARCMDEXT Processing

If the space management exit abends, DFSMShsm:

- Does not migrate the data set
- Stops processing the volume
- Holds migration
- Issues messages ARC0004I, ARC0535I, and ARC0734I with a return code of 54 and records them in the migration activity log

DFSMShsm does not disable the exit. Analyze the cause of the abend to determine if it is repeatable. If it is, disable the ARCMDEXT exit before releasing space management.

ARCMDEXT Parameter List

Register 1 contains the address of the ARCMDEXT parameter list as shown in Table 46 on page 175.

Table 46. ARCMDEXT Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of a 140-byte VTOC entry of the data set about to be migrated. If a non-VSAM data set is being migrated, this 140-byte area contains the real data set VTOC entry. If a VSAM data set is being migrated, the 140-byte area contains the real VTOC entry of the base data object with the base cluster name contained in the first 44 bytes.
04 (X'04')	4	The address of 6-byte volume serial number of the level 0 volume being migrated.
08 (X'08')	4	The address of the input data structure that contains the data set size, management class name, input flags, including the reconnection eligibility flag.
12 (X'0C')	4	The address of a fullword binary return code.

ARCMDEXT—Input Data Structure

The third word of Table 46 on page 175 points to the address of the area that is described in Table 47 on page 175.

Table 47. ARCMDEXT Input Data Structure

Offset	Length or Bit Pattern	Description
00 (X'00')	4	Approximate allocated size of the data set, in units of 1024 bytes. For multivolume data sets, this is the size of the data set across all volumes on which it resides. The size calculation is based on a 2KB blocking factor for any given DASD. The calculation gives only an estimate of the size. If your data sets use a blocking factor other than 2048 bytes, the space allocated for your data set might be larger or smaller than is needed.
04 (X'04')	8	Identifies the level of parameter list and data structure. If there is a value other than 'EXPAND1' at this offset, then this and subsequent fields are not part of the structure.

Table 47. ARCMDEXT Input Data Structure (continued)

Offset	Length or Bit Pattern	Description
12 (X'0C')	2	First byte – input flags: **1... ....** Data set is a candidate for reconnection. **0... ....** Data set is not a candidate for reconnection. **.1... ....** Data set is being processed for extent reduction. **.0.. ....** Data set is not being processed for extent reduction. **..1. ....** Interval migration is in progress. **..0. ....** Interval migration is not in progress. **...1 ....** Primary space management is in progress. **...0 ....** Primary space management is not in progress. **.... 1...** The Days(0) parameter was specified on the MIGRATE command. **.... 0...** The Days(0) parameter was not specified on the MIGRATE command. **.... .1..** The CONVERT parameter was specified on the MIGRATE command. **.... .0..** The CONVERT parameter was not specified on the MIGRATE command. **.... ..1.** The migration eligibility age field at offset X'2E' in the input data structure is valid. **.... ..0.** The migration eligibility age field at offset X'2E' in the input data structure is not valid. **.... ...1** Migration is a result of a migrate data set command. **.... ...0** Migration is not a result of a migrate data set command.

Table 47. ARCMDEXT Input Data Structure (continued)

Offset	Length or Bit Pattern	Description
		Second byte – input flags: **1... ....** On-demand migration is in progress. **0... ....** On-demand migration is not in progress. **.1.. ....** DS is processed by a MIGRATE VOLUME command **.0.. ....** DS is not processed by a MIGRATE VOLUME command **..1. ....** DS is a Class Transition candidate **..0. ....** DS is not a Class Transition candidate **...1 ....** MGCBF_MDEXT flag is set ON **...0 ....** MGCBF_MDEXT flag is set OFF **.... 1...** Data set is targeted to migrate to cloud storage. **.... 0...** Data set is not migrating to cloud storage. **.xxx .xxx** Reserved
14 (X'0E')	2	Length of management class name in the following field. If this field is zero, there is no associated management class.
16 (X'10')	30	Management class name associated with the data set.
46 (X'2E')	2	Number of days since the data set became eligible for migration. This field can contain a negative value if DAYS(0) is specified on the MIGRATE volume command for an SMS-managed volume and for SMS and non-SMS data sets being processed for extent reduction. In both cases, the negative value represents the number of days until the data set is eligible for migration, under the normal criteria.
48 (X'30')	4	Cloud name definition pointer. Zero, if a data set is not migrated to Cloud.

Table 48. Cloud Name Definition

Offset	Length or Bit Pattern	Description
00 (X'00')	2	Cloud name length.
02 (X'02')	30	Cloud name.

ARCMDEXT Return Codes

The following list contains the ARCMDEXT return codes. You must put one of these codes, in binary, into the area pointed to at offset 12 of the parameter list.

Return Code	Description
00 (X'00')	DFSMShsm migrates this data set to the target device type that DFSMShsm has selected. DFSMShsm can also reconnect this data set if it meets all reconnection eligibility requirements and if one of the following is true: • DFSMShsm has selected a target device type of ML2 tape • DFSMShsm has selected a target device type of ML1 DASD and the SETSYS TAPEMIGRATION(RECONNECT(ALL)) command has been specified. DFSMShsm neither migrates the data set to tape nor reconnects it, if it needs to be backed up. If the data set migrates, it is compacted according to the DFSMShsm compaction option.
04 (X'04')	DFSMShsm migrates this data set to the target device type DFSMShsm has selected. For data sets migrating to tape, DFSMShsm does not migrate the data set if it needs to be backed up. If the data set migrates, DFSMShsm does not compact it.
08 (X'08')	DFSMShsm does not migrate this data set.
12 (X'0C')	DFSMShsm migrates this data set to the target device type DFSMShsm has selected. For data sets migrating to tape, DFSMShsm need not check to see that a data set has been backed up. If the data set migrates, it is compacted according to the DFSMShsm compaction option.
16 (X'10')	DFSMShsm migrates this data set to the target device type that DFSMShsm has selected. For data sets migrating to tape, DFSMShsm need not check to see that the data set has been backed up. If the data set migrates, DFSMShsm does not compact it.
20 (X'14')	DFSMShsm migrates this data set to tape regardless of the type of target device that DFSMShsm has selected. DFSMShsm does not migrate the data set, if the data set needs to be backed up. If the data set migrates, it is compacted according to the DFSMShsm compaction options.
24 (X'18')	DFSMShsm migrates this data set to tape regardless of the type of target device that DFSMShsm has selected. DFSMShsm does not migrate the data set, if the data set needs to be backed up. If the data set migrates, DFSMShsm does not compact it.
28 (X'1C')	DFSMShsm migrates this data set to tape regardless of the type of target device that DFSMShsm has selected. DFSMShsm need not check to see that the data set has been backed up. If the data set migrates, it is compacted according to DFSMShsm compaction options.
32 (X'20')	DFSMShsm migrates this data set to tape regardless of the type of target device that DFSMShsm has selected. DFSMShsm need not check to see that a data set has been backed up. If the data set migrates, DFSMShsm does not compact it.
36 (X'24')	DFSMShsm migrates this data set to DASD regardless of the type of target device that DFSMShsm has selected. DFSMShsm need not check to see that the data set has been backed up. If the data set migrates, it is compacted according to DFSMShsm compaction options.
40 (X'28')	DFSMShsm migrates this data set to DASD regardless of the type of target device that DFSMShsm has selected. DFSMShsm need not check to see that the data set has been backed up. If the data set migrates, DFSMShsm does not compact it.
44 (X'2C')	DFSMShsm does not attempt to reconnect this data set, but to migrate it to the target device that DFSMShsm has selected. For data sets migrating to tape, DFSMShsm does not migrate the data set if it needs to be backed up. If the data set migrates, it is compacted according to the DFSMShsm compaction options.

Return Code	Description
48 (X'30')	DFSMShsm attempts to reconnect this data set, even if DFSMShsm would otherwise migrate it to ML1 DASD. DFSMShsm does not reconnect the data set if it needs to be backed up. If the reconnection attempt fails, the data set remains eligible for normal migration, as if a return code of 00 had been specified.
52 (X'34')	DFSMShsm does not perform class transition (or migration) for this data set.

Note:

1. When the SETSYS TAPEMIGRATION(RECONNECT) command allows reconnection, use RC44 to exclude individual data sets from being reconnected.
2. When the SETSYS TAPEMIGRATION(RECONNECT(ML2DRECTEDONLY)) command generally restricts reconnection to those data sets eligible for direct migration to ML2 tape, use RC48 to allow individual data sets to be reconnected.
3. Nonzero return codes other than 48 disallow reconnection, even for those data sets that are identified as reconnectable.
4. If the data set is a class transition candidate only, and MGCBF_MDEXT flag is set ON, and return code is RC20 through RC40, the data set is processed as a migration candidate according to the return code. This is not applied to the MIGRATE DATASET command.

Table 49 on page 179 is a summary of return code actions.

Table 49. Summary of Return Codes for the Space Management Exit (ARCMDEXT). A checkmark indicates applicable actions set by the return codes.

Return Code	Migrate to DASD	Migrate to Tape	Migrate if in Need of Backup	Compactable
0 (X'00')	X	X		X
4 (X'04')	X	X		
8 (X'08')				
12 (X'0C')	X	X	X	X
16 (X'10')	X	X	X	
20 (X'14')		X		X
24 (X'18')		X		
28 (X'1C')		X	X	X
32 (X'20')		X	X	
36 (X'24')	X			X
40 (X'28')	X			
44 (X'2C')	X	X		X
48 (X'30')		X		X
52 (X'34')				

ARCMMEXT: Second Level Migration Data Set Installation Exit

The second-level migration data set exit (ARCMMEXT) provides additional control over the following types of data set migrations:

- Level 1 to level 1

- Level 1 to level 2
- Level 2 to level 2

Characteristics of the ARCMMEXT Exit

The ARCMMEXT installation exit receives control when a process, including command migration of a single data set, selects to migrate an already migrated data set.

This exit is called for both system- and non-system-managed data sets that are eligible to migrate from a migration volume. A flag in the MCD record (MCDSMSFG) is passed to the exit indicating whether the data set is system-managed. Contrast this exit with the ARCMDEXT, which is used when data sets are migrated from a level 0 (user) volume.

Parameters passed to ARCMMEXT can be located above or below the 16 MB line and have 31-bit or 24-bit addresses. In order to handle both cases, the ARCMMEXT should be link-edited with AMODE 31.

Recovering from an Abend of ARCMMEXT Processing

If the second-level migration exit abends, DFSMShsm

- Does not migrate the data set
- Stops its present migration processing
- Holds migration
- Issues message ARC0004I to the system console

DFSMShsm does not disable the exit. Analyze the cause of the abend and determine if it is repeatable. If it is, disable the ARCMMEXT exit by issuing the SETSYS EXITOFF(MM) command before releasing migration.

ARCMMEXT Parameter List

Register 1 contains the address of the ARCMMEXT parameter list as shown in Table 50 on page 180.

Table 50. ARCMMEXT Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of the address of a copy of the MCD data set record, which contains information such as whether the migration copy is an SDSP data set (MCDFSDP). For information about the MCD record, see *z/OS DFSMShsm Diagnosis*.
04 (X'04')	4	The address of a fullword binary area containing the number of days since the data set was last referred to on a level 0 volume.

Table 50. ARCMMEXT Parameter List (continued)

Offset	Length or Bit Pattern	Description
08 (X'08')	4	The address of a 4-byte area with only bits in the first byte defined. The bits are: **0... ....** Level 1 to level 2 migration is to DASD. **1... ....** Level 1 to level 2 migration is to tape. **.0.. ....** The target volume is a level 2 volume. **.1.. ....** The target volume is a level 1 volume. **..0. ....** The FREEVOL...AGE(0) command was not specified. **..1. ....** The FREEVOL...AGE(0) command was specified. **...0 ....** Migration is not a result of a migrate data set command. **...1 ....** Migration is a result of a migrate data set command.
12 (X'0C')	4	The address of a fullword binary area for the return code.

ARCMMEXT Return Codes

The following list contains the ARCMMEXT return codes. You must put one of these codes, in binary, into the area pointed to at offset 12 of the parameter list.

Return Code
Description

00 (X'00')
Migrate this data set.

04 (X'04')
Do not migrate this data set.

ARCMVEXT: Space Management Volume Installation Exit

You can use the space management volume exit (ARCMVEXT) to record volumes that need defragmentation or to schedule related activity, such as a DFSMSdss job.

Characteristics of the ARCMVEXT Exit

The ARCMVEXT installation exit receives control after DFSMShsm completes processing a level 0 volume during volume space management. It is called for system- and non-system-managed level 0 volumes, but is not called for level 1 or level 2 migration volumes.

Volume space management occurs:

- During the automatic primary space management window
- During interval migration
- When a MIGRATE VOLUME command is issued

Therefore, this exit can be called any time during a 24-hour processing day. If the processing defined by this exit must be confined to the automatic primary space management window, determine the time that you want to call the exit.

Remember that it is your responsibility to allocate and serialize the volumes passed to this exit.

You must write ARCMVEXT to be reentrant.

Recovering from an Abend of ARCMVEXT Processing

If the space management volume exit abends, DFSMShsm:

- Disables the exit
- Issues message ARC0502I and records it in the migration activity log

Analyze the exit to determine if the abend is repeatable. If it is, replace the broken exit routine with a version that has been fixed. Reenable the exit with the steps described in "Installing DFSMShsm Exits" on page 156.

ARCMVEXT Parameter List

Register 1 contains the address of the ARCMVEXT parameter list as shown in Table 51 on page 182.

Table 51. ARCMVEXT Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of a 6-byte volume serial number of the volume that DFSMShsm space management has just processed.
04 (X'04')	4	The address of a 4-byte UCBTYP field at offset X'10' in the UCB of the volume that DFSMShsm space management has just processed. This is the device type.
08 (X'08')	4	The address of a fullword binary fragmentation index of the volume that DFSMShsm has just processed. The value is calculated after data sets are migrated or deleted. The index value is a number between 0 and 1000 where 0 represents an unfragmented volume and 1000 represents a totally fragmented pack.

ARCRDEXT: Recall Installation Exit

You can use the recall exit (ARCRDEXT) to select a recall target volume from a pool of primary volumes you have identified, or you can use it to select the recall target volume from the general volume pool. You can also write the exit to select volumes in a different order from that of recall target volumes provided by the system. The system returns a maximum of five recall target volumes to the exit.

This exit is called only for non-system-managed data sets. In a common recall queue environment, this exit is invoked only by the processing host.

Characteristics of the ARCRDEXT Exit

The ARCRDEXT installation exit receives control in a JES3 environment at converter or interpreter time for jobs going through converter or interpreter setup. For other jobs in either a JES2 or JES3 environment, the ARCRDEXT exit receives control just before DFSMShsm recalls the data set. This exit is not used for recall to a specific volume.

When the ARCRDEXT exit receives control, it accesses a candidate list of volumes in the general pool or in a pool you designate. This list contains recall target volumes selected by the system. You can override the order of the candidate list of volumes and place your choices in the target volume choice list.

This exit is called only for non-system-managed data sets.

ARCRDEXT Parameter List

Register 1 contains the address of the ARCRDEXT parameter list as shown in Table 52 on page 183.

Table 52. ARCRDEXT Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of the MCD record. This is a copy of the MCD record that contains information about the data set being recalled. See *z/OS DFSMShsm Diagnosis* for the content of this record.
04 (X'04')	4	The address of a candidate list containing the number of volume serial numbers with like attributes or the eligible volumes in a user-defined pool.
08 (X'08')	4	The address of a fullword binary area containing the approximate allocated size of the data sets in units of 1024 bytes. The size is the amount of space required to recall the data set.
12 (X'0C')	4	The address of the choice list (see Figure 48 on page 187). The choice list is a set of numbers that represent the position of volume serials in the target volume candidate list to which DFSMShsm can recall data sets. The installation exit places the choice of volumes on the list. The format of the list is an array of fullword binary fields.

Building Recall Exit Lists

When selecting and prioritizing target volumes for recalled data sets, the recall exit builds three lists related to candidate volumes:

- Target volume candidate list
- Target volume choice list
- Target volume unlike attribute array list

Figure 47 on page 184 shows the recall exit parameter list and the three associated recall target volume lists.

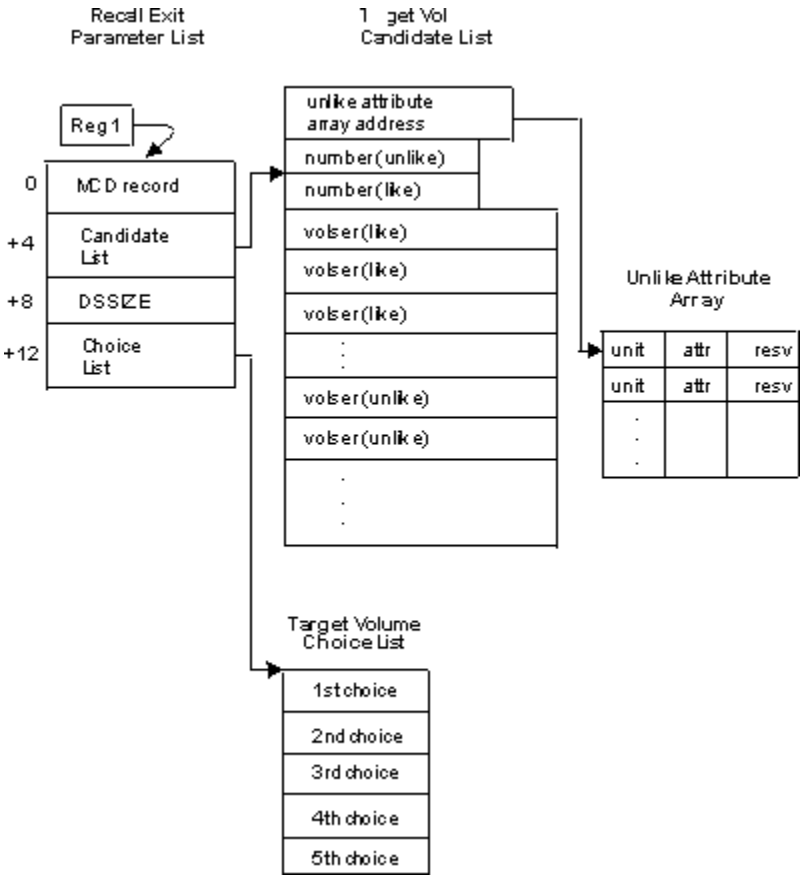


Figure 47. Recall Target Volume Data Structures

As described in Table 53 on page 184, the target volume candidate list has the following four components:

- Unlike attribute array address
- Number (unlike)
- Number (like)
- Volume serial (like) or volume serial (unlike)

Table 53. Target Volume Candidate List

Offset	Length or Bit Pattern	Description
-6	4	Unlike attribute array address The address of an array containing entries corresponding to candidate volumes with unlike DFSMShsm attributes
-2	2	Number (unlike) A halfword binary number indicating the number of volume serial numbers for unlike volumes in the candidate list
0	2	Number (like) A halfword binary number indicating the number of volume serial numbers for like volumes in the candidate list
2	6	Volume serial (like) A 6-byte candidate volume serial number.

Table 53. Target Volume Candidate List (continued)

Offset	Length or Bit Pattern	Description
8	6	Volume serial (like) A 6-byte candidate volume serial number.
14	6	Volume serial (like) An array of 6-byte candidate volume serial number.
20	6	Volume serial (like) Each entry is a 6-byte candidate volume serial number.
(See note)	6	Volume serial (unlike) A 6-byte candidate volume serial number.
(See note)	6	Volume serial (unlike) A 6-byte candidate volume serial number.
(See note)	6	Volume serial (unlike) Each entry is a 6-byte candidate volume serial number.

Note: The offset of the first unlike volume serial number is 2 + 6 × the number of like volume serials; each subsequent offset is 6 greater.

The contents of the target volume candidate list differs according to the way pools of target volumes are established and the subparameters of the SETSYS RECALL command. Details about volume pools can be found in the *z/OS DFSMShsm Storage Administration*. Data sets are recalled to one of three target volume categories:

- Volumes associated with a user-defined pool of volumes
- Volumes with like source and target volume attributes
- Volumes with unlike source and target volume attributes

ARCRDEXT Unlike Attribute Array

The unlike attribute array contains entries for each candidate volume in the candidate volume list with unlike attributes. Each entry indicates the attributes of the volume and the volume's unit type.

Table 54 on page 185 shows the fields and offsets for an unlike attribute array list entry.

Table 54. Unlike Attribute Array List Entry

Offset	Length or Bit Pattern	Description
00 (X'00')	4	A 4-byte area containing the UCB device type.

Table 54. Unlike Attribute Array List Entry (continued)

Offset	Length or Bit Pattern	Description
04 (X'04')	1	A 1-byte DFSMShsm volume attribute flag. The bits are: **1... ....** AUTOMIGRATION is specified on some processing units. **0... ....** NOAUTOMIGRATION is specified on some processing units. **.1.. ....** AUTOBACKUP is specified on some processing units. **.0.. ....** NOAUTOBACKUP is specified on all processing units. **..1. ....** BACKUPDEVICECATEGORY with TAPE or DASD is specified on the ADDVOL command. (The next bit indicates the keyword that was specified.) **..0. ....** BACKUPDEVICECATEGORY(NONE) is specified on the ADDVOL command. **...1 ....** BACKUPDEVICECATEGORY(TAPE) is specified on the ADDVOL command. **...0 ....** BACKUPDEVICECATEGORY(DASD) is specified on the ADDVOL command.
05 (X'05')	3	Reserved.

Recalling Data Sets to Target Volumes Associated with User-Defined Pools

If the data set being recalled is associated with a user-defined pool:

- The first two components (unlike attribute array address and the number of unlike volumes are not meaningful and are set to zero because there is no unlike attribute array.
- The third component (number of like volumes) gives the total number of volumes in the candidate list.
- The array of 6-byte volume serial numbers consists of eligible volumes in descending order of available free space. The volumes are not segregated by like or unlike attributes.
- The information found under "Recalling Data Sets to Target Volumes with Unlike Attributes" on page 186 and "Recalling Data Sets to Target Volumes with Like Attributes" on page 186 does not apply to user-defined pools.

Recalling Data Sets to Target Volumes with Unlike Attributes

If you specify SETSYS RECALL(ANYSTORAGEVOLUME(UNLIKE)) or SETSYS RECALL(PRIVATEVOLUME(UNLIKE)), the candidate list contains all eligible volumes with available space in the following order:

1. Volumes with like attributes sorted by the most available free space.
2. Volumes with unlike attributes sorted in the following order:
   a. Volumes with matching backup attributes and nonmatching space management attributes sorted by the most available free space.
   b. Volumes with nonmatching backup attributes sorted by the most available free space.

Recalling Data Sets to Target Volumes with Like Attributes

If you specify SETSYS RECALL(ANYSTORAGEVOLUME(LIKE)) or SETSYS RECALL(PRIVATEVOLUME(LIKE)), the candidate list contains only volumes with like attributes, sorted by the most available free space.

Overriding the System's Volume Priority

The recall target choice list is an array of five fullword binary fields into which the installation exit returns the choices for recall target volumes. You can write the exit to override the system's priority order for the target volumes by specifying the position of the desired volume in the recall target choice list.

Example: Specifying Choices for Recall Target Volumes:

The recall target choice list is numbered 1 through *n* where:

unlike + *like* = *n*

Indicate a maximum of five volumes in priority order that you want the recall to attempt. Figure 48 on page 187 is an example of a recall target choice list that could have been derived from the target volume candidate list in Figure 47 on page 184.

Example: Target Volume Choice List Results:

2 + 4 = 6

where:

2 is the number of volumes with unlike attributes

4 is the number of volumes with like attributes

6 is the sum of volumes with unlike and like attributes

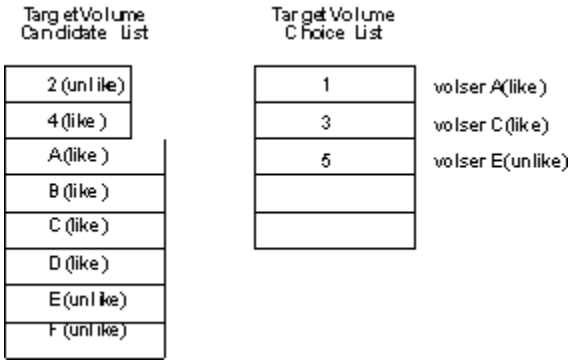


Figure 48. Recall Target Volume Choice Lists

A zero in the first choice position indicates that you have not made a selection and you want DFSMShsm to make its own selection. If some choices are valid, any choices that are not valid are ignored. A choice that is not valid is a number greater than the size of the candidate list or a negative number. Message ARC0316I is written to the migration activity log to indicate that a choice is not valid, and has been returned. If none of the five choices are valid, the recall fails.

Recovering from an Abend of ARCRDEXT Processing

If the recall exit abends, DFSMShsm

- Does not recall the data set
- Holds undirected recalls
- Does not hold directed recalls
- Issues message ARC0004I and records it in the migration activity log
- Stops returning volumes to JES3 for setup, and JES3 setups that refer to migrated data sets end in error

DFSMShsm does not disable the exit. Analyze the cause of the abend and determine if it is repeatable. If it is, disable the ARCRDEXT exit by issuing the SETSYS EXITOFF(RD) command before releasing recall and JES3 setup.

ARCRPEXT: Return-Priority Installation Exit

The return-priority exit (ARCRPEXT) is taken as each delete, recall, or recover request [in the form of a management work element (MWE)] is about to be queued on one of DFSMShsm's functional subtask queues for processing. Also, this exit can be used to prevent this processing. For a recall request, this exit is invoked by the host initiating the recall.

You can use the return-priority exit in the following ways:

- To set a priority (ranging from 0 to 100, where 100 is highest priority) for each request for which the user is waiting for results (WAIT type).
- To set a priority (ranging from 0 to 100) for each request for which the user is *not* waiting for results (NOWAIT type).

  DFSMShsm "primes" the priority field with either a default priority of 50 or the priority (for a data set recover request that is initiated from a volume recover request) that was assigned earlier to the initiating volume recover request. DFSMShsm then queues the nonpurged requests for work in priority order. A priority less than 0 is treated as 0; a priority greater than 100 is treated as 100. DFSMShsm queues two requests of the same type with the same priority in FIFO (first-in, first-out) order.

  For recall and delete requests, "priority order" on the recall queue means:

  – WAIT requests are queued before NOWAIT requests
  – Priorities of WAIT requests are relative to other WAIT requests
  – Priorities of NOWAIT requests are relative to other NOWAIT requests after WAIT requests:
    - NOWAIT requests (if any) with a relative priority of **50** are interleaved by the user ID of the submitter with other NOWAIT requests of priority 50
    - NOWAIT requests with a relative priority **greater than 50** are queued before those with a priority of 50, without interleaving
    - NOWAIT requests with a relative priority **less than 50** are queued after those with a priority of 50, without interleaving

  You can assign priority values other than 50 for those NOWAIT requests that you do not want interleaved by user ID.

  Note: *Interleave* means that the new MWE is placed on the queue so that if it is the *n*th request on the queue from that user ID, it follows all *n*th requests for any other user IDs, but precedes all *n+1* requests from any other user IDs. All *n*th requests are put on the queue in FIFO order.

  For **data set recover** requests, "priority order" on the data set recover queue means:

  – WAIT requests are queued before NOWAIT requests
  – Priorities of WAIT requests are relative to other WAIT requests
  – Priorities of NOWAIT requests are relative to other NOWAIT requests, *after* WAIT requests

  For **volume recover** requests, "priority order" on the volume recover queue means:

  – WAIT requests are queued before NOWAIT requests
  – Priorities of WAIT requests are relative to other WAIT requests
  – Priorities of NOWAIT requests are relative to other NOWAIT requests, *after* WAIT requests.
- To manage the EATTR parameter of the recalled data set. If the return code is set to RC4, the data set will be recalled with EATTR=OPT parameter. This does not guarantee that allocation will go to the Extended Addressing Space (EAS) of an Extended Address Volume (EAV), but having the EATTR=OPT does allow the data set to be EAS eligible.
- To direct DFSMShsm to enable extent-reduction recalls to a volume other than the original volume.

  – When MWEFEXT is set to 1, the recall is for extent reduction. Then you can use ARCRPEXT to set a bit in the output data structure. This bit indicates that the recall of the data set is permitted to select a level 0 volume other than the volume from which the data set has migrated.

  - When the data set is SMS managed, DFSMShsm uses the ACS routines for volume selection. For non-SMS managed data sets, DFSMShsm uses standard non-SMS volume selection.
- To purge a request. Note: there might be negative consequences from purging the recall requests. In some cases this can cause failures in applications or JCL that expect the data set to be recalled in order to complete an allocation request. Do not purge the request unless you fully understand how purging the recall request will affect the issuer of the request.
- To construct messages for a DFSMShsm activity log concerning the request.

The method of installing the exit means that you can activate and deactivate it dynamically. If the exit is not currently active or if it encounters an abend, DFSMShsm assigns a default priority of **50** to each recall, delete, and recover request being queued.

Characteristics of the ARCRPEXT Exit

The ARCRPEXT installation exit receives control after DFSMShsm has received a request to delete or to recall a data set, or a request to recover a data set or volume, but before the MWE for such a request has been queued for the appropriate functional subtask.

For a recall request from tape, DFSMShsm is normally able to flag that fact in the input data structure and flag that the device-type flag is valid. For a recall request from DASD, the MCD record is not read. Thus the device-type flag is not guaranteed to be valid, but the device type is nearly always DASD, even though the device type is flagged invalid (0).

For a recover request, if a data set or volume is being restored from a dump or a data set is being recovered, DFSMShsm is able to set the proper device-type flag in the input data structure, with an indication that the device type is valid. For an MWE to recover (not restore) a volume, the device-type flag cannot be considered valid.

This exit is called for both system- and non-system-managed data sets.

Note: Do not confuse this exit with the data set deletion exit ARCADEXT or the recall exit ARCRDEXT.

The ARCRPEXT routine must be reentrant.

Recovering from an Abend of ARCRPEXT Processing

If the return-priority exit abends when invoked for a *delete or recall* request, DFSMShsm:

- Assigns a priority of **50** to the request
- Does not set EATTR=OPT parameter for the recalled data set
- Queues the request for processing
- Issues message ARC0504I with the abend code and records it in the migration activity log
- Disables the exit for delete and recall requests, or, if the exit previously abended when invoked for a recover request, disables the exit fully

If the return-priority exit abends when invoked for a *recover* request, DFSMShsm:

- Assigns a priority of **50** to the request
- Queues the request for processing
- Issues message ARC0504I with the abend code and records it in the backup activity log
- Disables the exit for recover requests, or, if the exit previously abended when invoked for a delete or recall request, disables the exit fully

In either case, analyze the cause of the abend to determine if it is repeatable. If it is, replace the broken exit routine by following the procedure described in “Replacing DFSMShsm Exits” on page 157.

ARCRPEXT Parameter List

Register 1 contains the address of the ARCRPEXT parameter list as shown in Table 55 on page 190.

Table 55. ARCRPEXT Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of data describing the request and the environment of its issuer.
04 (X'04')	4	The address of the actual management work element (MWE) representing the request, not a copy. For information about the MWE, see *z/OS DFSMShsm Diagnosis*.
08 (X'08')	4	The address of an area to contain the results of the exit's feedback.

ARCRPEXT—Input Data Structure

The first word of "ARCRPEXT Parameter List" on page 189 points to the area described in Table 56 on page 190.

Table 56. ARCRPEXT Input Data Structure

Offset	Length or Bit Pattern	Description
00 (X'00')	8	An identifier ('RPEXT001') of the level of interface to ARCRPEXT.
08 (X'08')	2	Type of request in the MWE: **1** (Recall) **2** (Delete) **3** (Recover)
10 (X'0A')	00..	Request issued by the operator or generated by DFSMShsm.
	01..	Request issued from batch environment.
	10..	Request issued from interactive (TSO) environment.
	11..	Request issued from TMP (TSO background job) environment.
	..00	Reserved value.
	..01	Recovering a data set.
	..10	Restoring a volume from a dump (no incremental backups involved).
	..11	Recovering a volume from incremental backups (possibly involving a volume restore).
	.... 1...	Issuer is waiting for results.
	.... 0...	Nonwait request.
	.... .1..	Data coming from tape.
	.... .0..	Data coming from DASD.
	.... ..1.	Device-type flag (DASD or tape, above) is valid.
	.... ..0.	Device-type flag (DASD or tape, above) is not applicable.
	1	Data set recovery is initiated by volume command processing.
	0	Data set recovery is initiated by data set command processing.
11 (X'0B')	1	Reserved.
12 (X'0C')	8	User ID from the MWE. (An '*' in the first or second character identifies an internal user ID generated by DFSMShsm.)
20 (X'14')	44	Data set name, if the request is for a data set; otherwise, first byte is binary 0.

Table 56. ARCRPEXT Input Data Structure (continued)

Offset	Length or Bit Pattern	Description
64 (X'40')	6	Volume serial number, if the request is (1) to recover or restore a volume, or (2) a data set recover initiated by a command to recover this volume (bit 7 of offset 10 is 1). Otherwise, first byte is binary 0.

ARCRPEXT—Output Data Structure

The third word of "ARCRPEXT Parameter List" on page 189 points to the area described in Table 57 on page 191.

Table 57. ARCRPEXT Output Data Structure

Offset	Length or Bit Pattern	Description
00 (X'00')	4	Binary return code from exit ARCRPEXT. See "ARCRPEXT Return Codes" on page 193.
04 (X'04')	1	A flag byte containing the following bits: **0... ....** Use default priority for NOWAIT requests. **1... ....** Use priority at offset 06 for NOWAIT requests (if return code does not request a purge). **.0.. ....** Default - no change to DFSMShsm processing. **.1.. ....** Do not force extent-reduction recall to the original source volume. **..xx xxxx** Reserved.
05(X'05')	1	Reserved.
06 (X'06')	2	Desired binary priority for the request. DFSMShsm initializes this field with either a default value of 50 or the priority value (for a data set recover request initiated from a volume recover request) propagated from the initiating volume request.
08 (X'08')	4	Address of an area where your exit can construct messages for DFSMShsm to write to the pertinent activity log. See "ARCRPEXT Message Area" on page 191.

ARCRPEXT Message Area

Whether your exit has DFSMShsm continue with a request or purge it, the exit has the option of constructing up to five messages for DFSMShsm to write. Messages are written to the migration activity log for delete or recall requests and to the backup activity log for recover requests.

Table 58. ARCRPEXT Message Area

Offset	Length or Bit Pattern	Description
00 (X'00')	1	Number of messages (5) which can be built. (Set by DFSMShsm before invoking the exit.)
01 (X'01')	1	Number of characters in MESSAGE1 text following.
02 (X'02')	121	Area for text of MESSAGE1.
123 (X'7B')	1	Number of characters in MESSAGE2 text following.
124 (X'7C')	121	Area for text of MESSAGE2.

Table 58. ARCRPEXT Message Area (continued)

Offset	Length or Bit Pattern	Description
245 (X'F5')	1	Number of characters in MESSAGE3 text following.
246 (X'F6')	121	Area for text of MESSAGE3.
367 (X'16F')	1	Number of characters in MESSAGE4 text following.
368 (X'170')	121	Area for text of MESSAGE4.
489 (X'1E9')	1	Number of characters in MESSAGE5 text following.
490 (X'1EA')	121	Area for text of MESSAGE5.

Before passing control to the exit, DFSMShsm initializes each length field to binary 0 and each MESSAGE*n* field to blanks. When the exit returns control, DFSMShsm checks for messages, in the sequence MESSAGE1 through MESSAGE5. In each case, if the length field is nonzero, DFSMShsm treats the next byte as an ANSI printer control character and writes the following (maximum of 120) bytes as a message.

It is recommended that you use the following format for messages (although it is not enforced). The following message IDs are reserved for customer use:

- ARC90XX-Recall or delete request
- ARC91XX-Recover request

where "XX" is a two-digit number from 00 to 99.

Table 59. ARCRPEXT Suggested Message Format

Offset	Length or Bit Pattern	Description
00 (X'00')	1	ANSI-defined printer control characters (if not one of the following values, DFSMShsm substitutes a blank). Before printing the message: **Blank** = Space 1 line **0** = Space 2 lines **-** = Space 3 lines **1** = Skip to line 1 on new page
01 (X'01')	8	Time of day, in the form HH:MM:SS (unpacked from the contents of register zero returned by macro TIME DEC).
09 (X'09')	2	Two blank characters for spacing.
11 (X'0B')	8	Date, in the form YY/MM/DD.
19 (X'13')	2	Two blank characters for spacing.
21 (X'15')	7	Message identifier: ARC90XX or ARC91XX
28 (X'1C')	1	Message type: I
29 (X'1D')	1	Blank character for spacing.
30 (X'1E')	91	Message content.

ARCRPEXT Return Codes

The following are the ARCRPEXT return codes. You must put one of these codes, in binary, into the area pointed to at offset 00 of the output data structure found in Table 57 on page 191.

Return Code
Description

00 (X'00')
DFSMShsm queues the request, using the priority value (if a WAIT-type request, or if a NOWAIT-type request which indicates that a priority value is to be used) returned in the output data structure.

04 (X'04')
DFSMShsm recalls the data set with EATTR=OPT parameter.

08 (X'08')
DFSMShsm does not queue the request, but purges it without further processing.

Any other value
DFSMShsm treats the value like a return code of 00.

ARCSAEXT: Space Management and Backup Installation Exit

You can use the space management and backup installation exit (ARCSAEXT) instead of the ARCADEXT, ARCBDEXT, or ARCMDEXT exits when any of those exits is constrained for virtual storage. Space is saved because ARCSAEXT does not build work elements for data sets it eliminates and because the exit is taken before DFSMShsm performs any data set eligibility tests.

Characteristics of the ARCSAEXT Exit

The ARCSAEXT installation exit receives control once for each data set processed during volume space management or backup. As data sets are migrated and backed up, the VTOC entry for each data set that DFSMShsm processes is copied into a VTOC copy data set. The VTOC copy data set contains selected information from the DASD volume that DFSMShsm is backing up or migrating.

DFSMShsm calls this exit for system- and non-system-managed data sets. A flag in the data set VTOC entry (DS1SMSDS) is passed to the exit, indicating whether the data set is system managed. DFSMShsm provides different parameter lists for system- and non-system-managed data sets.

Note: This exit can be passed list and utility DSCBs, but the DSCBs are not placed in the VTOC copy data set because these data sets are not backed up or migrated. For more information about list and utility data sets, see *z/OS DFSMShsm Storage Administration*.

The ARCSAEXT exit is not called for any of the following data set VTOC entries:

- ICF catalogs
- CVOLs
- VTOCs
- VTOC indexes
- VVDSs

For integrated catalogs, the exit is called for backup, but not for space management.

The ARCSAEXT routine must be reentrant.

Recovering from an Abend of ARCSAEXT Processing

If the space management and backup exit abends, DFSMShsm:

- Does not process the data set
- Stops processing the volume
- Holds the function (backup or migration)

- Issues message ARC0004I to the system console

DFSMShsm does not disable the exit. Analyze the cause of the abnormal end to determine if it is repeatable. If it is, disable the ARCSAEXT exit by issuing the SETSYS EXITOFF(SA) command before releasing the held function (backup or migration).

ARCSAEXT Parameter List for Non-System-Managed Data Sets

Register 1 contains the address of the ARCSAEXT parameter list for non-system-managed data sets as shown in Table 60 on page 194.

Table 60. ARCSAEXT Parameter List for Non-System-Managed Data Sets

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of a 140-byte area containing the data set VTOC entry being processed.
04 (X'04')	4	The address of a 6-byte area containing the volume serial number being processed.
08 (X'08')	4	The address of a flag byte containing the following bits: **1... ....** The process is migration. **.1.. ....** The process is delete if backed up (DBU). **..1. ....** The process is delete by age (DBA). **...1 ....** The process is backup.
12 (X'0C')	4	The address of a fullword binary return code. See "ARCSAEXT Return Codes" on page 196.

ARCSAEXT Parameter List for System-Managed Data Sets

Register 1 contains the address of the ARCSAEXT parameter list for system-managed data sets as shown in Table 61 on page 194.

Table 61. ARCSAEXT Parameter List (System-Managed Data Sets)

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of a 140-byte area containing the data set VTOC entry being processed. If the data set VTOC entry cannot be found for an system-managed data set, the address of an area representing the data set VTOC entry is passed to the exit. The following fields are the only ones set in the pseudo data set VTOC entry: • Data set name (DS1DSNAM) • Format-1 identifier (DS1FMTID) • SMS indicator (DS1SMSDS) • VSAM indicator (DS1ORGAM)
04 (X'04')	4	The address of a 6-byte area containing the volume serial number being processed.
08 (X'08')	4	The address of a data set and processing information area, containing information described in "ARCSAEXT—Data Set and Processing Information Area" on page 195.

Table 61. ARCSAEXT Parameter List (System-Managed Data Sets) (continued)

Offset	Length or Bit Pattern	Description
12 (X'0C')	4	The address of a fullword binary return code field. See "ARCSAEXT Return Codes" on page 196.

ARCSAEXT—Data Set and Processing Information Area

The address at offset 8 of the parameter list points to the address of the 164-byte data set and processing information area in Table 62 on page 195. This data area applies only to system-managed data sets.

Table 62. ARCSAEXT Data Set and Processing Information Area

Offset	Length or Bit Pattern	Description
00 (X'00')	• 1...	• The process is migration.
	• .1..	• The process is delete if backed up (DBU).
	• ..1.	• The process is delete by age (DBA).
	• ...1	• The process is backup.
01 (X'01')	3	Reserved.
04 (X'04')	1 flag byte:	
	• 1...	• The data set does not have a data set VTOC entry.
	• .1..	• The data set does not have a VVR entry.
	• ..1.	• The data set has a duplicate VVR entry.
	• ...1	• This is the data component of a VSAM data set.
	• .... 1...	• This is the index component of a VSAM data set.
	• .... .1..	• This is an alternate index of a VSAM data set.
	• .... ..1.	• This is an alternate index with the upgrade attribute.
5 (X'05')	1 flag byte:	
	• 1...	• This is a VSAM key-sequenced data set.
	• .1..	• This is a VSAM entry-sequenced data set.
	• ..1.	• This is a VSAM relative-record data set.
	• ...1	• This is a VSAM linear data set.
	• .... 1...	• This is a generation data set.
	• .... .1..	• This is a VSAM key-range data set.
	• .... ..1.	• This is the first segment of a VSAM key-range data set.
	• 1	• This is a temporary data set.
6 (X'06')	2	Reserved.
8 (X'08')	• 1 • H • M • N	Status of a generation data set (character). Active generation data set. Rolled-off generation data set. Deferred generation data set.
9 (X'09')	1	Reserved.
10 (X'0A')	2	Length of the base cluster name.
12 (X'0C')	44	Base cluster name.
56 (X'38')	2	Length of the management class name.

Table 62. ARCSAEXT Data Set and Processing Information Area (continued)

Offset	Length or Bit Pattern	Description
58 (X'3A')	30	Management class name.
88 (X'58')	2	Length of the storage class name.
90 (X'5A')	30	Storage class name.
120 (X'78')	2	Length of the data class name.
122 (X'7A')	30	Data class name.
152 (X'98')	8	Last backup date (microsecond clock format).

ARCSAEXT Return Codes

The following are the ARCSAEXT return codes. You must put one of these codes, in binary, into the area pointed to at offset 12 on the parameter list.

Return Code
Description

00 (X'00')
DFSMShsm processes this data set for space management or backup.

04 (X'04')
DFSMShsm does not process this data set for space management or for backup. For backup, DFSMShsm includes this data set in the VTOC copy data set. For more information about VTOC copy data sets, see *z/OS DFSMShsm Storage Administration*.

08 (X'08')
DFSMShsm does not back up this data set. DFSMShsm does not include this data set in the VTOC copy data set during normal processing. This return code is not defined for space management processing.

ARCSDEXT: Shutdown Installation Exit

The shutdown exit (ARCSDEXT) can perform the tasks of your choosing whenever DFSMShsm is shutting down. Typically, such an exit would be used in tandem with the initialization exit (ARCINEXT). Data accumulated in tables obtained by ARCINEXT can be retrieved and processed and the table areas freed.

Characteristics of the ARCSDEXT Exit

The ARCSDEXT installation exit receives control just before DFSMShsm terminates its functional subtasks.

As with the initialization exit, the installation-reserved fields are the only fields in the MCVT record that you are allowed to change. All other fields are reserved for IBM's use and are subject to change.

All parameters passed to ARCSDEXT are located above the 16 MB line and have 31-bit addresses. You should link-edit ARCSDEXT in AMODE 31.

Recovering from an Abend of ARCSDEXT Processing

If the shutdown exit abends:

- DFSMShsm issues message ARC0004I along with the abend code from the system diagnostic work area (SDWA).
- DFSMShsm continues shutting down

ARCSDEXT Parameter List

Register 1 contains the address of the ARCSDEXT parameter list as shown in Table 63 on page 197.

Table 63. ARCSDEXT Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The pointer to the ADDRESS of the 140-byte area containing the data set VTOC entry being processed.
04 (X'04')	4	The address of a fullword binary return-code field. DFSMShsm does not use this field.

MCVT User-Reserved Fields (for use with the ARCINEXT and ARCSDEXT Exits)

Table 44 on page 171 illustrates the MCVT user-reserved fields for use with the ARCINEXT and ARCSDEXT installation exits.

ARCTDEXT: Tape Data Set Installation Exit

You can use the tape data set exit (ARCTDEXT) to change the expiration date (JFCBXPDT field) on backup or migration tapes to a date other than 99365.

Characteristics of the ARCTDEXT Exit

The ARCTDEXT installation exit receives control when an output tape data set is opened during backup, migration, recycle, or tape copy processing, and the tape security option is either expiration date (EXPIRATION) or expiration date including password-protected data sets (EXPIRATIONINCLUDE).

You must write ARCTDEXT to be reentrant.

Recovering from an Abend of ARCTDEXT Processing

If the tape data set exit abends, DFSMShsm:

- Stops processing the data set
- Stops processing the volume
- Holds the function
- Issues message ARC0004I

DFSMShsm does not disable the exit. Analyze the cause of the abend to determine if it is repeatable. If it is, disable the ARCTDEXT exit by issuing the SETSYS EXITOFF(TD) command before releasing space management.

ARCTDEXT Parameter List

Register 1 contains the address of the ARCTDEXT parameter list as shown in Table 64 on page 197.

Table 64. ARCTDEXT Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of a 176-byte copy of the job file control block to be sent to the open routine.

ARCTEEXT: Tape-Ejected Installation Exit

You can use the tape-ejected exit (ARCTEEXT) to dynamically insert nonlibrary DFSMShsm tapes in a library, as needed. This allows installations to run with a smaller tape library and have the choice of allocating tapes outside the library or dynamically re-inserting tapes that were previously ejected from the library. These actions can be done without causing jobs to fail or adversely affecting other DFSMShsm

activity. This exit gains control at a point before code that supports the TAPEINPUTPROMPT function is processed.

ARCTEEXT is called if DFSMShsm determines that not all of the required tapes are in a single Automated Tape Library (ATL) or Manual Tape Library (MTL), or when one or more tapes is outside of a library. This gives the installation the opportunity to insert tapes into a library or tell DFSMShsm to allocate the tape drives that are outside of a library or to fail the allocation request.

Characteristics of the ARCTEEXT Exit

The ARCTEEXT installation exit receives control after DFSMShsm confirms that the tapes needed during the following DFSMShsm functions are not in a library:

- **Recalling** data sets from migration level 2 tape volumes
- **Recovering** data sets or volumes from backup tape volumes
- **Restoring** data sets or volumes from dump tape volumes
- **Recycling** migration level 2 or backup tape volumes

The ARCTEEXT routine must be reentrant and can be either 24-bit mode or 31-bit mode.

Recovering from an Abend of ARCTEEXT Processing

If the tape-ejected exit abends, DFSMShsm:

- Issues message ARC0502I
- Disables the exit
- Continues processing as if the exit were set off

Analyze the cause of the abend to determine if the exit is likely to abend again. If it is, replace the broken exit routine with a version that has been fixed, before you invoke the exit again. Reenable the exit with the steps described in “Installing DFSMShsm Exits” on page 156.

User-Created Messages

Message numbers ARC9000 through ARC9299 have been set aside for use by DFSMShsm installation exits, either as samples supplied by DFSMShsm or messages written by customers. See “Creating User-Defined Messages” on page 159.

ARCTEEXT Parameter List

Register 1 contains the address of the ARCTEEXT parameter list as shown in Table 65 on page 199.

Table 65. ARCTEEXT Parameter List

Offset	Length or Bit Pattern	Description
00(X'00')	4	The address of a 4-byte data area containing flags that indicate the function being processed and the type of tapes needed. Byte 0 of the 4-byte data area contains the following bits: **1... ....** Data set recovery function. **.1.. ....** Volume recovery function. **..1. ....** Data set recall function. **...1 ....** Volume recycle function. **.... 1...** Data set restore function. **.... .1..** Volume restore function. Byte 1 of the 4-byte data area contains the following bits: **1... ....** Migration level 2 tape volume. **.1.. ....** Backup tape volume. **..1. ....** Dump tape volume.
04(X'04')	4	The address of a 2-byte count field followed by 6-byte volume serials needed to process this function. The count field contains the number of volumes needed.
08(X'08')	4	The address of a fullword binary return code described in “ARCTEEXT Return Codes” on page 199.

ARCTEEXT Return Codes

The following list contains the ARCTEEXT return codes. You must put one of these codes, in binary, into the area pointed to at offset 08 on the parameter list.

Return Code
Description

00 (X'00')
The tape volumes do not remain in a library and are not intended to be in a library.

DFSMShsm again validates that the needed tapes are not located in a library, or that the needed tapes are all located within a single library. If the tapes are not in a library, DFSMShsm allocates a tape drive that is not associated with a library. If the tapes are all in a single library, DFSMShsm allocates a tape drive associated with that library.

This exit will not be reinvoked for this tape drive allocation.

04 (X'04')
The tape volumes have now been moved by the customer into a library.

DFSMShsm again validates that all needed tapes are located within a single library. If the tapes are not all in a single library, DFSMShsm reinvokes this exit until they all are, or until the return code from this exit is something other than 04 (X'04').

08 (X'08')
The request fails without an allocation.

ARCTVEXT: Tape Volume Installation Exit

The tape volume exit (ARCTVEXT) lets a tape management system know that DFSMShsm is releasing ownership of a DFSMShsm tape.

Attention: DFSMS/MVS DFSMShsm Version 1 Release 4 and subsequent releases no longer use the ARCTVEXT exit to communicate with DFSMSrmm. Invoke the ARCTVEXT exit only if you are using another tape management product.

Characteristics of the ARCTVEXT Exit

The ARCTVEXT installation exit receives control during the following situations:

- DFSMShsm tape deletion processing.

  The ARCTVEXT exit receives control either when all valid data is removed from a tape, or when a RECYCLE FORCE command is issued. All valid data is removed after a backup or migration tape is recycled, after a dump tape has its contents deleted, or when the DELVOL PURGE command is used. This exit is called for each tape, regardless of whether the SCRATCHTAPE or HSMTAPE option is specified for the SETSYS TAPEDELETION command. If you jointly manage your tapes with a tape management system other than DFSMSrmm, you can write the exit routine to inform the tape management system that the tape no longer contains valid DFSMShsm data.(Users of DFSMSrmm do not need this exit because the equivalent DFSMShsm/DFSMSrmm interface is always invoked.)

  If an alternate tape exists for a tape that is being deleted, both the alternate tape and the original tape are deleted by a single call to delete the original tape.
- Backup of the control data sets and journal if the SETSYS CDSVERSIONBACKUP command has been specified.

  This exit is called after DFSMShsm uncatalogs the oldest backup version. The exit is called separately for each tape associated with the version being rolled off.
- Tape replace processing when an original tape is replaced by an alternate tape. The ARCTVEXT exit releases the original tape.
- Tape copy processing if a failure occurs after the alternate tape is mounted.
- EXPIREBV command processing when the EXPIREBV command is issued for ABARSVERSIONS.

  This exit is called after DFSMShsm successfully deletes each ABACKUP output file associated with the ABARS version being expired. The exit is called separately for each tape volume associated with the files being deleted as part of the expiration process. The exit is called without alternate volumes.
- ABACKUP or ARECOVER roll-off processing.

  This exit is called before DFSMShsm backs up or recovers an aggregate if DFSMShsm determines that one or more of the oldest ABARS version records should be rolled off. The exit is called separately for each tape associated with the files belonging to the version being rolled off.
- ABACKUP processing when a failure occurs.

  During ABACKUP cleanup processing, this exit is called separately for each tape associated with each ABACKUP output file being deleted during the cleanup process.

The tape volume exit does not receive control when tapes are being marked unavailable by DFSMShsm.

You must write ARCTVEXT to be reentrant.

Recovering from an Abend of ARCTVEXT Processing

If the tape volume exit abends, DFSMShsm:

- Disables the exit.
- Issues message ARC0502I and records it in the log for the function DFSMShsm has been performing.

If the tape volume exit abends during ABACKUP cleanup processing, DFSMShsm issues message ARC6187E and holds ABACKUP processing after cleanup has been completed.

Analyze the exit to determine if the abend is repeatable. If it is, replace the broken exit with a version that has been fixed. Reenable the exit with the steps described in "Installing DFSMShsm Exits" on page 156.

ARCTVEXT Parameter List

Table 66 on page 201 shows that register 1 contains the address of the ARCTVEXT parameter list.

Note: If there is an alternate volume, there are three addresses in the parameter list; if there is no alternate volume, there are only two addresses in the parameter list. The high order bit of the second address distinguishes between these two options.

Table 66. ARCTVEXT Parameter List

Offset	Length or Bit Pattern	Description
0 (X'00')	4	The address of an 8-byte area (see Table 67 on page 201), where the first 6 bytes contain the volume serial number of the volume having no valid data, followed by a 2-byte field.
4 (X'04')	4	The address of a fullword binary return code described in "ARCTVEXT Return Codes" on page 202. If the high order bit is 1, then there is no alternate volume and the parameter list is 8 bytes long. If the high order bit is 0 and the address at offset 8 is nonzero, then there is an alternate volume and the parameter list is 12 bytes long.
8 (X'08')	4	The address of an 8-byte area (see Table 67 on page 201), where the first 6 bytes contain the volume serial number of the alternate volume that has no valid data, followed by a 2-byte field.

ARCTVEXT Data Area

The parameter list points to the address of the 8-byte ARCTVEXT data area as shown in Table 67 on page 201.

Table 67. ARCTVEXT Data Area

Offset	Length or Bit Pattern	Description
00 (X'00')	6	The volume serial number of the tape with no valid data.
06 (X'06')	1	A flag field containing the following: **1... ....** The tape is being purged. This pattern is always set for alternate volumes. **0... ....** The tape is being unassigned or reassigned. **.1.. ....** The data sets on this tape have been expiration-date protected by DFSMShsm. Tapes with expiration-date-protected data sets might need to be reinitialized when they are purged from DFSMShsm control. **..1. ....** The data sets on this tape have been password protected by DFSMShsm. Tapes with password-protected data sets might need to be reinitialized when they are purged from DFSMShsm control.

Table 67. ARCTVEXT Data Area (continued)

Offset	Length or Bit Pattern	Description
07 (X'07')	1	A flag byte containing the following: **00.. ....** Reserved. **01.. ....** The SETSYS TAPEDELETION option has not been specified. This pattern is not set for alternate volumes. **10.. ....** The SETSYS TAPEDELETION(SCRATCHTAPE) option has been specified or is the default for this category of DFSMShsm tape. **11.. ....** The SETSYS TAPEDELETION(HSMTAPE) option has been specified or is the default for this category of DFSMShsm tape. **..1. ....** This is a TAPECOPY request to delete only the duplex alternate tape. **...x ....** Reserved. **.... 0000** Reserved. **.... 0001** This is a migration tape or migration alternate tape. **.... 0010** This is a backup tape or backup alternate tape. **.... 0011** This is a dump tape. This pattern is not set for alternate tapes. **.... 0100** This is a CDS backup tape. This pattern is not set for alternate tapes. **.... 0101** This is an ABARS output tape. This pattern is not set for alternate tapes.

ARCTVEXT Return Codes

The following list contains the ARCTVEXT return codes. You must put one of these codes, in binary, into the area pointed to at offset 08 of the parameter list.

Return Code
Description

00 (X'00')
ARCTVEXT exit processed normally.

04 (X'04')
ARCTVEXT exit processed abnormally. The exit is not called again until the problem is resolved.

If the exit is called during ABACKUP failure processing and the exit returns a code of 4, the exit is disabled for the currently active ABARS secondary address space.

Chapter 7. DFSMShsm ABARS Installation Exits

You can use DFSMShsm ABARS installation exits to customize DFSMShsm ABARS processing.

The DFSMShsm installation exits fall into two categories: exits that support basic DFSMShsm functions and exits that support DFSMShsm ABARS functions. This topic describes only the exits that support the DFSMShsm ABARS functions. For information about the DFSMShsm installation exits that support basic DFSMShsm functions, see Chapter 6, "DFSMShsm Installation Exits," on page 155.

Table 68 on page 203 lists the DFSMShsm ABARS exits.

Table 68. ABARS Installation Exits

Module Name	Description	When Available	Where to Look
ARCBEEXT	ABARS backup error exit	During aggregate backup.	"ARCBEEXT: ABARS Backup Error Installation Exit" on page 206
ARCCREXT	ABARS data set conflict resolution exit	During aggregate recovery verification when an INCLUDE data set to be recovered has the same name as a data set already existing at the aggregate recovery site.	"ARCCREXT: ABARS Conflict Resolution In Installation Exits" on page 207
ARCEDEXT	ABARS expiration date exit	Prior to allocation of an ABACKUP output file.	"ARCEDEXT: ABARS Expiration Date Installation Exit" on page 210
ARCM2EXT	ABARS migration level 2 data set exit	During aggregate backup when a data set on a migration level 2 volume is to be backed up.	"ARCM2EXT: ABARS Migration Level 2 Data Set Installation Exit" on page 211
ARCSKEXT	ABARS data set skip exit	During aggregate recovery for each data set being restored.	"ARCSKEXT: ABARS Data Set Skip Installation Exit" on page 212
ARCTVEXT	ABARS tape volume exit	When a DFSMShsm-owned ABARS tape no longer contains valid data, and therefore becomes empty. This exit also supports non-ABARS, DFSMShsm-owned tapes. See Chapter 6, "DFSMShsm Installation Exits," on page 155, "Characteristics of the ARCTVEXT Exit" on page 200 for more information.	"ARCTVEXT: Tape Volume Installation Exit" on page 213

Using DFSMShsm ABARS Exits

This section contains information that is common to all DFSMShsm ABARS installation exits.

Installing DFSMShsm ABARS Exits

The ABARS exits are loaded at the startup of the DFSMShsm ABARS secondary secondary address space, if the EXITON parameter is specified for the ABARS exits associated with ABACKUP or ARECOVER functions. The ARCBEEXT, ARCM2EXT, ARCEDEXT, and ARCTVEXT exits are associated with the ABACKUP function and the ARCCREXT, ARCSKEXT, and ARCTVEXT exits are associated with the ARECOVER function.

The ARCTVEXT exit must run in 24-bit addressing mode below the 16MB line.

The ABARS exits receive control in 31-bit mode, but all passed parameters are in 24-bit storage.

See "Adding a New Exit" on page 3.

Writing DFSMShsm ABARS Exits

You can tailor DFSMShsm ABARS functions with the DFSMShsm ABARS installation exits. As you write exits, remember that they run as if they were DFSMShsm code. Therefore, it is recommended that you routinely incorporate the following considerations whenever you write ABARS installation exits:

- Keep supervisor services such as I/O to a minimum.
- It is not necessary to write the ABARS installation exits as reentrant, with the exception of ARCTVEXT (ABARS tape volume exit).

Also always remember that DFSMShsm installation exits:

- Run enabled for interrupts.
- Run in problem program state.
- Run in either the DFSMShsm primary address space or the ABARS secondary address space.
- Have pageable storage.
- Are protected by ESTAE.
- Are entered in the standard address space protection key of 8. Control must return in the same key as at entry.
- Execute in an authorized program facility (APF) authorized address space.

Note: Since your load modules are not intended to be job step tasks, do not link edit them with APF authorization (AC=1). Doing so would be an unnecessary risk to system integrity and security.

Replacing ABARS Exits

The ABARS exits are unlike the other DFSMShsm exits because they are not dynamically loaded and unloaded when the SETSYS EXITON and SETSYS EXITOFF commands are specified. Instead, ABARS exits are loaded when the ABARS secondary address space is initialized if the SETSYS EXITON command has been specified.

To replace an ABARS exit:

1. Link-edit the new version of the exit and place it in the appropriate link library.
2. Issue the F LLA,REFRESH command.
3. Issue the SETSYS EXITON command for the ABARS exit you want to call during ABARS processing.
4. Issue the ABARS ABACKUP or ARECOVER command.

The new exit will be loaded and called by ABARS processing.

When the SETSYS EXITOFF command is issued for an exit that is active, the exit is no longer called, but it is not unloaded. When the SETSYS EXITON command is specified for an exit that is not active, the exit is not loaded or called until an ABARS command causes a secondary address space to be initialized.

Registers on Entry to DFSMShsm ABARS Installation Exits

Before the exit is called, the contents of the registers are:

Register
Contents

0
Not applicable

1
Address of input parameters

2–12
Not applicable

13
Address of register save area

14
Caller's return address

15
Address of exit entry point

Registers on Return from DFSMShsm ABARS Installation Exits

Before the exit routine ends, the contents of the registers are:

Register
Contents

0
Not applicable

1–14
Restored to contents at entry

15
Not applicable

Calling DFSMShsm ABARS Installation Exits

The DFSMShsm ABARS installation exits must be accessible by the LOAD macro and must communicate with the standard MVS linkage for registers. "Registers on Entry to DFSMShsm ABARS Installation Exits" on page 204 and "Registers on Return from DFSMShsm ABARS Installation Exits" on page 205 describe standard MVS linkage with which DFSMShsm complies. The exits must save and restore registers. DFSMShsm does not place return codes in the registers; it includes them as parameters in the parameter list for exits that provide return codes. An installation exit cannot issue the DFSMShsm supervisor call instruction (SVC).

Each exit parameter list contains pointers to the information described in the discussion of each individual exit. It points to copies of the information unless specifically stated otherwise.

Because DFSMShsm responds differently to abends that occur for different exits, the actions DFSMShsm takes are described for each exit. Generally, when an exit abends, DFSMShsm issues a HOLD on the function, writes a message, and does not call the exit again until either a DFSMShsm RELEASE command releases the function or a SETSYS command requests the function.

If an exit abends and you require a dump for problem determination, see *z/OS DFSMShsm Diagnosis*. Exits are not subject to TRAP commands.

Table 69 on page 205 lists the hex values for exits that do selective processing based on the device type DFSMShsm is processing.

Table 69. Hexadecimal Values for UCB Device Types

Unit Name	UCB Device Type Third and Fourth Bytes (Note 1)
3380	X'200E'

Table 69. Hexadecimal Values for UCB Device Types (continued)

Unit Name	UCB Device Type Third and Fourth Bytes (Note 1)
3390	X'200F'
3420	X'8003'
3423	X'8082'
3480	X'8080' (Note 2)
3480X	X'8080' (Note 2)
3490	X'8081'
3590-1	X'8083'
9345	X'2004'

Note:

1. The logical device names for some tape devices are different from their physical device names. 3480XF tape devices are identified with 3480X in the JCL; therefore, specify 3480X in the JCL when identifying 3480XF tape devices to DFSMShsm.
2. A tape device supports improved data recording capability if the 04 bit is on in the second byte of the four-byte UCB device-type code.

Creating User-Defined Messages

Message numbers ARC9000 through ARC9299 have been set aside for use by DFSMShsm installation exits, either as samples supplied by DFSMShsm or messages written by customers. For an explanation of these messages, customers have to locate the issuing exit or any user-created documentation for that exit.

The ARCRPEXT installation exit is able to pass messages back to DFSMShsm, and DFSMShsm writes these messages in the migration or backup activity log. Message numbers ARC9000 through ARC9199 are intended for use specifically for the ARCRPEXT exit. User messages issued by other DFSMShsm installation exits would generally be write-to-operator (WTO) messages.

ARCBEEXT: ABARS Backup Error Installation Exit

You can use the ABARS backup error exit to skip any data set associated with an I/O error, SDSP allocation error, data set noncataloged error, error during DFSMSdss dump processing, or an error enqueuing the ARCDSN resource, so that the data set is not backed up. This allows aggregate backup to complete without ending early should any of these errors occur.

Characteristics of the ARCBEEXT Exit

The ABARS backup error exit receives control during aggregate backup processing, after verification has completed successfully, when:

- An error occurs reading a data set, including a DFSMShsm control data set (CDS)
- An error occurs in allocating an small data set packing (SDSP) data set
- An error occurs while DFSMSdss is dumping level 0 DASD data sets in the INCLUDE list
- An uncataloged data set is encountered in the selection data set
- A data set fails serialization when ABARS is attempting to enqueue on the major name of ARCDSN and the minor name consisting of a data set name

A distinction is made between CDS I/O errors, non-CDS I/O errors, and SDSP allocation errors.

Recovering from an Abend of ARCBEEXT Processing

If the ABARS backup error exit abends, DFSMShsm:

- Fails aggregate backup
- Holds aggregate backup
- Issues message ARC6187E and records it in the ABARS activity log

DFSMShsm does not disable the exit. Analyze the cause of the abend to determine if it is repeatable. If it is, disable the exit by issuing the SETSYS EXITOFF(BE) command before releasing and restarting the aggregate backup task.

ARCBEEXT Parameter List

Register 1 contains the address of the ARCBEEXT parameter list as shown in Table 70 on page 207.

Table 70. ARCBEEXT Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of 44-byte area containing the name of the data set that encountered the error.
04 (X'04')	4	The address of a 4-byte flag field that shows the error type the data set encountered. The bits are: **1... ....** This is a non-CDS I/O error. **.1.. ....** This is a CDS I/O error. **..1. ....** This is an SDSP allocation error. **...1 ....** Data set not cataloged. **.... 1...** Data set failed serialization. **.... .1..** Data set failed DFSMSdss dump processing.
08 (X'08')	4	The address of a fullword binary return code field.

ARCBEEXT Return Codes

The following list contains the ARCBEEXT return codes. You must put one of these codes, in binary, into the area pointed to at offset 08 of the parameter list.

Return Code
Description

0 (X'00')
The aggregate backup fails.

4 (X'04')
Do not back up this data set. Continue the backup with the next data set.

ARCCREXT: ABARS Conflict Resolution In Installation Exits

You can use the ABARS conflict resolution exit to resolve like-named data set conflicts by failing the recovery, skipping the data set, replacing the existing data set, or renaming the data set.

Characteristics of the ARCCREXT Exit

The ABARS conflict resolution exit receives control during aggregate recovery verification when an INCLUDE data set to be recovered has the same name as a data set already existing at the aggregate recovery site. The ARCCREXT exit is called after ARCSKEXT.

Table 71 on page 208 shows the possible decisions you can make concerning the ARCCREXT installation exit and the procedures needed to inform DFSMShsm of your decision.

Table 71. Possible Results of ARCCREXT Processing

IF YOU WANT TO	THEN
Stop the aggregate recover	Place return code 0 in the return code area of the parameter list.
Skip the data set in conflict	Place return code 4 in the return code area of the parameter list.
Replace the existing data set with the data set being recovered	Place return code 8 in the return code area of the parameter list.
Rename the data set being recovered	Perform the following steps: 1. Put the new name in the area pointed to by the parameter list at offset 04. Pad the name on the right with blanks. You cannot rename a migrated VSAM data set. If you try, DFSMShsm skips the recovery of the VSAM data set. Migrated non-VSAM data sets can be renamed if only the high-level qualifier of the data set name is changed. Using this exit to rename more than just the data set's high-level qualifier causes an error (message ARC6325E) and the data set will not be processed during ARECOVER. 2. Place return code 12 in the return code area of the parameter list.

The disposition you specify for a data set in this installation exit takes precedence over subsequent processing. A data set is replaced or renamed, if specified, regardless of whether the ARECOVER REPLACE command is specified.

Recovering from an Abend of ARCCREXT Processing

If the ABARS conflict resolution exit abends, DFSMShsm:

- Fails aggregate recovery
- Holds aggregate recovery
- Issues message ARC6187E and records it in the ABARS activity log

DFSMShsm does not disable the exit. Analyze the cause of the abend to determine if it is repeatable. If it is, disable the exit by issuing the SETSYS EXITOFF(CR) command before releasing and restarting aggregate recovery.

ARCCREXT Parameter List

Register 1 contains the address of the ARCCREXT parameter list as shown in Table 72 on page 208.

Table 72. ARCCREXT Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of a 44-byte area containing the real name of the data set about to be recovered.

Table 72. ARCCREXT Parameter List (continued)

Offset	Length or Bit Pattern	Description
04 (X'04')	4	The address of a 44-byte area containing the name that you want replace the existing data set name with (you specify a return code of 12). You must pad the name on the right with blanks to total 44 characters.
08 (X'08')	4	The address of a 1-byte flag field containing the following: **1... ....** The allowable limit of 256 renamed data sets has been reached. **.1.. ....** The allowable limit has not been reached. **..1. ....** The recovery will be processed as a DFSMSdss restore. **...1 ....** This is a migrated data set. **.... 1...** This is a VSAM data set. **.... .1..** An existing GDG base name has been encountered. When this bit is set to '1' (on), the GDG base name is passed in the DSNAME field. If you want to recover associated GDSs using the existing GDG base, ensure that you code a return code of '8' for the exit's return code. If you code any other return code, the ARECOVER command fails unless the command is issued with the REPLACE parameter. **.... ..1.** This data set naming conflict occurs when you attempt to restore an ICF User Catalog.
12 (X'0C')	4	The address of a fullword binary return code.
16 (X'10')	4	The address of a 1-byte flag field containing the following: **1... ....** The data set is from the INCLUDE list. **.1.. ....** The data set is from the ACCOMPANY list. **..1. ....** The data set is from the ALLOCATE list.

ARCCREXT Return Codes

The following list contains the ARCCREXT return codes. You must put one of these codes, in binary, into the area pointed to at offset 12 of the parameter list.

Return Code
Description

0 (X'00')
The aggregate recovery fails.

4 (X'04')
Aggregate recovery does not recover the data set, but skips to the next data set to be processed.

8 (X'08')
Aggregate recovery replaces the existing data set at the recovery site with the like-named data set being recovered.

12 (X'0C')
Aggregate recovery renames the data set being recovered. You must specify the new name in the parameter list.

16(X'10')
Aggregate recovery renames the existing data set at the recovery site with the returned new high-level qualifier before recovery. The source data set is recovered with the original name. This action is comparable to the DATASETCONFLICT RENAMETARGET(level) parameter.

ARCEDEXT: ABARS Expiration Date Installation Exit

You can use the ABARS expiration date exit to modify the expiration date for ABARS ABACKUP output tapes. Aggregate backup processing sets the expiration date of aggregate (ABACKUP) output tapes to 99365 as a default. If you want to modify the expiration date for the ABACKUP tapes, you must override the default expiration date by using the ARCEDEXT installation exit.

Characteristics of the ARCEDEXT Exit

You do not need to write the ARCEDEXT to be reentrant.

The expiration date is expressed in packed decimal hex digits X'0cyydddf', where:

Variable
Description

0c
The century:

If C is 0, the year is 19*yy*

If C is 1, the year is 20*yy*

yy
The last two digits in the year.

ddd
The Julian day (1–366)

f
The sign digit

For example, January 1, 1996 would be represented as X'0096001F'. January 1, 2010 would be represented as X'0110001F'.

Recovering from an Abend of ARCEDEXT Processing

If the ABARS expiration date exit abends, DFSMShsm:

- Fails the aggregate backup
- Holds the aggregate backup processing
- Issues message ARC6187E and records it in the ABARS activity log

DFSMShsm does not disable the exit. Analyze the exit to determine if the abend is repeatable. If it is, disable the exit by issuing the SETSYS EXITOFF(ED) command before releasing aggregate backup.

ARCEDEXT Parameter List

Table 73 on page 210 shows that register 1 contains the address of the ARCEDEXT parameter list.

Table 73. ARCEDEXT Parameter List

Offset	Length or Bit Pattern	Description
0 (X'00')	4	The address of a 44-byte area containing the name of the aggregate backup file being processed.

Table 73. ARCEDEXT Parameter List (continued)

Offset	Length or Bit Pattern	Description
4 (X'04')	4	The address of a fullword packed decimal field containing the expiration date for the aggregate backup file. On input to the exit, the field is primed with a value of X'0099365F'. The exit can alter this value to indicate a new retention period or date. If the ABACKUP command is issued to back up an aggregate group defined in a pre-DFSMS/MVS 1.1.0 environment, the expiration date or retention period is passed in this field. The next field indicates whether an expiration date or retention period is being passed.
8 (X'08')	4	The address of a 4-byte field of flags. The first bit indicates whether the exit is being passed a retention period or an expiration date. **1... ....** Indicates that a retention period is being passed to the exit. **0... ....** Indicates that an expiration date is being passed to the exit.
12 (X'0C')	4	The address of a fullword binary return code.

ARCEDEXT Return Codes

The following list contains the ARCEDEXT return codes. You must put one of these codes, in binary, into the area pointed to at offset 12 of the parameter list.

Return Code
Description

0 (X'00')
You have not altered the expiration period.

4 (X'04')
You have altered the expiration period.

ARCM2EXT: ABARS Migration Level 2 Data Set Installation Exit

You can use the ABARS migration level 2 data set exit to skip any or all data sets residing on migration level 2 volumes so they are not backed up.

Characteristics of the ARCM2EXT Exit

The ABARS migration level 2 data set exit receives control during aggregate backup (after verification has completed successfully) when a data set on an migration level 2 volume is to be backed up.

Recovering from an Abend of ARCM2EXT Processing

If the ABARS migration level 2 data set exit abends, DFSMShsm:

- Fails aggregate backup
- Holds aggregate backup processing
- Issues message ARC6187E and records it in the ABARS activity log

DFSMShsm does not disable the exit. Analyze the abend to determine if it is repeatable. If it is, disable the ARCM2EXT exit by issuing the SETSYS EXITOFF(M2) command before releasing aggregate backup.

ARCM2EXT Parameter List

Register 1 contains the address of the ARCM2EXT parameter list as shown in Table 74 on page 212.

Table 74. ARCM2EXT Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of a 44-byte area containing the name of the migration level 2 data set about to be backed up.
04 (X'04')	4	The address of a fullword binary area containing the UCB device type (from the UCBTYP field of the UCB) of the device containing the data set. See Table 69 on page 205
08 (X'08')	4	The address of a fullword binary return code.

ARCM2EXT Return Codes

The following list contains the ARCM2EXT return codes. You must put one of these codes, in binary, into the area pointed to at offset 08 of the parameter list.

Return Code
Description

00 (X'00')
Permit aggregate backup to back up this data set.

04 (X'04')
Do not permit aggregate backup to back up this data set.

ARCSKEXT: ABARS Data Set Skip Installation Exit

You can use the ABARS data set skip exit to skip any data set so that it is not recovered.

Characteristics of the ARCSKEXT Exit

The ARCSKEXT exit receives control during aggregate recovery (during verification) after DFSMShsm determines that the data set should be recovered, but before actual recovery. This exit is called before the ARCCREXT exit and prior to any data set conflict resolution processing.

When a data set has been renamed, ARCSKEXT is called twice if either the ARECOVERNEWNAMELEVEL or ARECOVERNEWNAMEALL parameter is specified on the ARECOVER command. ARCSKEXT is first called with the original data set name; if it is not skipped and the return code from the exit is 0, it is called again with the new data set name that resulted from the rename action.

Recovering from an Abend of ARCSKEXT Processing

If the ABARS data set skip exit abends, DFSMShsm:

- Fails aggregate recovery
- Holds aggregate recovery processing
- Issues message ARC6187E and records it in the ABARS activity log

DFSMShsm does not disable the exit. Analyze the cause of the abend to determine if it is repeatable. If it is, disable the exit by issuing the SETSYS EXITOFF(SK) command before releasing and restarting the aggregate recovery task.

ARCSKEXT Parameter List

Register 1 contains the address of the ARCSKEXT parameter list as shown in Table 75 on page 213.

Table 75. ARCSKEXT Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	4	The address of a 44-byte area containing the name of the data set about to be recovered.
04 (X'04')	4	The address of a fullword binary return code.

ARCSKEXT Return Codes

The following list contains the ARCSKEXT return codes. You must put one of these codes, in binary, into the area pointed to at offset 04 of the parameter list.

Return Code
Description

00 (X'00')
Permit aggregate recovery to recover this data set.

04 (X'04')
Have aggregate recovery skip this data set.

ARCTVEXT: Tape Volume Installation Exit

You can use the tape volume exit to let a tape management system know that DFSMShsm is releasing ownership of a DFSMShsm tape.

Attention: ARCTVEXT exit is not required to communicate with DFSMSrmm. Invoke the ARCTVEXT exit only if you are using another tape management product.

The ARCTVEXT exit is used for both ABARS and non-ABARS DFSMShsm-owned tapes. You can find a description of this exit in Chapter 6, “DFSMShsm Installation Exits,” on page 155, under “ARCTVEXT: Tape Volume Installation Exit” on page 200.

Note: Many tape management product vendors provide their own version of this exit.

Chapter 8. DFSMSdss Installation Exits

This topic explains how you can customize DFSMSdss by writing exit routines. DFSMSdss exits routines are:

Table 76 on page 215 describes the DSS Installation Exits.

Table 76. DSS Installation Exits

Module Name	Description	When Available
ADRDYEXT _EXIT1	DFSMSdss Dynamic Exit	
ADRUPSWD	Authorization Exit Routine	Volume level and data set level
ADRUENQ	Enqueue Exit Routine	
ADRUIXIT	Options Exit Routine	When DFSMSdss is invoked and also before the processing of each task.
ADRREBLK	Reblock Exit Routine	See "Reblock Installation Exit Routine (ADRREBLK)" on page 237.

Installing and Replacing DFSMSdss Installation Exit Routines

See "Replacing an Existing Exit" on page 3. Your routine must have an entry point name that matches the exit name. Your CSECT is linked together with certain system CSECTs into a single load module.

Characteristics of DFSMSdss Installation Exit Routines

All DFSMSdss Installation Exits are called:

- In 31-bit addressing mode
- With authorized program facility (APF) authorization (if DFSMSdss is APF-authorized), and
- In key 8 problem state.

The exits are intended for use only by system programmers who must use caution with APF authorization. As with all subroutines, do *not* link edit installation exits with APF authorization.

DFSMSdss installation exits must:

- Observe standard register save and restore functions
- Be written in reentrant code
- Be link-edited with the DFSMSdss load module, ADRDSSU, except for ADRDYEXT_EXIT1
- Restore the execution environment (31-bit addressing mode, key 8, problem state) before returning to DFSMSdss.

DFSMSdss Dynamic Exit (ADRDYEXT_EXIT1)

The ADRDYEXT_EXIT1 dynamic exit uses the dynamic exits service, CSVDYNEX, which is described in *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN*. Using this service, the ADRDYEXT_EXIT1 dynamic exit has been defined to allow multiple exit routines to be invoked, to process the functions that the dynamic exit supports. The ADRDYEXT_EXIT1 dynamic exit will support functions related to data set notification related events. This includes:

- Closing allocated file systems. An exit routine will gain control to determine if any action needs to be taken for a data set for which serialization cannot be performed because the data set was allocated and

the SMS management class for class transition specified USEREXIT to indicate that the dynamic exit should be invoked.

Using this facility, the system provides a dummy exit routine, ADRDYX01, which is defined to be called when the ADRDYEXT_EXIT1 dynamic exit routine is invoked by the system. A sample version of this dummy exit routine is provided in the SAMPLIB as member ADRDYXS1. This exit routine can be replaced with installation code, or installations can associate their own exit routines with this dynamic exit. For more details, see "Adding an Exit Routine to an Exit" in *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN*.

Each defined exit routine to the ADRDYEXT_EXIT1 dynamic exit will be called when the dynamic exit is called by the system. Since the dynamic exits facility allows multiple exit routines to be simultaneously defined to a single exit, coordination of processing between these exit routines is necessary.

Characteristics of the DFSMSdss Dynamic Exit, ADRDYEXT_EXIT1

All DFSMSdss exit routines connected to ADRDYEXT_EXIT1 are called:

- In 31-bit addressing mode
- With authorized program facility (APF) authorization (if DFSMSdss is APF-authorized)
- In key 8 problem state.

They also must:

- Observe standard register save and restore functions
- Be written in reentrant code
- Restore the execution environment (31-bit addressing mode, key 8, problem state) before returning to DFSMSdss.

The parameter list and save area passed to the exit routines resides in fetch protected, key 0 storage.

Registers on Entry to ADRDYEXT_EXIT1

The registers on entry to the ADRDYEXT_EXIT1 exit routine contain:

Register
Contents

1
Address of the parameter list passed to each exit routine

13
Address of an 18-word save area

14
Return address

15
Address of the entry point to the called exit routine

ADRDYEXT_EXIT1 Parameter List

Register 1 contains the address of the exit routine parameter list as shown in Table 77 on page 216, with a storage protect key of 8. You can use the ADRDEX01 mapping macro to map this parameter list.

Table 77. ADRDYEXT_EXIT1 Parameter List

Offset	Length or Bit Pattern	Description
00 (X'00')	8	The EBCDIC 'ADRDEX01' ID.
08 (X'08')	2	The length of the parameter list, which is set to 48 bytes.

Table 77. ADRDYEXT_EXIT1 Parameter List (continued)

Offset	Length or Bit Pattern	Description
10 (X'0A')	1	The function code, as follows: **Value** **Meaning** **X'01'** The request is to process open data sets. A class transition could not be performed because DFSMSdss could not serialize on the data set and the SMS management class ACS routine for class transition specified that an EXIT should be invoked.
11 (X'0B')	1	The function code extension. For function code X'01', processing of open data sets, the function extension codes are: **Value** **Meaning** **X'01'** The call is to close an open data set **X'02'** The call is to open a closed data set

Table 77. ADRDYEXT_EXIT1 Parameter List (continued)

Offset	Length or Bit Pattern	Description
12 (X'0C')	4	Flags set to describe the DSS being performed, as follows: • Byte 0: **Bit** **Meaning** **0** DEFRAG OPERATION **1** COPY OPERATION **2** RESTORE OPERATION **3** DUMP OPERATION **4** PRINT OPERATION **5** COPYDUMP OPERATION **6** NEWALLOC OPERATION **7** COMPRESS OPERATION • Byte 1: **Bit** **Meaning** **0** RELEASE OPERATION **1** CONVERT VOLUME OPERATION **2** BUILDSA OPERATION **3** DEFRAG CONSOLIDATE **4** CGCREATED OPERATION **5** CONSOLID OPERATION **6** RESERVED **7** RESERVED • Byte 2: **Bit** **Meaning** **0** INPUT DSN IS VSAM • Byte 3 is unused.
16 (X'10')	4	The address on entry of a the data set name to be processed.
20 (X'14')	4	The address on entry of the 44-byte name of the integrated catalog facility catalog in which the data set is cataloged.

Table 77. ADRDYEXT_EXIT1 Parameter List (continued)

Offset	Length or Bit Pattern	Description
24 (X'18')	4	The address of the volume serial number.
28 (X'1C')	4	The address of an 18-byte area that defines who holds the data set enqueue. In order, this includes: • The 2-byte ASID of the ENQ requester • The 8-byte job name of the ENQ requester • The 8-byte system name of the ENQ requester.

Registers on Return from ADRDYEXT_EXIT1

The registers on return from the ADRDYEXT_EXIT1 exit routine contain:

Register
Contents

0
Possible diagnostic code set by the exit routine

15
Return code

ADRDYEXT_EXIT1 Return Codes

For function code X'01', the return codes are defined as follows:

Return Code
Description

00 (X'00')
The data set was not processed by the exit routine. For a call to close an open data set, this return code indicates that the call to open the data set is not necessary. This return code indicates that no further action is required by DFSMSdss.

04 (X'04')
The data set was processed successfully by the exit routine. The data set was closed for a call to close an open data set and opened for a call to open a data set. For a call to close an open data set, this return code indicates that DFSMSdss will move the data set and that the dynamic exit will be called to open the closed data set.

08 (X'08')
The data set was processed by the exit routine, but completed in error. A diagnostic code as set by the exit routine could be placed in register zero. This return codes indicates that no further action is required by DFSMSdss.

Authorization Installation Exit Routine (ADRUPSWD)

DFSMSdss supports authorization checking of password- and RACF-protected data sets and volumes. DFSMSdss provides an exit routine to allow you to control or override the authorization checks (RACF, password, or operator) done by DFSMSdss.

The ADRUPSWD exit is not called if any of the following is true:

- DFSMSdss is APF-authorized (when DFSMSdss is called by an APF-authorized program or invoked from an APF-authorized library using JCL) and NOPASS is specified on the PPT statement in the SCHEDxx parmlib member. The PPT statement defines the program properties table.
- The user has the proper level of RACF DASDVOL authority.
- For DEFRAG or CONSOLIDATE, if a RACF DASDVOL profile exists for the volume that is being processed. If the DASDVOL profile exists at all, either it allows the volume to be accessed (so there is no need to

call the exit), or it causes the DEFRAG or CONSOLIDATE task to fail with a 913 abend (so DFSMSdss cannot call the exit).

Otherwise, the exit routine supplied with DFSMSdss causes DFSMSdss to perform the following processing:

- Not relocate protected data sets for which you do not have read access during DEFRAG or CONSOLIDATE. You must supply a password for each protected data set.
- Check authorization at the data set level for all functions and CONVERTV.

Note: DASDVOL is not supported for system-managed volumes.

If a data set is protected by using both RACF and a password, only RACF checking is done. Authorization for password-protected non-VSAM data sets is checked by comparing the user-supplied password to that in the PASSWORD data set. Authorization for VSAM data sets is also checked. The authorization check for VSAM data sets not cataloged in the integrated catalog facility catalog fails if the data set has more than 16 extents.

You can use the authorization exit to bypass authorization checking and improve DFSMSdss performance. However, there is a trade-off between security and performance.

Installation-Supplied Authorization Exit Routine

You can write an exit routine to bypass authorization checking for both non-VSAM and VSAM data sets. The name of the routine must be ADRUPSWD. This routine is not given control for a COPYDUMP or CONVERTV operation.

The routine gets control at two levels: the volume level and the data set level (except for DEFRAG or CONSOLIDATE, which is given control at the volume level only). Control is always given at the volume level first, unless the data set is being renamed during a RESTORE operation. During RESTORE with a renamed data set, a data set level check is done against the source data on the dump tape first, and then a volume level check is done against the target volume. This is then followed by a data set level check against the target data set. A description of the two levels follows.

At the **volume level**, for DUMP, COPY, PRINT, and RESTORE, you can determine if the user is allowed to proceed with the function. During DEFRAG or CONSOLIDATE, you can determine if protected data sets can be relocated. This exit is not called for each data set when performing full volume operations.

Your installation authorization exit routine can do one of the following:

- End the operation.
- Request processing without authorization checking at the data set level.
- Request that the installation authorization exit be entered at the data set level.

At the **data set level**, the routine is given control during DUMP, COPY, PRINT, RESTORE, COMPRESS, and RELEASE only if the authorization exit routine instructs DFSMSdss to check authorization at the data set level. However, the routine is not given control at this level during DEFRAG or CONSOLIDATE. At the data set level, you can:

- End processing of the volume.
- End processing of the current data set.
- Bypass authorization checking of the current data set.
- Have DFSMSdss check authorization for the current data set.
- For the previous three options, you can specify whether or not the exit routine is entered for subsequent data sets.

Registers on Entry to the ADRUPSWD Exit

Register
Contents

1
Address of the ADRUPSWD parameter list

13
Address of an 18-word save area

14
Return address

15
Address of the entry point to ADRUPSWD

Register 1 contains the address of the ADRUPSWD parameter list. You can use the ADRUPB mapping macro to map this parameter list (see Table 78 on page 222). The four word parameter list includes:

First word
For a volume (fourth word, byte 2, bit 0 is on), this field is zero. For a VSAM data set not cataloged in the integrated catalog facility, or for a non-VSAM data set (fourth word, byte 2, bits 0 and 1 are off), this is a pointer to the format-1 or format-8 DSCB for the protected data set. (For the format-1 or format-8 DSCB description, see *z/OS DFSMSdfp Advanced Services*.) For a VSAM data set cataloged in the integrated catalog facility (fourth word, byte 2, bit 0 is off and bit 1 is on), this is a pointer to the 44-byte name of the protected VSAM cluster.

Second word
For a volume (fourth word, byte 2, bit 0 is on), or for a VSAM data set not cataloged in the integrated catalog facility, or for a non-VSAM data set (fourth word, byte 2, bits 0 and 1 are off), this field is zero. For a VSAM data set cataloged in the integrated catalog facility (fourth word, byte 2, bit 0 is off and bit 1 is on), this is a pointer to the 44-byte name of the integrated catalog facility catalog in which the protected VSAM cluster is cataloged.

Third word
Pointer to the volume serial number.

Fourth word
Flags are set as follows:

Byte 0

Bit
Meaning

0
End of the parameter list. This is always set to 1.

1
Set to 1 if read access is required on the volume. Set to 0 if write access is required.

2
Set to 1 for a DUMP operation.

3
Set to 1 for a RESTORE operation.

4
Set to 1 for a COPY operation.

5
Set to 1 for a PRINT operation.

6
Set to 1 for a DEFRAG operation.

7
Set to 1 for a COMPRESS operation.

Byte 1

Bit
Meaning

0
Set to 1 for DEFRAG, PRINT, DUMP, or RESTORE for a data set operation.

1
Set to 1 for a full-volume DUMP, RESTORE, or COPY operation.

2
Set to 1 for a tracks DUMP, RESTORE, COPY, or PRINT operation.

3
Set to 1 for a RELEASE operation.

4 to 7
Reserved.

Byte 2

Bit
Meaning

0
Set to 1 for volume level entry. Set to 0 for data set-level entry.

1
Set to 1 for a VSAM data set cataloged in the integrated catalog facility. Set to 0 for all other data sets.

Byte 3
Reserved.

ADRUPB Parameter List

Table 78. ADRUPB Parameter List

Offset	Length or Bit Pattern	Name	Description
0 (X'0')	16	ADRUPB	
0 (X'0')	4	UPDSCBAD	0 if UPIND = 1. FORMAT 1 DSCB address if both UPIND and UPVSMDAT = 0. Address of 44-byte cluster name if UPIND = 0 and UPVSMDAT = 1.
4 (X'4')	4	UPCATP	Address of 44 byte catalog name if UPVSMDAT = 1.
8 (X'8')	4	UPUCBVSA	UCB volume serial number address
12 (X'C')	1	UPFLG1	Flag byte 1
	1...	UPEOL	End of list indicator. Always 1.
	.1..	UPFRTO	1 = source volume, 0 = destination volume.
	..1.	UPDUMP	1 = DUMP function
	...1	UPREST	1 = RESTORE function
	.... 1...	UPCOPY	1 = COPY function
	.... .1..	UPPRINT	1 = PRINT function
	.... ..1.	UPDEFRAG	1 = DEFRAG function
	1	UPCOMPR	1 = COMPRESS function

Table 78. ADRUPB Parameter List (continued)

Offset	Length or Bit Pattern	Name	Description
13 (X'D')	1	UPFLG2	Flag byte 2
	1...	UPDSN	1 = data set operation
	.1..	UPFULL	1 = full volume operation
	..1.	UPPART	1 = partial volume operation
	...1	UPRLSE	1 = RELEASE function
	.... 1...	UPBLDSA	1 = BUILDSA IPL(DASD)
	.... .1..	UPCGCR	1 = CGCREATED
	.... ..1.	UPCONS	1 = CONSOLIDATE
14 (X'E')	1	UPFLG3	Flag byte 3
	1...	UPIND	1 = volume level check, 0 = data set level check
	.1..	UPVSMDAT	1 = cluster and catalog names passed
15 (X'F')	1	UPFLG4	Flag byte 4 reserved
16 (X'10')			Force word alignment

Registers on Return from the ADRUPSWD Exit

Your ADRUPSWD routine should issue a return code in register 15 to indicate the course of action for further processing by DFSMSdss.

ADRUPSWD Return Codes

The return codes, in decimal, and their meanings are shown in Table 79 on page 223.

Note: The authorization installation exits routine supplied with DFSMSdss passes a return code of 12.

The first time ADRUPSWD is entered, the volume serial number address is passed. If check the passwords and have ADRDSSU return to you for each data set, on the next entry the DSNAME address is also passed.

Return Codes for Volume Level Entry

The codes listed in Table 79 on page 223 are valid for all functions except COPYDUMP and CONVERTV.

Table 79. Return Codes for Volume Level Entry

Return Code	End Processing for Volume?	Data Set Level Checks Required?	Should ADRUPSWD be Entered at Data Set Level?
0	No	No	No
8	No	Yes	Yes
12	No	Yes	No
20	Yes	-	-

Notes:

For DEFRAG or CONSOLIDATE:

- If a nonzero return code less than or equal to 12 is returned, all data sets on the volume are to be checked for authorization.
- A nonzero return code greater than 12 ends the function.

- If any data sets are protected and you do not have DASDVOL authority, a return code of 20 ends the function.

For all other functions: The function is ended with message ADR402E-15 if ADRUPSWD returns a return code other than those return codes listed for Volume Level Entry.

Return Codes for Data Set Level Entry

The codes listed in Table 80 on page 224 are valid for all functions except DEFRAG, CONSOLIDATE, and COPYDUMP.

Table 80. Return Codes for Data Set Level Entry

Return Code	End Processing for Volume?	End Processing of Data Set?	Perform Authorization Checking for Data Set?	Should ADRUPSWD be Entered Again?
0	No	No	No	No
4	No	No	No	Yes
8	No	No	Yes	Yes
12	No	No	Yes	No
16	No	Yes	-	Yes
20	Yes	-	-	-

Example of the ADRUPSWD Exit

This exit sets the return code to zero, which results in all data set authorization checks being bypassed.

Note: Use Figure 49 on page 224 as a learning aid. It is not guaranteed to run on a particular system without some modification.

```
*****************************************************
* ADRUPSWD USER EXIT.                               *
*   SETS RETURN CODE TO 0 INDICATING THAT DATA SET  *
*   AUTHORIZATION CHECKS WILL NOT BE PERFORMED.     *
*****************************************************
ADRUPSWD CSECT
ADRUPSWD AMODE 31
ADRUPSWD RMODE 24
          STM   14,12,12(13)          SAVE REGS IN PREVIOUS SAVEAREA
          LR    12,15                 ESTABLISH BASE REGISTER
          USING ADRUPSWD,12           SET ADDRESSABILITY TO THE EXIT
          LM    14,12,12(13)          RESTORE OTHER REGISTERS
          LA    15,0                  SET RETURN CODE TO 0
          BR    14                    RETURN
          END
```

Figure 49. Sample Listing of ADRUPSWD

Enqueue Installation Exit Routine (ADRUENQ)

The enqueue installation exits routine that is supplied with DFSMSdss passes a return code of zero. This causes DFSMSdss to do the following:

- For a full or tracks COPY or DUMP or tracks PRINT, enqueues the VTOC during the entire operation
- For a physical data set DUMP enqueues the VTOC during the entire DUMP

Restriction: The enqueue installation exit routine is not called during COMPRESS, CONVERTV, data set COPY, COPYDUMP, DEFRAG, CONSOLIDATE, logical data set DUMP, data set PRINT, or RELEASE processing. In addition, the installation exit routine is not called during DUMP full to a cloud storage object.

You can use this exit to cause DFSMSdss to enqueue the VTOC only for the duration of the VTOC access. If you do not enqueue the VTOC for the entire operation, you can improve performance and decrease the chance of deadlock. However, there is a trade-off. With the improved performance and reduced chance of deadlock, there is also decreased data integrity.

Installation-Supplied Enqueue Exit Routine

To access a volume while it is being dumped, either by another job under the control of a second initiator or by another processor in a shared DASD environment, you can write an exit routine to enqueue the VTOC only until it is processed.

The name of the routine must be ADRUENQ.

Registers on Entry to the ADRUENQ Exit

Register
Contents

1
Address of the ADRUENQ parameter list

13
Address of an 18-word save area

14
Return address

15
Address of the entry point to ADRUENQ

ADRUENQ Parameter List

Register 1 contains the address of the ADRUENQ parameter list. You can use the ADRUNQB mapping macro to map this parameter list (see Table 81 on page 226).

The first word of the parameter list is a pointer to the volume serial field for the volume being dumped. The second word of the parameter list contains flags that are set as follows:

Byte 0 Bit
Meaning

0
End of the parameter list. This is always set to 1.

1
Reserved.

2
Set to 1 for a DUMP operation

3
Reserved.

4
Set to 1 for a COPY operation.

5
Set to 1 for a PRINT operation.

6
Reserved.

7
Reserved.

Byte 1 Bit
Meaning

0
Set to 1 for a data set operation.

1
Set to 1 for a full volume DUMP or COPY operation.

2
Set to 1 for a tracks DUMP, COPY, or PRINT operation.

3 to 7
Reserved.

Byte 2 Bit
Meaning

0
Set to 1 if TOLERATE(ENQFAILURE) is used.

1-7
Reserved.

Byte 3
Reserved.

Table 81. ADRUNQB Parameter List

Offset	Length or Bit Pattern	Name	Description
0 (X'0')	8	ADRUNQB	
0 (X'0')	4	UNUCBVSA	UCB volume serial number address
4 (X'4')	1	UNFLG1	Flag byte 0
	1...	UNEOL	End of list indicator, always 1
	.x..		Reserved
	..1.	UNDUMP	1 = DUMP operation
	...x		Reserved
	.... 1...	UNCOPY	1 = COPY operation
	.... .1..	UNPRINT	1 = PRINT operation
	xx		Reserved
5 (X'5')	1	UNFLG2	Flag byte 1
	1...	UNDSN	1 = data set operation
	.1..	UNFULL	1 = full volume operation
	..1.	UNPART	1 = partial volume operation
	...x xxxx		Reserved
6 (X'6')	1	UNFLG3	Flag byte 2
	1...	UNTOLNQF	1 = TOLERATE(ENQFAILURE) specified
7 (X'7')	1		Reserved

Registers on Return from the ADRUENQ Exit

Your ADRUENQ routine should issue a return code in register 15 to indicate the course of action for further processing by DFSMSdss.

ADRUENQ Return Codes

Return Code
Description

0
Enqueue on the volume being dumped or copied for the duration of the operation.

4
Enqueue on the volume being dumped or copied only for the duration of VTOC access.

Example of the ADRUENQ Exit

This exit sets the return code to four, which results in the volume only being enqueued for the duration of the VTOC access during dump and copy operations.

Note: Use Figure 50 on page 227 as a learning aid. It is not guaranteed to run on a particular system without some modification.

```
*****************************************************
* ADRUENQ USER EXIT.                                *
*   SETS RETURN CODE TO 4 INDICATING THAT THE VOLUME *
*   WILL ONLY BE ENQUEUED FOR THE DURATION OF THE    *
*   VTOC ACCESS FOR DUMP AND COPY OPERATIONS.        *
*****************************************************
ADRUENQ  CSECT
ADRUENQ  AMODE 31
ADRUENQ  RMODE 24
          STM   14,12,12(13)          SAVE REGS IN PREVIOUS SAVEAREA
          USING ADRUENQ,15            SET ADDRESSABILITY TO THE EXIT
          LM    14,12,12(13)          RESTORE OTHER REGISTERS
          LA    15,4                  SET RETURN CODE TO 4
          BR    14                    RETURN
          END
```

Figure 50. Sample Listing of ADRUENQ

Options Installation Exit Routine (ADRUIXIT)

The options installation exits routine can control certain DFSMSdss tasks or change some defaults or specified options. It is given control when DFSMSdss is invoked and also before the processing of each task. During initial entry, it can specify where the I/O and application interface buffers are to reside, change the default SERIAL mode to PARALLEL, and request that RACF class checking be bypassed. It can specify to fall back to the EXCP support for DUMP output, RESTORE input, and COPYDUMP. When entered for each task, the routine can change the options, defaults, or values.

During a DEFRAG or CONSOLIDATE, an extent of a protected data set is relocated, and the old extent is cleared (written over with zeros) for security. This adds time to the DEFRAG and CONSOLIDATE process. You can prevent the clearing of the old extent with the options installation exits. If your exit sets the UFOERASE bit to zero, it prevents clearing an old extent.

The exit routine can force the DUMP function to end if a write error occurs on any of the output copies. This is done by setting bit UFOIACPY on. The default is to continue to produce as many error-free output copies as possible.

When you are doing full volume or tracks COPY or RESTORE and the VTOC index on the target volume must be rebuilt, the exit can control whether or not ICKDSF is invoked to rebuild the VTOC index. Invoking

ICKDSF from DFSMSdss can cause deadlock problems. If this happens, the exit can set UFOBLDIX to zero, and the VTOC index can be rebuilt manually by calling ICKDSF.

DFSMSdss uses the system authorization facility (SAF) interface to ensure that RACF at the 1.8.1 level or above is installed and active. If you are using a RACF-equivalent program that does not set the same level of information that DFSMSdss checks for, you can notify DFSMSdss that SAF with a RACF-equivalent program is installed by setting bit UFSAFOK on.

The exit routine supplied with DFSMSdss does not change any defaults, options, or values.

There are three categories of keywords that the installation enqueue exit routine can change:

- **Simple keywords with no subparameters:** The exit can set or reset the option. For example, the exit can force use of the CATALOG option or force the DELETE option not to be used.

  This category includes the following commands: ALLEXCP, CATALOG, CICSVRBACKUP, COMPRESS, CONCURRENT, COPYVOLID, DELETE, DUMPCONDITIONING, DYNALLOC, FCNOCOPY, FCWITHDRAW, FORCE, NULLMGMTCLAS, NULLSTORCLAS, PURGE, REPLACE, REPLACEUNCONDITIONAL, RACFLOG, RESET, SERIAL, PARALLEL, SHARE, SPHERE, UNCATALOG, VALIDATE, and WRITECHECK.

  Note: Although the exit can override NULLSTORCLAS and NULLMGMTCLAS, it cannot affect the STORCLAS or MGMTCLAS keywords. That is, it cannot change NULLSTORCLAS to STORCLAS(*x).*
- **Keywords with one or more subparameters having a limited number of values:** The exit can control the values of the subparameters. For example, the exit can force OPTIMIZE(4) to be used on all full volume dumps.

  This category includes the following commands: DEBUG(FRMSG(MINIMAL|SUMMARIZED|DETAILED)), FASTREPLICATION(REQUIRED|PREFERRED|NONE), FORCECP(*nnn*), FRAGMENTATIONINDEX(*n*), MINSECQTY(*n*), MINTRACKSUNUSED(*n*), OPTIMIZE(*n*), PROCESS(SYS1), PROCESS(UNDEF), READIOPACING(*n*), SELECTMULTI(ALL|ANY|FIRST), TOLERATE(ENQFAILURE), TGTALLOC(*x*), VOLCOUNT(*|N(*nn*)|SRC|ANY), and WAIT(*n*,*m*).
- **Keywords with a subparameter having many possible values, including one specific value:** The exit can either override the keyword entirely or use the one specific value. For example, the exit can force no reblocking to occur by overriding the REBLOCK keyword, or it can cause all eligible data sets to be reblocked by forcing REBLOCK(*). The exit cannot specify a data set name list with the REBLOCK keyword.

  This category includes the following commands: ALLDATA(*), REBLOCK(*), and RECATALOG(*).

Note: The exit cannot set conflicting options. For example, setting CONCURRENT and DELETE is an error.

You can use this exit to stop DFSMSdss functions. Then you can force the use of your own options and controls.

Installation-Supplied Options Exit Routine

You can write an exit routine to override certain defaults and user-specified parameters in DFSMSdss commands. The name of the routine must be ADRUIXIT. It must be written in reentrant code.

Registers on Entry to the ADRUIXIT Exit

Register
Contents

1
Address of the ADRUIXIT parameter list

13
Address of an 18-word save area

14
Return address

15
Address of the entry point to ADRUIXIT

ADRUFO Parameter List

Register 1 contains the address of the ADRUFO parameter list. You can use the ADRUFO mapping macro to map the parameter list. These sections are shown in the ADRUFO parameter list.

UFOHDR
contains information describing the type of entry and, for a function entry, details of the function to be scheduled. It also contains an offset to UFOFUNCT or UFOPARM and the addresses of UFOVOL for input and UFOVOL for output.

UFOFUNCT
contains information about the function to be scheduled and which ones can be altered by this exit routine. The bits are described in the parameter list. It is created for the function entry.

UFOPARM
contains bits that can be set to change defaults or override commands. The bits are described in the parameter list. It is created for the PARM change entry.

UFOVOL
is an array of entries describing the volumes used for input and output for the specific function commands.

```
1   ADRUFO

      OFFSET   OFFSET
      DECIMAL    HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
     ======== ======== ================  ============== ================================
            0      (0) STRUCTURE       24  ADRUFOB
            0      (0) CHARACTER       24   UFOHDR        HEADER OF UFO PARM LIST
            0      (0) CHARACTER        4    UFID         IDENTIFIER EBCDIC "UFO "
            4      (4) SIGNED           2    UFLEN        LENGTH OF PARM LIST
            6      (6) SIGNED           2    UFBDYOFF     OFFSET TO UFOFUNCT OR UFOPARM
            8      (8) ADDRESS          4    UFVOLI@      ADDR OF INPUT VOL LIST | 0
           12      (C) ADDRESS          4    UFVOLO@      ADDR OF OUTPUT VOL LIST | 0
           16     (10) BIT(16)          2    UFFUNCT      FUNCTION BEING PERFORMED
                                                          '0000'X = PARM CHANGE ENTRY
                                                          (THIS FIELD MUST BE KEPT IN
                                                          SYNCHRONIZATION WITH FBMAJOR
                                                          IN ADRFUNCB)
           16     (10) BIT(8)           1     UFFUNCT1    FUNCTION BYTE ONE
                       1... ....              UFFUDEF     1 = DEFRAG OPERATION
                       .1.. ....              UFFUCOPY    1 = COPY OPERATION
                       ..1. ....              UFFUREST    1 = RESTORE OPERATION
                       ...1 ....              UFFUDUMP    1 = DUMP OPERATION
                       .... 1...              UFFUPRT     1 = PRINT OPERATION
                       .... .1..              UFFUCPYD    1 = COPYDUMP OPERATION
                       .... ..1.              *           RESERVED
                       .... ...1              UFFUCOMP    1 = COMPRESS OPERATION
           17     (11) BIT(8)           1     UFFUNCT2    FUNCTION BYTE TWO
                       1... ....              UFFURLSE    1 = RELEASE OPERATION
                       .1.. ....              UFFUCONV    1 = CONVERTV OPERATION
                       ..1. ....              UFFUBLSA    1 = BUILDSA OPERATION
                       ...1 ....              *           RESERVED
                       .... 1...              UFFUCGCR    1 = CGCREATE OPERATION
                       .... .1..              UFFUCONS    1 = CONSOLIDATE OPERATION
                       .... ..1.              UFFUSREL    1 = SPACEREL OPERATION
                       .... ...1              *           RESERVED
           18     (12) CHARACTER        1    UFFIND       FUNCTIONAL INDICATORS
                       1... ....              UFFIFULL    1 = FULL VOLUME REQUEST (DUMP,
                                                          RESTORE, COPY & DEFRAG)
                       .1.. ....              UFFIPART    1 = PARTIAL REQUEST (DUMP,
                                                          RESTORE, COPY, DEFRAG & PRINT)
                       ..1. ....              UFFIFILT    1 = REQUEST BY FILTER/DSNAME
                                                          (DUMP, RESTORE & PRINT)
                       ...1 ....              UFFIPRTV    1 = PRINT VTOC
                       .... 1...              UFFLOGCL    1=LOGICAL PROCESSING FOR COPY,
                                                          DUMP, OR RELEASE: EITHER NO
                                                          INPUT VOLUMES SPECIFIED
                                                          (CATALOG FILTERING) OR 1 OF
                                                          FOLLOWING: LOGINDDNAME,
                                                          LOGINDYNAM, LOGDDNAME,
                                                          LOGDYNAM.
                       .... .1..              UFFPATH     1=UNIXFILE PROCESSING
                       .... ..11              *           RESERVED
```

```
         19     (13) CHARACTER        1    UFAIFLGS     APPLICATION INTERFACE FLGS
                     1... ....              UFAIINV      1=INVOKED BY APPL. INTERF.
                     .1.. ....              UFUIMAL      1=ADRUIM NOT TO BE GIVEN
                                                         CONTROL
                     ..1. ....              UFUIMCH      1=DO NOT ALLOW ADRUIM TO
                                                         MODIFY OPTIONS, VALUES
                     ...1 ....              UFSTOP       1=DO NOT SCHEDULE TASK (SAME
                                                         AS RETURN CODE 8 FROM
                                                         ADRUIXIT)
                     .... 1...              UFSYSIN      1=SYSIN OR ALTERNATE NOT
                                                         PRESENT ALLOWED IF
                                                         UFPARAM=XX1000XX
                     .... .1..              UFSYSPR      1=SYSPRINT/ALTERNATE NOT
                                                         PRESENT ALLOWED IF
                                                         UFPARAM=XX1000XX
                     .... ..1.              UFNOIN       1=NO INPUT TAPE - ONLY FOR
                                                         RESTORE
                     .... ...1              UFNOOUT      1=NO OUTPUT TAPE - ONLY FOR
                                                         DUMP
         20     (14) CHARACTER        1    UFFLAGS      FLAGS
                     1... ....              UFBYFCCK     1=BYPASS FACILITY CLASS CHECKS
                                                         IF USER AUTHORIZED
                     .1.. ....              UFSAFOK      1=IT IS OK TO USE THE SAF
                                                         INTERFACE AT THE HIGHEST
                                                         SUPPORTED LEVEL
                     ..1. ....              UFFREWCL     1=rewind on close
                     ...1 ....              UFIGCTNN     1=IGNORE CATALOG ENTRIES FOR
                                                         NEW NAMED DATA SET, VALID ONLY
                                                         FOR LOGICAL DUMP
                     .... 1...              UFFCFRRT     1=RETRY FLASHCOPY WITHOUT FAST
                                                         REVERSE RESTORE OPTION
                     .... .1..              UFBYFRVF     1=BYPASS CHECKING FOR EXISTING
                                                         FC RELATIONS DURING FAST
                                                         REVERSE RESTORE OPERATION
                     .... ..11               *           reserved
         21     (15) CHARACTER        3    *            RESERVED
         24     (18) CHARACTER        0   *

       OFFSET   OFFSET
       DECIMAL    HEX  TYPE         LENGTH  NAME (DIM)     DESCRIPTION
      ======== ======== ================  ============== ================================
             0      (0) STRUCTURE       48  UFOFUNCT       LIST OF OPTIONS
                                                           FOR FUNCTION. POINTED TO BY
                                                           THE ADDRESS OF UFOHDR +
                                                           UFBDYOFF. PRESENT IF ANY BITS
                                                           ARE ON IN UFFUNCT
             0      (0) BIT(8)           1   UFO1FLGS      1ST SET OF OPTION FLAGS
                     1... ....              UFO1COMP     1 = COMPRESS (DUMP)
                     .1.. ....              UFO1CVOL     1 = COPYVOLID (RESTORE & COPY)
                     ..1. ....              UFO1PURG     1 = PURGE (DUMP, COPY &
                                                         RESTORE)
                     ...1 ....              UFO1RESE     1 = RESET CHANGE BIT (DUMP)
                     .... 1...              UFO1WRCK     1 = WRITECHECK
                     .... .1..              UFO1ALD      1 = ALLDATA
                     .... ..1.              UFO1ALDL     1 = ALLDATA(LIST), 0 =
                                                         ALLDATA(*), VALID ONLY IF
                                                         UFO1ALD = 1. BIT MAY NOT BE
                                                         SET ON BY THE EXIT BUT MAY BE
                                                         RESET TO CHANGE ALLDATA(LIST)
                                                         TO ALLDATA(*)
                     .... ...1              UFO1ALLE     1 = ALLEXCP
             1      (1) BIT(8)           1   UFO2FLGS      2ND SET OF OPTION FLAGS
                     1... ....              UFO2DYNQ     0 = USE ENQ TO HOLD DATASET, 1
                                                         = USE DYNALOC TO HOLD DS
                                                         (DUMP, DEFRAG, PRINT &
                                                         RESTORE)
                     .1.. ....              UFO2ENQE     1 = ENQ EXCLUSIVE (DUMP,
                                                         RESTORE AND PRINT)
                     ..1. ....              UFO2ENQS     1 = ENQ SHARED IF EXCL FAILS-
                                                         DUMP, RESTORE & PRINT
                     ...1 ....              UFO2ENQN     1 = DO NOT ENQ IF EXCL & SHR
                                                         FAIL-DUMP, RESTORE & PRINT
                     .... 1...              UFO2DEL      1 = DELETE AFTER DS DUMP
                     .... .1..              UFO2CTLG     1 = CATALOG DATA SETS DURING A
                                                         DATA SET RESTORE, COPY
                     .... .1..               UFO2RECT    1 = RECATALOG DURING A DATA
                                                         SET RESTORE, COPY
                     .... ..1.              UFO2UNC      1 = UNCATALOG DATA SETS AFTER
                                                         A DATA SET DUMP, COPY
                     .... ...1              UFO2VALD     1 = VALIDATE: DUMP THE VSAM
                                                         INDEXED DATA SET IN NEW FORMAT
                                                         (LOGICAL DATA SET DUMP)
```

```
 2      (2) UNSIGNED         1   UFDUOPTM      OPTIMIZE VALUE (DUMP/COPY) (1,
                                                2, 3, OR 4)
 3      (3) BIT(8)           1   UFOINSOP      INSTALLATION OPTIONS @REL11
             1... ....             UFOERASE     1 = ERASE DASD TRACKS
             .1.. ....             UFOIACPY     1 = DUMP MUST PRODUCE ALL
                                                OUTPUT COPIES OR NONE AT ALL
             ..1. ....             UFOBLDIX     1 = INVOKE ICKDSF TO REBUILD
                                                VTOC INDEX
             ...1 ....             UFORACLG     RACFLOG=YES SPECIFIED OR FORCE
                                                RACF LOGGING
             .... 1...             UFOBK32K     TAPE BLK SIZE 32K
             .... .1..             UFOARBA      1=AUTORELBLKA
                                                SPECIFIED@LA71950
             .... ..1.             UFOMKMV      1=MAKEMULTI SPECIFIED
             .... ...1             UFOFLEAV     1=CLOSE LEAVE FLAG
 4      (4) UNSIGNED         4   UFOFRAGI      FRAGMENTATION INDEX(DEFRAG)
                                                9,90,900=900, 09=90, 009=9
 8      (8) UNSIGNED         4   *             VOLCOUNT stuff
 8      (8) BIT(8)           1    UFOVCFLG     VOLCOUNT flags
             1... ....             UFOVCCUR     1 = VOLCOUNT(*)
             .1.. ....             UFOVCSRC     1 = VOLCOUNT(SRC)
             ..1. ....             UFOVCNUM     1 = VOLCOUNT(N(nn))
             ...1 ....             UFOVCANY     1 = VOLCOUNT(ANY)
             .... 1...             UFOSMALL     1 = SELECTMULTI(ALL)
             .... .1..             UFOSMANY     1 = SELECTMULTI(ANY)
             .... ..1.             UFOSM1ST     1 = SELECTMULTI(FIRST)
             .... ...1             UFOCPFRC     1 = FORCECP keyword
 9      (9) UNSIGNED         1    UFOVCVAL     VALUE FOR N VOLUMES
10      (A) UNSIGNED         1    UFOCPDAY     DAYS value for FORCECP
11      (B) CHARACTER        1    *            reserved
12      (C) UNSIGNED         4   UFOMNSQT      MINIMUM SECONDARY ALLOCATION
                                                (MINSECQTY) FOR RELEASE
16     (10) UNSIGNED         4   UFOMNTUS      MINIMUM UNUSED TRACKS
                                                (MINTRACKSUNUSED) FOR RELEASE
20     (14) BIT(8)           1   UFO3FLGS      3RD OPTION FLAG BYTE
             1... ....             UFO3FORC     1= FORCE UNMOVABLES ON COPY,
                                                RESTORE
             .1.. ....             UFO3REPL     1 = REPLACE(DATASET COPY,
                                                RESTORE)
             ..1. ....             UFOFRBLK     1 = FORCE REBLOCKING OF DATA
                                                SETS (COPY, RESTORE)
             ...1 ....             UFODRBLK     1 = DISABLE REBLOCKING OF DATA
                                                SETS (COPY, RESTORE)
             .... 1...             UFOALLMU     1=SEARCH ALL VOLUMES FOR COPY
                                                OR DUMP
             .... .1..             UFOSPHER     1=PERFORM SPHERE PROCESSING
             .... ..1.             UFONOSMS     1=NULLSTORCLAS SPECIFIED
             .... ...1             UFONMGMT     1=NULLMGMTCLAS SPECIFIED
21     (15) UNSIGNED         1   UFOWAITS      WAIT TIME IN SECONDS BETWEEN
                                                RESERVE & ENQ RETRIES(ALL)
22     (16) UNSIGNED         1   UFOWAITR      NUMBER OF RETRIES ON RESERVE
                                                OR ENQ FAILURES(ALL COMMANDS)
23     (17) BIT(8)           1   UFOTGTAL      TGTALLOC FLAGS
             1... ....             UFOTGTCY     1 = CYLINDER
             .1.. ....             UFOTGTTR     1 = TRACK
             ..1. ....             UFOTGTBL     1 = BLOCK
             ...1 ....             UFOTGTSR     1 = SOURCE
             .... 1111             *            RESERVED
24     (18) BIT(8)           1   UFOPROCK      PROCESS OPTIONS( THIS
                                                FIELD MUST BE KEPT IN
                                                SYNC WITH FBPROCKW)
             1... ....             UFOPRUND     1 = PROCESS UNDEFDSORG
             .1.. ....             UFOPRSYS     1 = PROCESS SYS1
             ..11 1111             *            UNDEFINED
25     (19) BIT(8)           1   UFO4FLGS      FOURTH SET OF OPTION FLAGS
             1... ....             UFOT0REQ     CONCURRENT COPY REQUESTED
             .1.. ....             UFODCOND     DUMPCONDITIONING
             ..1. ....             UFOCVRBK     CICSVRBACKUP
             ...1 ....             UFOFCNC      FLASHCOPY NOCOPY
             .... 1...             UFOFCWD      FLASHCOPY WITHDRAW
             .... .1..             UFOFC2PP     ALLOW FC TO PPRC PRIM
             .... ..1.             UFOFCN2C     FLASHCOPY NOCOPY TO COPY
             .... ...1             UFOFCFRZ     FLASHCOPY CG FREEZE
26     (1A) UNSIGNED         2   UFORIOPC      READ I/O PACING
28     (1C) BIT(8)           1   UFO5FLGS      FIFTH OPTIONS FLAGS BYTE
             1... ....             UFOFRREQ     FASTREPLICATION(REQUIRED)
             .1.. ....             UFOFRPRF     FASTREPLICATION(PREFERRED)
             ..1. ....             UFOFRNO      FASTREPLICATION(NONE)
             ...1 ....             UFO5REPU     REPLACEUNCONDITIONAL
             .... 1...             UFOFCINC     FCINCREMENTAL
             .... .1..             UFOFCINL     FCINCREMENTALLAST
             .... ..1.             UFOFCVFR     FCVERIFY(REVERSE)
```

```
                   .... ...1             UFOFCVFN     FCVERIFY(NOREVERSE)
        29     (1D) BIT(8)           1   UFO6FLGS     SIXTH OPTIONS FLAGS BYTE
                   1... ....             UFOFRMSM     DEBUG(FRMSG(MINIMAL))
                   .1.. ....             UFOFRMSS     DEBUG(FRMSG(SUMMARIZED))
                   ..1. ....             UFOFRMSD     DEBUG(FRMSG(DETAILED))
                   ...1 ....             UFOHCOMP     HARDWARE COMPRESSION
                   .... 1...             UFODBTRC     DEBUG(TRACE)
                   .... .1..             UFODBSMS     DEBUG(SMSMSG)
                   .... ..1.             UFOFCVFY     FORCE FCCGVERIFY STOP
                   .... ...1             *            RESERVED
        30     (1E) UNSIGNED         1   UFOFCWTS     FCWAIT TIME IN SECONDS
        31     (1F) UNSIGNED         1   UFOFCWTR     FCWAIT MAX RETRY COUNT
        32     (20) BIT(8)           1   UFO7FLGS     SEVENTH OPTIONS FLAGS BYTE
                   1... ....             UFOFCSEF     FCSETGTOK(FAILRELATION)
                   .1.. ....             *            RESERVED FOR SPE
                   ..1. ....             UFO7CCAR     CONCURRENT(ANYREQ)
                   ...1 ....             UFO7CCVR     CONCURRENT(VIRTUALREQ)
                   .... 1...             UFO7CCCR     CONCURRENT(CACHEREQ)
                   .... .1..             UFO7CCAP     CONCURRENT(ANYPREF)
                   .... ..1.             UFO7CCVP     CONCURRENT(VIRTUALPREF)
                   .... ...1             UFO7CCCP     CONCURRENT(CACHEPREF)
        33     (21) BIT(8)           1   UFO8FLGS     EIGHT OPTIONS FLAGS BYTE
                   1... ....             UFOPMREQ     FCTOPPRCP(PRESMIRREQ)
                   .1.. ....             UFOPMPRE     FCTOPPRCP(PRESMIRPREF)
                   ..1. ....             UFOPMNON     FCTOPPRCP(PRESMIRNONE)
                   ...1 ....             UFOFCFRR     FCFASTREVERSERESTORE
                   .... 1...             UFOFCFVR     FCFULLVOLUMERELATION
                   .... .1..             UFO8RESY     FORCE RESET(YES)
                   .... ..1.             UFO8RESN     FORCE RESET(NO)
                   .... ...1             UFO8RESD     FORCE RESET(DUMP)
        34     (22) UNSIGNED         2   UFOMAXTM     MAXTIME NUMBER OF MINUTES
        36     (24) BIT(8)           1   UFO9FLGS     NINTH OPTIONS FLAGS BYTE
                   1... ....             UFOBRCLK     BCSRECOVER(LOCK)
                   .1.. ....             UFOBRCSU     BCSRECOVER(SUSPEND)
                   ..1. ....             UFOZCNON     ZCOMPRESS(NONE)
                   ...1 ....             UFOZCPRE     ZCOMPRESS(PREF)
                   .... 1...             UFOZCREQ     ZCOMPRESS(REQ)
                   .... .1..             UFOFCTXRC    FCTOXRCP
                   .... ..11             *            UNUSED
        37     (25) CHARACTER        8   UFO_ZWEBT_DD Web Toolkit Debug OutDD
        45     (2D) BIT(8)           1   UFO10FLG     TENTH OPTIONS FLAGS BYTE
                   1... ....             UFOSRMSM     DEBUG(SRMSG(MINIMAL))
                   .1.. ....             UFOSRMSS     DEBUG(SRMSG(SUMMARIZED))
                   ..1. ....             UFOSRMSD     DEBUG(SRMSG(DETAILED))
                   ...1 ....             UFOCLNNO     CLONE(NONE)
                   .... 1...             UFOCLNPF     CLONE(PREF)
                   .... .1..             UFOCLNRQ     CLONE(REQ)
                   .... ..11             *            RESERVED
        46     (2E) CHARACTER        2   *            RESERVED
        48     (30) CHARACTER        0   *

  OFFSET   OFFSET
  DECIMAL    HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
 ======== ======== ================  ============== ================================
        0      (0) STRUCTURE       32  UFOPARM        EXECUTE CARD
                                                      PARAMETER OPTION LIST. POINTED
                                                      TO BY THE ADDRESS OF UFOHDR +
                                                      UFBDYOFF. PRESENT IF ALL BITS
                                                      ARE OFF IN UFFUNCT
        0      (0) CHARACTER        1  UFSERPAR     SERIAL / PARALLEL
                   1... ....             UFFORSER     1 = FORCE TO SERIAL

  OFFSET   OFFSET
  DECIMAL    HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
 ======== ======== ================  ============== ================================
                   .1.. ....             UFDEFPAR     1 = DEFAULT TO PARALLEL.
                                                      IGNORED IF UFFORSER IS 1.
                   ..11 1111             *            RESERVED
        1      (1) CHARACTER        1  UFXAFLAG     CNTL FLAGS FOR XA MODE
                   1... ....             UFXABUFF     1=I/O BUF ABOVE 16M REQ
                   .1.. ....             UFAI31B      1=AI BUF ABOVE 16M REQD
                   ..1. ....             UFPZB64R     1=64 BIT REAL REQD
                   ...1 1111             *            RESERVED
        2      (2) CHARACTER        8  UFWKUNIT     WORKUNIT PARAMETER
       10      (A) CHARACTER        6  UFWKVOL      WORKVOL PARAMETER
       16     (10) CHARACTER        1  UFOPFLG1     RESERVED (R1H0 AND ABOVE)
                   1... ....             UFOUEXCP     1=USE EXCP FOR DUMP OUTPUT,
                                                      RESTORE INPUT AND COPYDUMP
                   .111 1111             *            RESERVED
       17     (11) UNSIGNED         1  UFOMXTSK     MAX PARALLEL TASKS
       18     (12) CHARACTER       14  *            RESERVED (R1H0 AND ABOVE)
       32     (20) CHARACTER        0  *
```

```
      OFFSET   OFFSET
     DECIMAL     HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
    ======== ======== ================  ============== ==============================
           0      (0) STRUCTURE        *  UFOVOL         VOLUME LIST HEADER. POINTED
                                                         TO BY UFVOLI@ AND UFVOLO@.
           0      (0) CHARACTER        4   UFOVHDR       HEADER
           0      (0) SIGNED           2    UFOVCNT      COUNT OF VOLUME LIST ENTRIES
           2      (2) CHARACTER        2    *            RESERVED
           4      (4) CHARACTER       20   UFOVENT(*)    VOLUME LIST ENTRY
           4      (4) ADDRESS          4    UFOVUCB@     ADDR OF UCB | 0
           8      (8) CHARACTER        8    UFODDNAM     DDNAME | BLANK
          16     (10) CHARACTER        6    UFOVOLID     VOLUME SERIAL | BLANK
          22     (16) CHARACTER        2    *            RESERVED
1     CROSS REFERENCE

                                                  HEX          HEX
      NAME                                        OFFSET      VALUE     LEVEL
      ====                                        ======     ========   =====
      ADRUFOB                                          0                    1
      UFAIFLGS                                        13                    3
      UFAIINV                                         13           80       4
      UFAI31B                                          1           40       3
      UFBDYOFF                                         6                    3
      UFBYFCCK                                        14           80       4
      UFBYFRVF                                        14           04       4
      UFDEFPAR                                         0           40       3
      UFDUOPTM                                         2                    2
      UFFCFRRT                                        14           08       4
      UFFIFILT                                        12           20       4
      UFFIFULL                                        12           80       4
      UFFIND                                          12                    3
      UFFIPART                                        12           40       4
      UFFIPRTV                                        12           10       4
      UFFLAGS                                         14                    3
      UFFLOGCL                                        12           08       4
      UFFORSER                                         0           80       3
      UFFPATH                                         12           04       4
      UFFREWCL                                        14           20       4
      UFFUBLSA                                        11           20       5
      UFFUCGCR                                        11           08       5
      UFFUCOMP                                        10           01       5
      UFFUCONS                                        11           04       5
      UFFUCONV                                        11           40       5
      UFFUCOPY                                        10           40       5
      UFFUCPYD                                        10           04       5
      UFFUDEF                                         10           80       5
      UFFUDUMP                                        10           10       5
      UFFUNCT                                         10                    3
      UFFUNCT1                                        10                    4
      UFFUNCT2                                        11                    4
      UFFUPRT                                         10           08       5
      UFFUREST                                        10           20       5
      UFFURLSE                                        11           80       5
      UFFUSREL                                        11           02       5
      UFID                                             0                    3
      UFIGCTNN                                        14           10       4
      UFLEN                                            4                    3
      UFNOIN                                          13           02       4
      UFNOOUT                                         13           01       4
      UFO_ZWEBT_DD                                    25                    2
      UFOALLMU                                        14           08       3
      UFOARBA                                          3           04       3
      UFOBK32K                                         3           08       3
      UFOBLDIX                                         3           20       3
      UFOBRCLK                                        24           80       3
      UFOBRCSU                                        24           40       3
      UFOCLNNO                                        2D           10       3
      UFOCLNPF                                        2D           08       3
      UFOCLNRQ                                        2D           04       3
      UFOCPDAY                                         A                    3
      UFOCPFRC                                         8           01       4
      UFOCVRBK                                        19           20       3
      UFODBSMS                                        1D           04       3
1     CROSS REFERENCE

                                                  HEX          HEX
      NAME                                        OFFSET      VALUE     LEVEL
      ====                                        ======     ========   =====
      UFODBTRC                                        1D           08       3
      UFODCOND                                        19           40       3
      UFODDNAM                                         8                    3
```

```
      UFODRBLK                                 14          10       3
      UFOERASE                                  3          80       3
      UFOFCFRR                                 21          10       3
      UFOFCFRZ                                 19          01       3
      UFOFCFVR                                 21          08       3
      UFOFCINC                                 1C          08       3
      UFOFCINL                                 1C          04       3
      UFOFCNC                                  19          10       3
      UFOFCN2C                                 19          02       3
      UFOFCSEF                                 20          80       3
      UFOFCTXRC                                24          04       3
      UFOFCVFN                                 1C          01       3
      UFOFCVFR                                 1C          02       3
      UFOFCVFY                                 1D          02       3
      UFOFCWD                                  19          08       3
      UFOFCWTR                                 1F                   2
      UFOFCWTS                                 1E                   2
      UFOFC2PP                                 19          04       3
      UFOFLEAV                                  3          01       3
      UFOFRAGI                                  4                   2
      UFOFRBLK                                 14          20       3
      UFOFRMSD                                 1D          20       3
      UFOFRMSM                                 1D          80       3
      UFOFRMSS                                 1D          40       3
      UFOFRNO                                  1C          20       3
      UFOFRPRF                                 1C          40       3
      UFOFRREQ                                 1C          80       3
      UFOFUNCT                                  0                   1
      UFOHCOMP                                 1D          10       3
      UFOHDR                                    0                   2
      UFOIACPY                                  3          40       3
      UFOINSOP                                  3                   2
      UFOMAXTM                                 22                   2
      UFOMKMV                                   3          02       3
      UFOMNSQT                                  C                   2
      UFOMNTUS                                 10                   2
      UFOMXTSK                                 11                   2
      UFONMGMT                                 14          01       3
      UFONOSMS                                 14          02       3
      UFOPARM                                   0                   1
      UFOPFLG1                                 10                   2
      UFOPMNON                                 21          20       3
      UFOPMPRE                                 21          40       3
      UFOPMREQ                                 21          80       3
      UFOPROCK                                 18                   2
      UFOPRSYS                                 18          40       3
      UFOPRUND                                 18          80       3
      UFORACLG                                  3          10       3
      UFORIOPC                                 1A                   2
      UFOSMALL                                  8          08       4
      UFOSMANY                                  8          04       4
      UFOSM1ST                                  8          02       4
1     CROSS REFERENCE

                                            HEX          HEX
      NAME                                  OFFSET       VALUE      LEVEL
      ====                                  ======     ========     =====
      UFOSPHER                                 14          04       3
      UFOSRMSD                                 2D          20       3
      UFOSRMSM                                 2D          80       3
      UFOSRMSS                                 2D          40       3
      UFOTGTAL                                 17                   2
      UFOTGTBL                                 17          20       3
      UFOTGTCY                                 17          80       3
      UFOTGTSR                                 17          10       3
      UFOTGTTR                                 17          40       3
      UFOT0REQ                                 19          80       3
      UFOUEXCP                                 10          80       3
      UFOVCANY                                  8          10       4
      UFOVCCUR                                  8          80       4
      UFOVCFLG                                  8                   3
      UFOVCNT                                   0                   3
      UFOVCNUM                                  8          20       4
      UFOVCSRC                                  8          40       4
      UFOVCVAL                                  9                   3
      UFOVENT                                   4                   2
      UFOVHDR                                   0                   2
      UFOVOL                                    0                   1
      UFOVOLID                                 10                   3
      UFOVUCB@                                  4                   3
      UFOWAITR                                 16                   2
      UFOWAITS                                 15                   2
```

```
      UFOZCNON                                  24          20       3
      UFOZCPRE                                  24          10       3
      UFOZCREQ                                  24          08       3
      UFO1ALD                                    0          04       3
      UFO1ALDL                                   0          02       3
      UFO1ALLE                                   0          01       3
      UFO1COMP                                   0          80       3
      UFO1CVOL                                   0          40       3
      UFO1FLGS                                   0                   2
      UFO1PURG                                   0          20       3
      UFO1RESE                                   0          10       3
      UFO1WRCK                                   0          08       3
      UFO10FLG                                  2D                   2
      UFO2CTLG                                   1          04       3
      UFO2DEL                                    1          08       3
      UFO2DYNQ                                   1          80       3
      UFO2ENQE                                   1          40       3
      UFO2ENQN                                   1          10       3
      UFO2ENQS                                   1          20       3
      UFO2FLGS                                   1                   2
      UFO2RECT                                   1          04       4
      UFO2UNC                                    1          02       3
      UFO2VALD                                   1          01       3
      UFO3FLGS                                  14                   2
      UFO3FORC                                  14          80       3
      UFO3REPL                                  14          40       3
      UFO4FLGS                                  19                   2
      UFO5FLGS                                  1C                   2
      UFO5REPU                                  1C          10       3
      UFO6FLGS                                  1D                   2
1     CROSS REFERENCE

                                              HEX          HEX
      NAME                                    OFFSET       VALUE     LEVEL
      ====                                    ======     ========    =====
      UFO7CCAP                                    20           04       3
      UFO7CCAR                                    20           20       3
      UFO7CCCP                                    20           01       3
      UFO7CCCR                                    20           08       3
      UFO7CCVP                                    20           02       3
      UFO7CCVR                                    20           10       3
      UFO7FLGS                                    20                    2
      UFO8FLGS                                    21                    2
      UFO8RESD                                    21           01       3
      UFO8RESN                                    21           02       3
      UFO8RESY                                    21           04       3
      UFO9FLGS                                    24                    2
      UFPZB64R                                     1           20       3
      UFSAFOK                                     14           40       4
      UFSERPAR                                     0                    2
      UFSTOP                                      13           10       4
      UFSYSIN                                     13           08       4
      UFSYSPR                                     13           04       4
      UFUIMAL                                     13           40       4
      UFUIMCH                                     13           20       4
      UFVOLI@                                      8                    3
      UFVOLO@                                      C                    3
      UFWKUNIT                                     2                    2
      UFWKVOL                                      A                    2
      UFXABUFF                                     1           80       3
      UFXAFLAG                                     1                    2
```

Registers on Return from the ADRUIXIT Exit

Your ADRUIXIT routine should issue a return code in register 15 to indicate the course of action for further processing by DFSMSdss.

ADRUIXIT Return Codes

Return Code
Description

0
No changes were made by this exit routine.

4
The parameter list has been modified.

8
Do not schedule this function (valid only for function command entry).

This exit is entered for the following reasons only, during processing of DFSMSdss:

- During program initialization. This is called the *PARM change* entry. (The UFFUNCT field contains all zeros.)
- Just before scheduling a function command. This is called the *function* entry. (The function type is indicated by the UFFUNCT field.)

Example of the ADRUIXIT Exit

This exit performs different actions, depending on the type of entry.

For a PARM change entry, this exit sets the return code to zero, indicating that the parameter list has not been modified and that DFSMSdss is to proceed normally.

For a function entry, this exit sets the return code to four, indicating that the parameter list has been modified. The exit also turns on the UFO1RESE and UFOSM1ST bits, which override the specification of the RESET and SELECTMULTI keywords, respectively. Both RESET processing and SELECTMULTI(FIRST) processing will be performed.

Note: Use Figure 51 on page 237 as a learning aid. It is not guaranteed to run on a particular system without some modification.

```
*********************************************************************
*                                                                   *
* Module Name       = ADRUIXIT                                      *
*                                                                   *
* Descriptive Name  = DFSMsdss Installation-Wide Exit Routine       *
*                                                                   *
* Function          = Controls certain DFSMSdss tasks or changes    *
*                     some defaults or specified options.           *
*                                                                   *
* Function Entry    = Sets return code to 4 indicating the parameter *
*                     list has been modified. Turns on the UFO1RESE  *
*                     and UFOSM1ST flags.                           *
*                                                                   *
* Parm Change Entry = Sets return code to 0 indicating the parameter *
*                     list has not been modified.                   *
*                                                                   *
*********************************************************************
*
ADRUIXIT CSECT
ADRUIXIT AMODE 31
ADRUIXIT RMODE 24
         STM   14,12,12(13)        Save registers
         USING ADRUIXIT,15         Addressability to ADRUIXIT
         USING ADRUFOB,1           Addressability to ADRUFO
         SR    2,2                 Zero register 2
         CH    2,UFFUNCT           Check entry type
         BNE   FUNCENT             Branch to function entry
         SR    3,3                 Parm change entry, save RC 0
         B     FINISH              Finished
FUNCENT  LH    2,UFBDYOFF          Get offset to UFOFUNCT
         AR    2,1                 Calculate address of UFOFUNCT
         USING UFOFUNCT,2          Addressability to UFOFUNCT
         OI    UFO1FLGS,UFO1RESE   Turn on  UFO1RESE (reset) bit
         NI    UFOVCFLG,X'FF'-(UFOSMALL+UFOSMANY+UFOSM1ST)
         OI    UFOVCFLG,UFOSM1ST   Turn on  UFOSM1ST bit
         LA    3,4                 Save return code 4
         DROP  1                   Done using 1 for ADRUFO
         DROP  2                   Done using 2 for UFOFUNCT
         DROP  15                  Done using 15 for ADRUIXIT
FINISH   LR    15,3                Set return code
         L     14,12(,13)          Restore register 14
         LM    0,12,20(13)         Restore registers 0 thru 12
         BR    14                  Return
         ADRUFO                    Include ADRUFO control block
         END
```

Figure 51. Sample Listing of ADRUIXIT

Reblock Installation Exit Routine (ADRREBLK)

You can use this exit to verify or change block size. The reblock installation exits can set a reblockable indicator that causes the data set to be automatically reblocked when the data set is copied by DFSMSdss or another program.

Installation-Supplied ADRREBLK Exit Routine

The reblock installation exits routine gets control when a sequential or partitioned data set is being processed during a data set COPY or RESTORE if either:

- You specify REBLOCK(dsn [...]) in the COPY or RESTORE statement, the options installation exits routine (ADRUIXIT) does not override it, and the data set being copied meets the filtering criteria passed in the REBLOCK sublist (fully- or partially-qualified data set names), or
- The options installation exits routine forces all data set COPY or RESTORE functions to use the reblock installation exits routine.

Notes:

- The reblock installation exits is not given control for RECFM=U data sets. However, PDS load module data sets are reblocked, if requested, by IEBCOPY COPYMOD.
- The reblock installation exit will *not* be called when processing zEDC compressed format sequential data set, or an encrypted-format data set.
- Reblocking requests are ignored for data sets being restored from an object storage cloud.

The installation options exit routine cannot change the REBLOCK sublist (fully or partially qualified data set names), but:

- It can prevent DFSMSdss from reblocking if you specified REBLOCK(dsn [...]).
- It can force DFSMSdss to reblock all sequential or partitioned data sets being copied (equivalent to REBLOCK(**)).
- If the reblockable indicator is on in the data set's Format-1 DSCB (DS1REBLK), DFSMSdss automatically reblocks the data set to a system-determined block size. This indicator is on if the system determined a block size when the data set was created. The reblock installation exits routine is not called.

The reblock installation exits routine supplied with DFSMSdss passes a return code of zero, which allows DFSMSdss to select a block size.

You can replace the DFSMSdss-supplied exit routine to override the DFSMSdss-selected block size. If the exit routine passes an invalid block size, the data set is not processed and message ADR453E is issued.

The name of the reblock installation exits routine must be ADRREBLK.

Registers on Entry to the ADRREBLK Exit

Register
Contents

1
Address of the ADRREBLK parameter list

13
Address of an 18-word save area

14
Return address

15
Address of the entry point to ADRREBLK

ADRRBLKB Parameter List

Register 1 contains the address of the ADRRBLKB parameter list. You can use the ADRRBLKB mapping macro to map this parameter list. These sections are shown in Table 82 on page 239.

First word
Pointer to a 140-byte work area that has the Format 1 DSCB from the first volume. See *z/OS DFSMSdfp Advanced Services* for the Format 1 DSCB description. If it is a sequential extended-format data set, the block size in the sixth word of the parameter list excludes the 32-byte suffix.

Second word
Pointer to the 6-byte volume serial number of the input volume. For a multivolume data set, this volume serial number is for the first input volume.

Third word
Pointer to the 4-byte device type field of the input volume. This is in the same format as the UCBTYP field in the UCB.

Fourth word
Pointer to the 6-byte volume serial number of the output volume. For a data set that spans multiple output volumes, the serial number is for the first volume.

Fifth word
Pointer to the 4-byte device type field of the output volume. Its format is similar to that for the input volume.

Sixth word
The high-order bit (bit 0) is on to indicate the end of the parameter list. If bit 1 is on, the system calculated the optimum block size for the target data set. If bit 1 is off, DFSMSdss calculated the block size. If bit 2 is off, the request is COPY, if bit 2 is on the request is RESTORE.

Bytes 2 and 3 of the sixth word contain the block size selected by DFSMSdss for the data set on the output volume. This field is in fixed binary.

Table 82. ADRRBLKB Parameter List

Offset	Length or Bit Pattern	Name	Description
0 (X'0')	24	ADRRBLKB	
0 (X'0')	4	RBDSCB@	Pointer to format-1 or format-8 DSCB
4 (X'4')	4	RBIVOL@	Pointer to input volume serial number
8 (X'8')	4	RBIDEV@	Pointer to input device type
12 (X'C')	4	RBOVOL@	Pointer to output volume serial number
16 (X'10')	4	RBODEV@	Pointer to output device type
20 (X'14')	1	RBEND	Last word in parameter list
	1...	RBENDLST	End of parameter list
	.1..	RBDASDCS	0 = DFSMSdss calculates 1 = System calculates
	..1.	RBRSTCPY	1 = called for RESTORE. 0 = called for COPY
	...x xxxx		Reserved
21 (X'15')	1		Reserved
22 (X'16')	2	RBLKSIZE	Selected block size

Registers on Return from the ADRREBLK Exit

Your ADRREBLK routine should issue a return code in register 15 to indicate the course of action for further processing by DFSMSdss.

ADRREBLK Return Codes

Return Code
Description

0
Block size not changed. Use the DFSMSdss-selected block size.

4
Block size has been changed by the exit; the new block size is indicated in last 2 bytes of word 6 of the parameter list.

8
Use the input block size (do not reblock).

12
System determined block size is used. In addition, the reblockable indicator in the Format 1 DSCB (DS1REBLK) is turned on.

Example of the ADRREBLK Exit

This exit sets the return code to eight, which results in reblock processing being bypassed.

Note: Use Figure 52 on page 240 as a learning aid. It is not guaranteed to run on a particular system without some modification.

```
*****************************************************
* ADRREBLK USER EXIT.                               *
*   SETS RETURN CODE TO 8 INDICATING THAT REBLOCKING *
*   SHOULD NOT BE PERFORMED.                        *
*****************************************************
ADRREBLK CSECT
ADRREBLK AMODE 31
ADRREBLK RMODE 24
          STM   14,12,12(13)          SAVE REGS IN PREVIOUS SAVEAREA
          LR    12,15                 ESTABLISH BASE REGISTER
          USING ADRREBLK,12           SET ADDRESSABILITY TO THE EXIT
          LM    14,12,12(13)          RESTORE OTHER REGISTERS
          LA    15,8                  SET RETURN CODE TO 8
          BR    14                    RETURN
          END
```

Figure 52. Sample Listing of ADRREBLK

Chapter 9. IEHINITT Dynamic Exits

This chaptertopic describes the setup, use, and invocation of exits associated with the IEHINITT utility.

Introduction

IEHINITT is a system utility used to place volume label sets onto any number of magnetic tapes mounted on one or more tape units. The IEHINITT Utility supports the REKEY function. You can use the REKEY function to replace the key label structure that is stored on a tape cartridge with a new key label structure. For details, see “The Re-keying Exit ” on page 247.

These exits use the dynamic exits service, CSVDYNEX, which is currently described in *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN*. By means of this facility a single exit, IEHINITT_EXIT has been defined for the INITT function, with two associated exit points, a pre-label and a post-label. For the REKEY function, a separate exit REKEY_EXIT has been defined with only one exit point, a re-keying exit point.

Using this same facility, installations can associate their own exit routines with these exits. See "Adding an Exit Routine to an Exit" in *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN* for details. The Dynamic Exits Facility allows multiple exit routines to be simultaneously defined to a single exit and, as such, coordination of processing between these exit routines is critical.

As mentioned previously, there are two exits points defined for the single exit IEHINITT_EXIT:

- The pre-label exit is invoked just prior to issuance of the labelling I/O. The intent of this exit is to allow the installation to indicate whether labelling of the volume is to occur, and, to a degree, the values which are to be used in doing the labelling.
- The post-label exit is invoked just after the issuance of the labelling I/O has or would have occurred. The intent of the post-label exit is to inform the installation exit routines of the results of the labelling request and associated pre-label exit processing.

The **same exit routine** is called at each exit point. This means that the exit must determine whether it is being entered for Pre or Post Label exit processing. A flag is set in the parameter list passed to the exit routine to indicate which exit point called the exit routine.

Although the Dynamic Exit Services Facility allows a degree of specification on order of call, that specification applies only to the last routine associated with an exit. So, in fact, there is no means of ensuring the order of call of routines. Therefore, all existing exit routines are always called for both Pre and Post label processing.

Exceptions to this rule:

- The Dynamic Exits Service itself fails for any reason:
  - If a failure occurs during pre-label processing, no more pre-label exit routines will be called and the volume will not be labeled. The post-label exit routine will be called.
  - If a failure occurs during post-label processing, no more post-label exit routines will be called.
- A failure occurs during internal processing not related to Dynamic Exits Services
- If a failure occurs before calling pre-label exit routines:
  - The pre-label exit routines will not be called
  - Labelling I/O will not take place
  - Post-label exit routines will be called
- If the operator replies S(kip) to message IEC701D:
  - No volume label will be read
  - Pre-label exit routines will not be called

– post-label exit routines will be called

General Programming Considerations

- The exit routines run in Key 5, with APF authorization. As with all subroutines, do not link edit installation exits with APF authorization.
- They are required to be AMODE 31
- The parameter list passed to the exit routines resides in non fetch protected, key 5 storage
- Although the service will not enforce reentrancy, it is recommended that the routine be reentrant

The Pre-Label Exit

Overview

The pre-label exit is taken just before labelling I/O is to occur. The intent of the pre-label exit is to allow the installation to indicate whether labelling I/O is to be issued, and, to a degree, the values which are to be used in labelling.

Registers on Entry

Register	Contents
1	address of the parameter list, INXPLIST, which is mapped by the IEHUEXIT macro
13	address of standard 72-byte save area
14	return address of the caller
15	entry point address of exit routine

Note: The 72 byte save area is provided solely for exit routines that expect to be able to save the registers on entry. Its contents are not used by the Dynamic Exits Service facility nor by IEHINITT.

Registers on Exit

There is no requirement to return any values in registers from the exit routines, so it is not necessary to restore any registers on exit from the exit routine.

There is a requirement to return a return and reason code. A default of 4 ('no vote') is primed in the parameter list on entry to the exit routine and need not be changed if not appropriate. If updating the return and reason codes is appropriate the fields to be changed are:

- INXRC, which contains the exit routine return code
- INXRSN, which contains the exit routine reason code

Return and Reason Code Values

The exit routines can indicate one of four return codes and one of two reason codes associated with return code 8.

Return Code	Reason Code	Meaning
0	0	Label the tape.
4	0	I don't care.
8	0	Do not label the tape.

Return Code	Reason Code	Meaning
8	4	Don't label this tape, mount a new tape and label it.

- **Return code 0 notes:**

  When return code 0 is returned by the exit routine, the following fields, which contain values to be used in labelling the volume, may be modified:

  – INXRQVOL, volume serial number
  – INXRQOWN, owner ID
  – INXRQACC, access code

  Changes to any other fields are ignored.
- **Return code 4 notes:** The exit routine return code is set to a value of 4 before each exit routine is called.

  Changes to any fields are ignored.
- **Return code 8, Reason code 0 notes:**

  It is important to note that once a no label indication is set, there is no way to override it. The volume will not be labeled.

  Changes to any fields are ignored.
- **Return code 8, Reason code 4 notes:**

  When this return code and reason code is returned, the value in INXRQVOL must be changed. A mount for the volume indicated in this field is issued. All subsequent pre-label exit routines are passed the original volume serial, not the remount volume serial.

  The post-label exit routines are not called for this volume.

  When the remount is successfully processed and the actual volume is mounted, the pre-label exit routines are called again and the volume serial that has been passed into the exit routines is now the new volume serial.

  Changes to any other fields are ignored.

Summary of Information Passed to the Pre-Label Exit Routines

The address of a parameter list mapped by macro IEHUEXIT is passed to the exit routines in Register 1. Some of the information in this parameter list can be changed and returned to IEHINITT for processing. Please review the macro for details on field names and values. Some of the fields passed in are:

- A one byte hex value indicating the function for which the routine is being invoked.

  Currently only pre-label and post-label functions are defined. A subfunction byte is passed in, but is not currently defined for this function.
- 4 bytes of flags that indicate the status of data passed into the routine.

  A variety of conditions concerning the result of attempting to read the existing labels on the volume are indicated here. Flags are set for such things as whether the labels were successfully read, whether the VOL1 label information passed in was constructed from sense information when allowed by VOLNSNS specification in the active DEVSUPxx PARMLIB member, whether an I/O error was encountered when trying to read the labels, in addition to other conditions.

  All exit routines should see the same flag settings in these bytes.
- 4 bytes of flags that indicate the cumulative results of processing the exit routines and any reasons for not labelling the volume.

  There are a variety of reasons that a volume will not be labeled. Things such as:

  – A failure in the CSVDYNEX service,
  – An ABEND in an exit routine,

- The operator responding with S(kip) to the IEHINITT mount message,
- An exit routine indicated that the volume is not to be labeled.
- Invalid characters are specified for the original or modified volume serial number, owner ID, and/or access code.
- A conflict in information returned by an exit routine

  Please see "Conflict Processing" on page 245 for details.

Each pre-label exit routine sees the cumulative result of all previous exit routine processing. By cumulative we mean that all results of previous exit routine processing will be indicated in these flags. So, for example, if a conflict is detected after calling the third exit routine, all subsequent exit routines will see the conflict flag, IXWONTL, set. Another example would be that once a routine indicates that the volume is not to be labeled, all subsequent routines will see the IXCANTL flag set.

All post-label exit routines will see the final cumulative result of all pre-label exit routine processing in these flags.

- Other informational data that is provided includes the:
  - VOL1 and HDR1 labels if available,
  - Labels that were read where VOL1 or HDR1 labels were expected but the format was not as expected,
  - Volume serial returned from a LACS VERIFY call if different from the read volume serial,
  - SAF and RACROUTE AUTH return and reason codes

    When the volume label is not read for whatever reason, the requested volume serial number is used as the internal volume serial in all authorization checking.

    A call to SAF/RACROUTE AUTH is made before the pre-label exits are invoked. The results of this call are passed, uninspected, to the exit routines. If an exit routine returns an RC=0, indicating the volume is to be labelled, it is assumed that the exit reviewed the SAF/RACROUTE return and reason codes and that it is allowing labelling regardless of those values.

    If there are no exit routines associated with this exit, or if all routines return an RC=4, then the results of the SAF/RACROUTE call will be inspected and labelling will be performed only if SAF/RACROUTE authorization is indicated.
  - A pointer to the UCB
  - Sense data that was generated as a result of any non-recoverable I/O error produced while trying to read the VOL1 or HDR1 labels
- Fields INXRQVOL (volume serial), INXRQOWN (owner ID), and INXRQACC (access code) are provided as input, but can be modified by the exit routines under the following conditions:
  - A return code 0 is returned, in which case the currently mounted volume will be labeled with the value in these fields. This applies to all three fields.
  - A return code 8, reason code 4 is returned, in which case the value in INXRQVOL must be changed or a conflict condition will be indicated and the volume will not be labelled. The owner ID and access code fields may also be changed.

  Caution in using this facility should be exercised when more that one exit routine is active. A conflict condition can result if one exit routine specifies a changed field that does not agree with the value returned by another exit routine, or if one exit indicates to label the tape and another indicates to label the volume with a new value.

  - INXRQVOL

    A 6 character field containing the volume serial with which the volume is to be labeled. If more than 1 routine returns a modified volume serial, the values must be the same or a conflict is indicated and the volume is not labeled.

    This field is also used to indicate the new volume serial that is to be mounted when a remount request is returned, RC=8, RSN=4.

Note: All pre-label exit routines will see the originally specified volume serial, not the modified volume serial.

– INXRQOWN

The owner ID with which the volume is to be labeled. If more than one exit routine modifies this field, the modified value must be the same or a conflict will be indicated and the volume will not be labeled.

Note: All pre-label exit routines will see the originally specified owner ID, not the modified owner ID.

– INXRQACC

The access code value (if the request is for an ANSI label) with which the volume is to be labeled. If more than one exit routine modifies this field, the modified value must be the same or a conflict will be indicated and the volume will not be labeled.

Note: All pre-label exit routines will see the originally specified access code, not the modified access code.

Conflict Processing

In an environment where a single exit routine exists, the result of calling the routine is straight forward. However, the existence of multiple exit routines is permissible. For this case extensive conflict checking is done to ensure that the exits do not return conflicting information. Whenever a conflict is detected the volume will not be labeled.

Typically a single exit routine will take ownership of the volume and indicate what the result of the labelling request is to be. Ownership is implied by returning a return code 0 or 8. Routines that do not take ownership of the volume should return a return code of 4.

Once again, in a multiple exit routine environment, more than one routine can take ownership of a volume and the volume will be labeled as long as there is no conflict in the information returned from any of the routines. A conflict will be indicated when:

- A return code of 0 and a return code of 8 are returned for the same volume or more than one exit returned an RC=8, but the reason codes are not consistent.
- A return code of 0 is returned by more than one exit and one of the exits specifies a new value for a modifiable field while another exit specifies a different value or does not specify a change.
- A remount was requested by an exit routine but no new volume serial was specified.
- A remount was requested by more than one exit routine and the modified volume serial values don't match.

Special Considerations for NUMBTAPE Processing:

When NUMBTAPE is specified on the IEHINITT control statement the current restriction is that the original volume serial must be all numeric. This restriction will continue to apply, however, the modified volume serial need not adhere to this restriction.

Note: Iincrementation of the numeric volume serial value is performed based on the value of the originally specified volume serial. For example, if a volume serial of 002000 is specified on the INITT card and exit processing results in the volume serial being modified to 050000, the next volume serial to be processed will be 002001, not 050001.

Labelling Write Protected Volumes

When a write-protected volume is mounted, pre-label exit routines will be passed the label information just read. If exit routines all agree to label that volume, IEHINITT will attempt to label the volume which will fail. post-label exit routines will be called with LBLRFAIL on. Afterwards, message IEC701D will be issued again. Pre-label and post-label exit routines will again be called as before.

READLBL Related Support

IEHINITT supports a keyword in the IEHINITT control statement,

```
READLDL = YES | NO
```

- When READLBL=YES (default)
  - The volume label of the mounted volume is read before calling the pre-label exit. The information gathered from the read volume label is subsequently passed to the pre-label and post-label exits.
- When READLBL=NO
  - The volume label of the mounted volume is not read. INXLBNRD (label-not-read) in INXFLAG2 is set.
  - No volume label information related to the mounted volume is passed to the pre or post-label exits.
  - The requested volume serial is used in LACS verify and authorization checking.
  - In particular,
    - No flags related to the mounted volume label are set nor should they be interrogated. These flags include: INXVRDOK, INXHRDOK, INXVLSNS, INXNOLBL, INXVRDTM, INXBLANK, INXIOERR, INXSTDAL, INXVH1OK.
    - The values of VOL1, INXRDVOL,and HDR1, are not read from the mounted volume. Therefore, INXVOL1, INXRDVOL, and INXHDR1 are zero.
    - For pre-label exit, INXSNSOK and INXSENSE refer to the result of I/O issued to read the mounted volume label.

      If READLBL=NO was specified, INXSNSOK and INXSENSE is never set and is not interrogated.

      If READLBL=YES was specified or is defaulted, INXSNSOK and INXSENSE may contain valid values if an I/O error occurred on the I/O.
    - For the post-label exit, INXSNSOK and INXSENSE refer to the result of I/O issued to write the volume label, hence they are independent of READLBL values and may be non-zero when passed to the post-label exits if an I/O error occurred.

The Post-Label Exit

Overview

The intent of the post-label exit is to inform the installation exit routines of the results of the labelling request and associated pre-label exit processing.

No action is currently taken based on the return code/reason code returned by the routines.

Registers on Entry

Register	Contents
1	address of the parameter list, INXPLIST, which is mapped by the IEHUEXIT macro
13	address of standard 72 byte save area
14	return address of the caller
15	entry point address of exit routine

Note: The 72 byte save area is provided solely in case the exit routine expects to be able to save the registers on entry. Its contents are not used by the Dynamic Exits Service Facility nor by IEHINITT.

Registers on Exit

There is no requirement to return any values in registers from the exit routines, so it is not necessary to restore any registers on exit from the exit routine.

Return and Reason Code Values

The exit routine return code is set to a value 4 each time an exit is called.

There is currently no requirement to return a return and reason code other than the default values of return code 4, reason code 0.

In an effort to enforce coexistence with any future enhancements to post-label processing, the return and reason code will be inspected and if anything other than RC=4, RSN=0 is returned a warning message will be issued.

Return code 4, reason code 0 indicates that the routines takes no vote on the processing to be done for this exit point. As stated before, there is currently no action taken for this exit, but if this changes in the future a return code of 4/reason code of 0 will allow compatible processing.

Summary of Information Passed to the Post-Label Exit Routines

All the pre-label exit processing information as it existed after the last call to the pre-label exit routines will be passed to the post-label exit routines. This allows all post-label routines to inspect the results of the pre-label processing.

One exception to this is the sense information that may have been passed to the pre-label exits. Any sense passed to the post-label exit will be sense generated by the labelling I/O, if any was done.

In addition, post-label specific information passed in will be:

- VOL1 label with which the volume was labeled, if labelling occurred.

  if no labelling I/O was issued, this field will be zero and LBLRNATM will be on.
- one byte of flags indicating the results of the labelling I/O.

Output

If the labelling I/O completes successfully, no new messages nor new return codes are returned in the IEHINITT return code.

If the labelling I/O is not attempted because of pre-label exit processing, a return code 20 is returned (24, if the SYSPRINT card is mi/ssing).

If the labelling I/O is not attempted, or the label I/O fails due to errors related to Pre- or Post-label exit processing IEHINITT messages are shown.

The Re-keying Exit

Overview

The re-keying exit is invoked once for each tape volume after the volume has successfully been re-keyed. The intent is to inform the installation exit routines of the new key label information.

Registers on Entry

Register	Contents
1	address of the parameter list, INXPLIST, which is mapped by the IEHUEXIT macro
13	address of standard 72 byte save area

Register	Contents
14	return address of the caller
15	entry point address of exit routine

Note: The 72 byte save area is provided solely in case the exit routine expects to be able to save the registers on entry. Its contents are not used by the Dynamic Exits Service Facility nor by IEHINITT.

Registers on Exit

There is no requirement to return any values in registers from the exit routines, so it is not necessary to restore any registers on exit from the exit routine.

Return and Reason Code Values

The exit routine return code is set to a value 4 each time an exit is called.

There is currently no requirement to return a return and reason code other than the default values of return code 4, reason code 0.

Summary of Information Passed to the Re-keying Exit Routines

A one byte hex value indicating the function for which the routine is being invoked.

INXFUNC = INXREKEY Exit is called for Re-keying processing.

The volume serial and new key label information are passed to the exit routines.

```
INXRKVOL Six-byte volume serial number
INXKYLB1 64-byte key label 1
INXENCD1 One-byte key label 1 encoding mechanism
INXKYLB2 64-byte key label 2
INXENCD2 One-byte key label 2 encoding mechanism
```

Output

There is currently no requirement to return any return and reason or any values from the exit routines. All returned values are ignored.

Appendix A. Accessibility

Accessible publications for this product are offered through IBM Documentation (www.ibm.com/docs/en/zos).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the Contact the z/OS team web page (www.ibm.com/systems/campaignmail/z/zos/contact_z) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Notices

This information was developed for products and services that are offered in the USA or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for IBM Documentation. Use of hyperlinks in other output formats of this information is at your own risk.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Site Counsel
2455 South Road

Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com®/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMSdfp, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those

products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming interface information

This publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/OS DFSMS.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at Copyright and Trademark information (www.ibm.com/legal/copytrade.shtml).

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, Other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, and service names, which may be trademarks or service marks of others.

Index

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Glossary

This glossary defines technical terms and abbreviations used in DFSMS documentation. If you do not find the term you are looking for, refer to the index of the appropriate DFSMS manual.

The following cross-reference is used in this glossary:

See:
This refers the reader to (a) a related term, (b) a term that is the expanded form of an abbreviation or acronym, or (c) a synonym or more preferred term.

A

ABE
Abnormal-end appendage, an appendage of EXCP.

ACB
Access method control block.

access method services
A multifunction service program that manages both VSAM and non-VSAM data sets and integrated catalog facility catalogs. It defines data sets and allocates space for VSAM data sets and ICF catalogs. It converts indexed-sequential data sets to key-sequenced data sets; modifies data set attributes in the catalog; reorganizes data sets; facilitates data portability among operating systems; creates backup copies of data sets; helps make inaccessible data sets accessible; lists the records of data sets and catalogs; defines and builds alternate indexes; and converts CVOLs and to integrated catalog facility catalogs.

ACS
See *Automatic class selection (ACS)*.

ACS installation exit
User-written code, run after an ACS routine, that provides capabilities beyond the scope of the ACS routine.

ACS interface routine
This calls an ACS routine from an ACS installation-exit routine.

ADDR
Addressed processing or addressed.

ADR
Same as ADDR.

aggregate backup
The process of copying an aggregate group and recovery instructions so that a collection of data sets can be recovered later as a group.

aggregate group
A collection of related data sets and control information that have been pooled to meet a defined backup or recovery strategy.

AIX
Alternate index.

APF
See *Authorized program facility (APF)*.

ASCB
Address space control block

ASCII
American National Standard Code for Information Interchange.

ASI
Asynchronous interrupt.

authorized program facility (APF)
A facility that permits identification of programs authorized to use restricted functions.

automatic class selection (ACS) routine
A sequence of instructions for having the system assign data class, storage class, management class, and storage group for data sets, and storage class, management class, and storage groups for objects.

automatic backup
(1) In DFSMShsm, the process of automatically copying data sets from primary storage volumes or migration volumes to backup volumes.

(2) In OAM, the process of automatically copying a primary copy of an object from disk, or an optical or tape volume to a backup volume contained in an object backup storage group.

B

backup
The process of creating a copy of a data set or object to be used in case of accidental loss.

backup control data set (BCDS)
In DFSMShsm, a VSAM key-sequenced data set that contains information about backup versions of data sets, backup volumes, dump volumes, and volumes under control of the backup and dump functions of DFSMShsm.

backup-while-open (BWO)
This makes a backup copy of a data set while the data set is open for update. The backup copy can contain partial updates.

basic catalog structure (BCS)
The name of the catalog structure in the catalog environment.

basic format
The format of a data set that has a data set name type (DSNTYPE) of BASIC. A basic format data set is a sequential data set that is specified to be neither large format nor extended format. The size of a basic format data set cannot exceed 65 535 tracks on each volume.

BDW
Block descriptor word.

binder
The DFSMS program that processes the output of language translators and compilers into an executable program (load module or program object). It replaces the linkage editor and batch loader in z/OS.

BPI
Bytes per inch.

BUFC
Buffer control block.

C

CA
Control area.

candidate volume
A direct-access storage volume that has been defined in an ICF catalog as a VSAM volume; VSAM can automatically allocate space on this volume, as needed.

CDS
See *Control data set (CDS)*.

CI
Control interval. Also compatibility interface.

class
See *SMS class*.

class transition
An event that brings about change to an object's service-level criteria, causing OAM to invoke ACS routines to assign a new storage class or management class to the object.

compress
(1) To reduce the amount of storage required for a given data set by having the system replace identical words or phrases with a shorter token associated with the word or phrase.

(2) To reclaim the unused and unavailable space in a partitioned data set that results from deleting or modifying members by moving all unused space to the end of the data set.

compressed format
A particular type of extended-format data set specified with the (COMPACTION) parameter of data class. VSAM can compress individual records in a compressed-format data set. SAM can compress individual blocks in a compressed-format data set. See *compress*.

construct
One of the following: data class, storage class, management class, storage group, aggregate group, base configuration.

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. With respect to DFSMSrmm, a VSAM key-sequenced data set (KSDS) containing the complete inventory of your removable media library.

convert in place
See in-place conversion.

CVAF
Common VTOC access facility.

CVT
Communication vector table.

D

DADSM
Direct access device space management.

DASD calculation services (DCS)
A function of DFSMS.

DASD volume
A DASD space identified by a common label and accessed by a set of related addresses. See also *volume, primary storage, migration level 1, migration level 2*.

data class
A collection of allocation and space attributes, defined by the storage administrator, that are used to create a data set.

data set
In DFSMS, the major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access. In z/OS non-UNIX environments, the terms *data set* and *file* are generally equivalent and sometimes are used interchangeably. See also *file*. In z/OS UNIX environments, the terms *data set* and *file* have quite distinct meanings.

DCB
Data control block.

DCBE
Data control block extension.

DCS
DASD calculation services.

DECB
Data event control block.

default management class
Part of the SMS base configuration, it identifies the management class that should be used for system-managed data sets that do not have a management class assigned.

device category
A storage device classification used by SMS. The device categories are as follows SMS-managed DASD, SMS-managed tape, non-SMS-managed DASD non-SMS-managed tape.

DFSMSdfp
A DFSMS functional component or base element of z/OS, that provides functions for storage management, data management, program management, device management, and distributed data access.

DFSMSdss
A DFSMS functional component or base element of z/OS, used to copy, move, dump, and restore data sets and volumes.

DFSMShsm
A DFSMS functional component or base element of z/OS, used for backing up and recovering data, and managing space on volumes in the storage hierarchy.

DFSMShsm-managed volume
(1) A primary storage volume, which is defined to DFSMShsm but which does not belong to a storage group.

(2) A volume in a storage group, which is using DFSMShsm automatic dump, migration, or backup services. Contrast with *system-managed volume, DFSMSrmm-managed volume*.

DFSMS
See *Data Facility Storage Management Subsystem.*

DFSMSrmm
A DFSMS functional component or base element of z/OS, that manages removable media.

DFSMSrmm-managed volume
A tape volume that is defined to DFSMSrmm. Contrast with *system-managed volume, DFSMShsm-managed volume*.

DFSORT
Data Facility Sort.

DSAB
Data set association block.

DSCB
Data set control block.

dummy storage group
A type of storage group that contains the serial numbers of volumes no longer connected to a system. Dummy storage groups allow existing JCL to function without having to be changed. See also *storage group*.

E

EBCDIC
Extended binary-coded decimal interchange code.

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.

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.

EOM
End-of-module.

EP
Entry point.

EXCD
Exceptional conditions.

expiration
(1) The process by which data sets or objects are identified for deletion because their expiration date or retention period has passed. On DASD, data sets and objects are deleted. On tape, when all data sets have reached their expiration date, the tape volume is available for reuse.

(2) In DFSMSrmm, all volumes have an expiration date or retention period set for them either by vital record specification policy, by user-specified JCL when writing a data set to the volume, or by an installation default. When a volume reaches its expiration date or retention period, it becomes eligible for release.

extended format
The format of a data set that has a data set name type (DSNTYPE) of EXTENDED. The data set is structured logically the same as a data set that is not in extended format but the physical format is different. Data sets in extended format can be striped or compressed. Data in an extended format VSAM KSDS can be compressed. See also *striped data set, compressed format*.

F

file
A collection of information treated as a unit. In non-z/OS UNIX environments, the terms *data set* and *file* are generally equivalent and are sometimes used interchangeably. See also *data set*.

filtering
The process of selecting data sets based on specified criteria. These criteria consist of fully or partially-qualified data set names or of certain data set characteristics.

FREEMAIN
An assembler language macro instruction that releases one area of virtual storage that had previously been allocated as a result of a GETMAIN macro instruction.

G

GEN
Generic key search.

generation
One member of a generation data group.

GETMAIN
An assembler language macro instruction that is used to allocate an area of virtual storage.

group
(1) With respect to partitioned data sets, a member and the member's aliases that exist in a PDS or PDSE, or in an unloaded PDSE.

(2) A collection of users who can share access authorities for protected resources.

H

HA
Home address.

I

improved data recording capability (IDRC)
A recording mode that can increase the effective cartridge data capacity and the effective data rate when enabled and used. IDRC is always enabled on the 3490E Magnetic Tape Subsystem.

in-place conversion
The process of bringing a volume and the data sets it contains under the control of SMS without data movement, using DFSMSdss.

installation exit
The means specifically described in an IBM software product's documentation by which an IBM software product may be modified by a customer's system programmers to change or extend the functions of the IBM software product. Such modifications consist of exit routines written to replace an existing module of an IBM software product, or to add one or more modules or subroutines to

an IBM software product for the purpose of modifying (including extending) the functions of the IBM software product. Contrast with *user exit routine.*

installation exit routine
A routine written by a system programmer to take control at an installation exit of an IBM software product.

Interactive Storage Management Facility (ISMF)
The interactive interface of DFSMS that allows users and storage administrators access to the storage management functions.

Interactive System Productivity Facility (ISPF)
An IBM licensed program that serves as a full-screen editor and dialogue manager. Used for writing application programs, it provides a means of generating standard screen panels and interactive dialogues between the application programmer and terminal user.

interval migration
In DFSMShsm, automatic migration that occurs when a threshold level of occupancy is reached or exceeded on a DFSMShsm-managed volume, during a specified time interval. Data sets are moved from the volume, largest eligible data set first, until the low threshold of occupancy is reached.

IOB
Input/output block.

J

JFCB
Job file control block.

JFCBE
Job file control block extension for 3800 printer.

JSCB
Job step control block.

K

KB
Kilobyte (equals two to the tenth power bytes, or 1024).

L

large format
The format of a data set that has a data set name type (DSNTYPE) of LARGE. A large format data set has the same characteristics as a sequential (non-extended format) data set, but its size on each volume can exceed 65 535 tracks. There is no minimum size requirement for a large format data set.

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.

locate
Pertains to functions that do not change the status of a catalog; that is, read-only operations are performed.

LPALIB
Link pack area library.

LRI
Logical record interface.

M

MACRF
Macro reference.

management class
A collection of management attributes, defined by the storage administrator, used to control the release of allocated but unused space; to control the retention, migration, and backup of data sets; to control the retention and backup of aggregate groups, and to control the retention, backup, and class transition of objects.

MBBCCHHR
Absolute disk address. (Module#, bin#, cylinder#, head#, record#).

MCD
MCDS data set record. See migration control data set.

memory
As used in this information, a synonym for the private address space in virtual storage.

migration
The process of moving unused data to lower cost storage in order to make space for high-availability data. If you wish to use the data set, it must be recalled. See also *migration level 1, migration level 2*.

migration control data set (MCDS)
In DFSMShsm, a VSAM key-sequenced data set that contains statistics records, control records, user records, records for data sets that have migrated, and records for volumes under migration control of DFSMShsm.

migration level 1
DFSMShsm-owned DASD volumes that contain data sets migrated from primary storage volumes. The data can be compressed. See also *storage hierarchy*. Contrast with *primary storage, migration level 2*.

migration level 2
DFSMShsm-owned tape or DASD volumes that contain data sets migrated from primary storage volumes or from migration level 1 volumes. The data can be compressed. See also *storage hierarchy*. Contrast with *primary storage, migration level 1*.

MVS/DFP
An IBM licensed program which is the base for the Storage Management Subsystem.

MWE
Management work element

N

NSI
Next sequential instruction.

NUP
No update.

O

OAM
See *Object access method*.

OAM Storage Management Component (OSMC)
In the Object Access Method, the component that determines where objects should be stored, manages object movement within the object storage hierarchy, and manages expiration attributes based on the installation storage management policy.

object
A named byte stream having no specific format or record orientation.

object access method (OAM)
An access method that provides storage, retrieval, and storage hierarchy management for objects and provides storage and retrieval management for tape volumes contained in system-managed libraries.

object backup storage group
A type of storage group that contains optical or tape volumes used for backup copies of objects. See also *storage group*.

object storage group
A type of storage group that contains objects on DASD, tape, or optical volumes. See also *storage group*.

O/C/EOV
Open, close, end-of-volume.

offline control data set (OCDS)
In DFSMShsm, a VSAM key-sequenced set that contains information about tape backup volumes and tape migration level 2 volumes.

optical volume
Storage space on an optical disk, identified by a volume label. See also *volume*.

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.

z/OS
z/OS is a network computing-ready, integrated operating system consisting of more than 50 base elements and integrated optional features delivered as a configured, tested system.

OSMC
OAM storage management component.

OSR
Object storage and retrieval.

P

partitioned data set (PDS)
A data set on direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

partitioned data set extended (PDSE)
A data set that contains an indexed directory and members that are similar to the directory and members of partitioned data sets. A PDSE can be used instead of a partitioned data set.

PDAB
Parallel data access block.

PDSCB
Partial data set control block.

PDSE
Partitioned data set extended.

performance
(1) A measurement of the amount of work a product can produce with a given amount of resources.

(2) In a system-managed storage environment, a measurement of effective data processing speed with respect to objectives set by the storage administrator. Performance is largely determined by throughput, response time, and system availability.

permanent data set
A user-named data set that is normally retained for longer than the duration of a job or interactive session. Contrast with *temporary data set*.

PLH
Placeholder list.

pool storage group
A type of storage group that contains system-managed DASD volumes. Pool storage groups allow groups of volumes to be managed as a single entity. See also *storage group*.

PR
Pseudo register.

primary storage
A DASD volume available to users for data allocation. The volumes in primary storage are called primary volumes. See also *storage hierarchy*. Contrast with *migration level 1, migration level 2*.

PSL
Page save list.

Q

QCT
Queue control block

R

R0
Record zero.

RB
Request block.

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.

RDW
Record descriptor word.

record zero (R0)
Track capacity record on a DASD device.

recovery
The process of rebuilding data after it has been damaged or destroyed, often by using a backup copy of the data or by reapplying transactions recorded in a log.

relative track and record address (TTR)
Relative track and record address on a direct-access device, where TT represents two bytes specifying the track relative to the beginning of the data set, and R is one byte specifying the record on that track.

RETURN
An assembler language macro instruction that is used to return control to the calling CSECT, and to signal normal or abnormal termination of the returning CSECT.

Rn
General purpose register n.

RPS
Rotational position sensing.

RTN
Routine.

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.

SCRA
Catalog recovery area in system storage.

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.

SDUMP
System dump.

search limit
The track following the last track that should actually be searched in a data set.

SEQ
Sequential or sequential processing.

SETL
Set lower limit of sequential retrieval

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.

SIO
Start I/O.

small-data-set-packing data set
In DFSMShsm, a VSAM key-sequenced data set allocated on a migration level 1 volume and containing small data sets that have been migrated.

SKP
Skip sequential or skip sequential processing.

SMS
Storage Management Subsystem or System Managed Storage.

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.

sphere
The collection of base cluster, alternate indexes, and upgrade alternate indexes opened to process one or more related paths.

SRA
Sphere record area.

storage administrator
A person in the data processing center who is responsible for defining, implementing, and maintaining storage management policies.

storage class
A collection of storage attributes that identify performance goals and availability requirements, defined by the storage administrator, used to select a device that can meet those goals and requirements.

storage group
A collection of storage volumes and attributes, defined by the storage administrator. The collections can be a group of DASD volumes or tape volumes, or a group of DASD, optical, or tape volumes treated as a single object storage hierarchy. See also *pool storage group, tape storage group, object storage group, object backup storage group*, and *dummy storage group*.

storage hierarchy
An arrangement of storage devices with different speeds and capacities. The levels of the storage hierarchy include main storage (memory, DASD cache), primary storage (DASD containing uncompressed data), migration level 1 (DASD containing data in a space-saving format), and migration level 2 (tape cartridges containing data in a space-saving format). See also *primary storage, migration level 1, migration level 2, object storage hierarchy*.

Storage Management Subsystem (SMS)
A DFSMS facility used to automate and centralize the management of storage. Using SMS, a storage administrator describes data allocation characteristics, performance and availability goals, backup and retention requirements, and storage requirements to the system through data class, storage class, management class, storage group, and ACS routine definitions.

string
The part of a control block structure built around a placeholder (PLH) that allows VSAM to keep track of one position in the data set that the control block structure describes.

stripe
In DFSMS, the portion of a striped data set, such as an extended format data set, that resides on one volume. The records in that portion are not always logically consecutive. The system distributes

records among the stripes such that the volumes can be read from or written to simultaneously to gain better performance. Whether it is striped is not apparent to the application program.

striping
A software implementation of a disk array that distributes a data set across multiple volumes to improve performance.

striped data set
In DFSMS, an extended-format data set consisting of two or more stripes. SMS determines the number of stripes to use based on the value of the SUSTAINED DATA RATE in the storage class. Striped data sets can take advantage of the sequential data striping access technique. See *stripe, striping*.

SVCLIB
Supervisor call library.

SVRB
Supervisor request block.

SVT
Supervisor vector table.

SWA
Scheduler work area.

system data
The data sets required by z/OS or its subsystems for initialization and control.

system-managed data set
A data set that has been assigned a storage class.

system-managed storage
Storage managed by the Storage Management Subsystem. SMS attempts to deliver required services for availability, performance, and space to applications. See also *system-managed storage environment*.

DFSMS environment
An environment that helps automate and centralize the management of storage. This is achieved through a combination of hardware, software, and policies. In the DFSMS environment for z/OS, this function is provided by DFSMS, DFSORT, and RACF. See also *system-managed storage*.

system-managed tape library
A collection of tape volumes and tape devices, defined in the tape configuration database. A system-managed tape library can be automated or manual. See also *tape library.*

system-managed volume
A DASD, optical, or tape volume that belongs to a storage group. Contrast with *DFSMShsm-managed volume, DFSMSrmm-managed volume*.

system programmer
A programmer who plans, generates, maintains, extends, and controls the use of an operating system and applications with the aim of improving overall productivity of an installation.

T

tape library
A set of equipment and facilities that support an installation's tape environment. This can include tape storage racks, a set of tape drives, and a set of related tape volumes mounted on those drives. See also *system-managed tape library*.

Tape Library Dataserver
A hardware device that maintains the tape inventory that is associated with a set of tape drives. An automated tape library dataserver also manages the mounting, removal, and storage of tapes.

tape storage group
A type of storage group that contains system-managed private tape volumes. The tape storage group definition specifies the system-managed tape libraries that can contain tape volumes. See also *storage group*.

tape volume
A tape volume is the recording space on a single tape cartridge or reel. See also *volume*.

TCB
Task control block or Trusted computer base (in Trusted Computer Security Criteria).

TIOT
Task I/O table.

temporary data set
An uncataloged data set whose name begins with & or &&, that is normally used only for the duration of a job or interactive session. Contrast with *permanent data set*.

track overflow
A user-specified option that allows a record whose space requirements exceed the space remaining on the track to be partially written on that track and completed on the next track. MVS no longer supports this hardware feature.

transfer control (XCTL)
An assembler language macro that causes control to be passed to a specified entry point.

TTR
Relative track record address.

U

UCB
Unit control block.

uncatalog
To remove the catalog entry of a data set from a catalog.

unit control block (UCB)
A data area used by MVS for device allocation and for controlling input/output, (I/O) operations.

UPD
Update mode, or data modify.

user exit
A point in an IBM-supplied program at which a user exit routine may be given control.

user exit routine
A user-written routine that receives control at predefined user exit points. User exit routines can be written in assembler or a high-level language.

USVR
User security-verification routine.

V

VIR
VTOC index record.

VIXM
VTOC index map.

VMDS
VTOC map of DSCBs.

volume
The storage space on DASD, tape, or optical devices, which is identified by a volume label. See also *DASD volume*, *optical volume*, and *tape volume*.

VRP
VSAM resource pool.

VSAM volume data set (VVDS)
A data set that describes the characteristics of VSAM and system-managed data sets residing on a given DASD volume; part of a catalog. See also *basic catalog structure*.

VSRT
VSAM shared resource table.

VVDS
VSAM volume data set.

VVR
VSAM volume record.

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.

X

XCTL
See *XCTL (Transfer control)*.

XCTL (transfer control)
An assembler language macro that causes control to be passed to a specified entry point without return expected.

Product Number: 5650-ZOS

SC23-6850-50