z/OS
2.5

DFSMSdfp Advanced Services

SC23-6861-50

Note

Before using this information and the product it supports, read the information in “Notices” on page 483.

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: 2023-03-24

Contents

Figures

Tables

About this document

This document is intended to help system programmers modify and extend the data management capabilities of the operating system and for programmers to write advanced application programs.

For information about the accessibility features of z/OS, for users who have a physical disability, see Appendix D, “Accessibility,” on page 481.

Required product knowledge

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

- Assembler language
- Standard program linkage conventions
- The utility programs IEHLIST and IEHPROGM
- Data management access methods and macro instructions
- The storage management functions provided by DFSMSdss™ and DFSMShsm™. DFSMSdss moves data from one device to another, backs up and recovers data sets, and reduces free-space fragmentation on DASD volumes. DFSMShsm manages storage by migrating and recalling data sets through a hierarchy of storage devices.

  To learn about physical storage administration, see *z/OS DFSMSdfp Storage Administration*.

Referenced documents

The following publications are referenced in this document:

Table 1. Referenced Publications

Publication Title	Order Number
IBM High Level Assembler/MVS & VM & VSE Programmer's Guide	SC26-4941
IBM High Level Assembler/MVS & VM & VSE Language Reference	SC26-4940
z/OS MVS Programming: Assembler Services Guide	SA23-1368
z/OS MVS Programming: Assembler Services Reference ABE-HSP	SA23-1369
z/OS MVS Programming: Authorized Assembler Services Guide	SA23-1371
z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN	SA23-1372
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 Planning: Global Resource Serialization	SA23-1389
z/OS MVS Program Management: User's Guide and Reference	SA23-1393
z/OS MVS Program Management: Advanced Facilities	SA23-1392
z/Architecture Principles of Operation	SA22-7832
IBM 2821 Control Unit Component Description	GA24-3312
IBM 3203 Printer Component Description and Operator's Guide	GA33-1515

Table 1. Referenced Publications (continued)

Publication Title	Order Number
IBM 3211 Printer, 3216 Interchangeable Train Cartridge, and 3811 Printer Control Unit Component Description and Operator's Guide	GA24-3543
IBM 3262 Model 5 Printer Product Description	GA24-3936
IBM 3800 Printing Subsystem Programmer's Guide	GC26-3846
IBM 3800 Printing Subsystem Model 3 Programmer's Guide: Compatibility	SH35-0051
IBM 3800 Printing Subsystem Models 3 and 8 Programmer's Guide	SH35-0061
3900 Advanced Function Printer Product Description	GA32-0135
IBM 4245 Printer Model 1 Component Description and Operator's Guide	GA33-1541
IBM 4248 Printer Model 1 Description	GA24-3927
IBM 6262 Printer Model 014 Product Description	GA24-4134
IBM 6262 Printer Model 014 User's Guide	GA24-4132
IBM 6262 Printer Print Band Manual	GA24-4131
Device Support Facilities (ICKDSF) User's Guide and Reference	GC35-0033
z/OS MVS JCL Reference	SA23-1385
z/OS MVS JCL User's Guide	SA23-1386
z/OS JES2 Initialization and Tuning Guide	SA32-0991
z/OS JES3 Initialization and Tuning Guide	SA32-1003
z/OS DFSMS Access Method Services Commands	SC23-6846
z/OS DFSMSdfp Diagnosis	SC23-6863
z/OS DFSMS Installation Exits	SC23-6850
z/OS DFSMS Macro Instructions for Data Sets	SC23-6852
z/OS DFSMS Using Data Sets	SC23-6855
z/OS DFSMS Using Magnetic Tapes	SC23-6858
z/OS DFSMSdfp Utilities	SC23-6864
z/OS DFSMS Implementing System-Managed Storage	SC23-6849
z/OS MVS Initialization and Tuning Guide	SA23-1379
z/OS HCD Planning	GA32-0907
z/OS Security Server RACF Security Administrator's Guide	SA23-2289
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

Table 1. Referenced Publications (continued)

Publication Title	Order Number
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 TSO/E Command Reference	SA32-0975

z/OS information

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

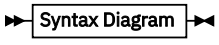
When possible, this information uses cross document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see *z/OS Information Roadmap*.

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

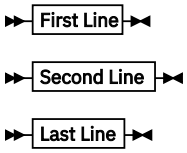
How to Read Syntax Diagrams

Throughout this library, diagrams are used to illustrate the programming syntax. Keyword parameters are parameters that follow the positional parameters. Unless otherwise stated, keyword parameters can be coded in any order. The following list tells you how to interpret the syntax diagrams:

- Read the diagrams from left-to-right, top-to-bottom, following the main path line. Each diagram begins on the left with double arrowheads and ends on the right with two arrowheads facing each other.

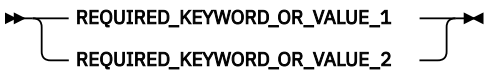
- If a diagram is longer than one line, each line to be continued ends with a single arrowhead and the next line begins with a single arrowhead.

- Required keywords and values appear on the main path line. You must code required keywords and values.

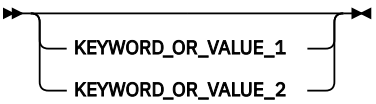
  If several mutually exclusive required keywords or values exist, they are stacked vertically in alphanumeric order.

- Optional keywords and values appear below the main path line. You can choose not to code optional keywords and values.

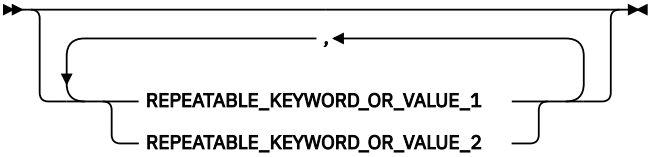
  

  If several mutually exclusive optional keywords or values exist, they are stacked vertically in alphanumeric order below the main path line.



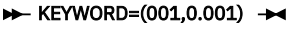
- An arrow returning to the left above a keyword or value on the main path line means that the keyword or value can be repeated. The comma means that each keyword or value must be separated from the next by a comma.

- An arrow returning to the left above a group of keywords or values means more than one can be selected, or a single one can be repeated.

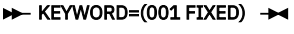


- A word in all uppercase is a keyword or value you must spell exactly as shown. In this example, you must code ***KEYWORD***.

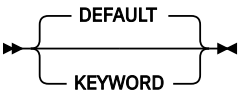
  If a keyword or value can be abbreviated, the abbreviation is discussed in the text associated with the syntax diagram.
- If a diagram shows a character that is not alphanumeric (such as parentheses, periods, commas, and equal signs), you must code the character as part of the syntax. In this example, you must code ***KEYWORD=(001,0.001)***.



- If a diagram shows a blank space, you must code the blank space as part of the syntax. In this example, you must code KEYWORD=(001 FIXED).



- Default keywords and values appear above the main path line. If you omit the keyword or value entirely, the default is used.

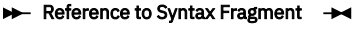


- A word in all lowercase italics is a *variable*. Where you see a variable in the syntax, you must replace it with one of its allowable names or values, as defined in the text.

- References to syntax notes appear as numbers enclosed in parentheses above the line. Do not code the parentheses or the number.

  Notes:

  1 An example of a syntax note.
- Some diagrams contain *syntax fragments*, which serve to break up diagrams that are too long, too complex, or too repetitious. Syntax fragment names are in mixed case and are shown in the diagram and in the heading of the fragment. The fragment is placed below the main diagram.



Syntax Fragment

Address and Register Conventions

The notation used to code an operand appears in the following format:

symbol
The operand can be any valid assembler-language symbol.

(*0*)
General register 0 can be used as an operand. When used as an operand in a macro instruction, the register must be specified as the decimal number 0 enclosed in parentheses as shown.

(*1*)
General register 1 can be used as an operand. When used as an operand in a macro instruction, the register must be specified as the decimal number 1 enclosed in parentheses as shown. When you use register 1, the instruction that loads it is not included in the macro expansion.

(*2-12*)
The operand specified can be any of the general registers 2 through 12. All registers as operands must be coded in parentheses; for example, if register 3 is coded, it is coded as (3). A register from 2 through 12 can be coded as a decimal number, symbol (equated to a decimal number), or an expression that results in a value of 2 through 12.

RX-Type Address
The operand can be specified as any valid assembler-language RX-type address as shown in the following examples:

Name	Operation	Operand
ALPHA1	L	1,39(4,10)
ALPHA2	L	REG1,39(4,TEN)
BETA1	L	2,ZETA(4)
BETA2	L	REG2,ZETA(REG4)
GAMMA1	L	2,ZETA
GAMMA2	L	REG2,ZETA
GAMMA3	L	2,=F'1000'
LAMBDA1	L	3,20(,5)

Both ALPHA instructions specify explicit addresses; REG1 and TEN have been defined as absolute symbols. Both BETA instructions specify implied addresses, and both use index registers. ZETA is a relocatable symbol. Indexing is omitted from the GAMMA instructions. GAMMA1 and GAMMA2 specify implied addresses. The second operand of GAMMA3 is a literal. LAMBDA1 specifies an explicit address with no indexing.

A-Type Address
The operand can be specified as any address that can be written as a valid assembler-language A-type address constant. An A-type address constant can be written as an absolute value, a relocatable symbol, or relocatable expression. Operands requiring an A-type address are inserted into an A-type address constant during the macro expansion process. For more details about A-type address constants, see *High Level Assembler/MVS & VM & VSE Language Reference.*

absexp
The operand can be an absolute value or expression. An absolute expression can be an absolute term or an arithmetic combination of absolute terms. An absolute term can be a nonrelocatable symbol, a self-defining term, or the length attribute reference. For more details about absolute expressions, see *High Level Assembler/MVS & VM & VSE Language Reference.*

relexp

The operand can be a relocatable symbol or expression. If the program containing a relocatable symbol or expression is relocated *n* bytes away from its originally assigned area of storage, the value of a relocatable symbol or expression changes by *n*. For more details about relocatable symbols and expressions, see *High Level Assembler/MVS & VM & VSE Language Reference.*

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.

February 2023 refresh

- "Data Facilities Area (DFA) Fields" on page 438 was updated to include DFAMMMULTVOL. (APAR OA336878, which also applies to z/OS V2R4)

August 2022 refresh

- Return codes 178 (X'B8') and 184 (X'B8') are added to Table 12 on page 47.
- LSPACE Examples is updated. For more information, see "LSPACE Examples" on page 37
- New fields are added to MAPPING FOR THE DEVICE PERFORMANCE STATISTICS OUTPUT BUFFER and MAPPING FOR THE SUBSYSTEM GET OUTPUT BUFFER FOR SYNCH I/O LINK STATISTICS. For more information, see "Passed Argument List -- SSGARGL" on page 395.

June 2022 refresh

- For APAR OA62225, a new field, DFAGDGLimitMax, is added. For more information, see "Data Facilities Area (DFA) Fields" on page 438.

April 2022 refresh

- For APAR OA61850, a new bit, DFAOCEExits, is added. For more information, see "Data Facilities Area (DFA) Fields" on page 438.

Prior to April 2022 refresh

- New return and reason codes are added for the IGGENC macro. See Table 58 on page 224.

Changed

The following content is changed.

November 2022 refresh

The PRLDDIG field is added to "Description" on page 46.

August 2022 refresh

- "Determining Whether zHPF is Supported for a Device" on page 165 is updated.

February 2022 refresh

- For APAR OA61235, Chapter 10, "Using the DESERV Exit," on page 341 is updated to indicate there are two calls to the DESERV EXIT, and the results of function requested is available after the second call.

January 2022 refresh

- For APAR OA62660, “Passed Argument List -- SSGARGL” on page 395 SSGSCHST is updated.

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

The following changes are made for z/OS V2R4. The most recent changes are listed at the top of each section.

New

December 2020 refresh

DFAMMMULTVOL

- With APAR OA56622, offset 7 was added to Table 50 on page 216 and DFASEQENCRYPT was updated for offset 82(52) in “Data Facilities Area (DFA) Fields” on page 438.
- “Data Facilities Area (DFA) Fields” on page 438 was updated to include DFAMMDUALLOG.

November 2020 refresh

- With APAR OA56622, added “Encrypting and Decrypting with the IGGENC Macro” on page 212.
- “Data Facilities Area (DFA) Fields” on page 438 was updated to include the TCTCOMPRESSION keyword.
- With APAR OA59541, the Reserved field is updated for Table 15 on page 57.

Changed

June 2021 refresh

The parameter_list_address—RX-type address is updated. See “PURGE macro specification” on page 289 for more information.

December 2020 refresh

Updated notes for “Reading a DSCB by Data Set Name” on page 37.

August 2020 refresh

- With APAR OA59629, the SSGARGL parameter list passed to IDCSS01 has been updated to reflect new fields. For more information, see “Passed Argument List -- SSGARGL” on page 395.

Deleted

No content was removed from this information.

Summary of changes for z/OS Version 2 Release 3 (V2R3)

The following changes are made for z/OS Version 2 Release 2 (V2R3).

New

- “Data Facilities Area (DFA) Fields” on page 438 has been updated.
- “Data Control Block Extension (DCBE) Fields” on page 186 has been updated.
- OA55706 has the following updates: “CVAFDIR Macro Overview and Specification” on page 71, “CVCLID: Specify the address of a 4-byte field ” on page 73, “PLISTVER: Specify version to be generated” on page 74
- “VALIDATE: Validate Modified fields in a DSCB” on page 73, “Generating a CVPL (CVAF Parameter List)” on page 57, and “Buffer List Entry” on page 60 have been updated.
- “LSPACE—Standard Form” on page 21, “LSPACE-Execute Form” on page 22, “LSPACE—List Form” on page 26, and “LSPACE–DSECT Form” on page 29 have been updated.

- "Reading and Modifying a Job File Control Block (RDJFCB Macro)" on page 273 has been updated.

Changed

- "Input/Output Block (IOB) Fields" on page 189 has been updated.
- OA52670, DEVTYPE XTIOT keyword has been added. See "UCBLIST or INFOLIST Type of Call" on page 254.
- References to 3380 devices have been changed to 3390 track capacity in following examples:
  - Scratch and CAMLST Macro Specification "Example" on page 135
  - Rename and CAMLST Macro Specification "Example" on page 140
  - Cataloging a Non-VSAM Data Set (CATALOG and CAMLST CAT) "Example" on page 149
  - "TRKCALC Macro Examples" on page 300

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 xxxi.

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 DFSMSdfp Advanced Services, SC23-6861-50
- 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.

Chapter 1. Using the Volume Table of Contents

This information is intended to help you to use system macros to access and modify the volume table of contents (VTOC) and VTOC index. The direct access device storage management (DADSM) routines control space allocation on direct access volumes through the VTOC for a volume and through the VTOC index if one exists. A storage administrator uses the ICKDSF utility to build these structures. See “Creating the VTOC and VTOC Index” on page 131.

VTOC Components

The VTOC is a data set that describes the contents of the direct access volume on which it resides. It is a contiguous data set; that is, it resides in a single extent on the volume and starts after cylinder 0, track 0 and before track 65,535. A VTOC's address is located in the VOLVTOC field of the standard volume label (see Figure 1 on page 1). The volume label is described in *z/OS DFSMS Using Data Sets*. A VTOC consists of complete tracks. Some other System z® operating systems support VTOCs that are compatible with z/OS. Some of them allow the VTOC to begin in the middle of a track. Normal z/OS operations do not support such a VTOC. However if you do not create or extend data sets from z/OS in such a VTOC, you may be able to read data sets on z/OS that were created on the other operating system.

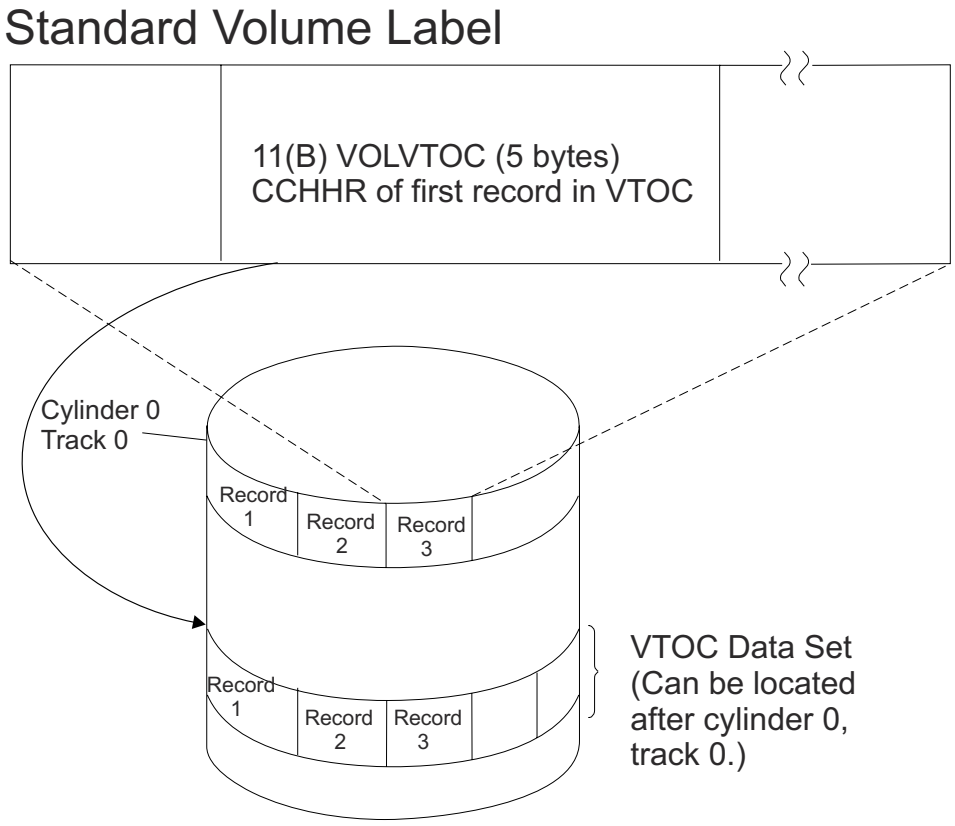


Figure 1. Locating the volume table of contents

The VTOC is composed of 140-byte[1] data set control blocks (DSCBs) that correspond either to a data set currently residing on the volume, or to contiguous, unassigned tracks on the volume.

A special type of data is in a hierarchical file system (HFS) data set. Each one contains a UNIX type of file system that is compliant with the IEEE POSIX standard and OSF XPG/4.2 standard. Certain linear

[1] The 140 bytes are divided into a 44-byte key portion followed by a 96-byte data portion. You can refer to the logical 140-byte DSCB or to either of its parts.

data sets contain z/OS file system (zFS) file systems. They also meet these standards. Each data set containing an HFS or zFS file system can contain UNIX files and directories belonging to multiple users. You can access the files through z/OS UNIX System Services, BSAM, QSAM, and VSAM. See *z/OS DFSMS Using Data Sets*. DSCBs for data sets describe their characteristics and location. DSCBs for contiguous, unassigned tracks indicate their location.

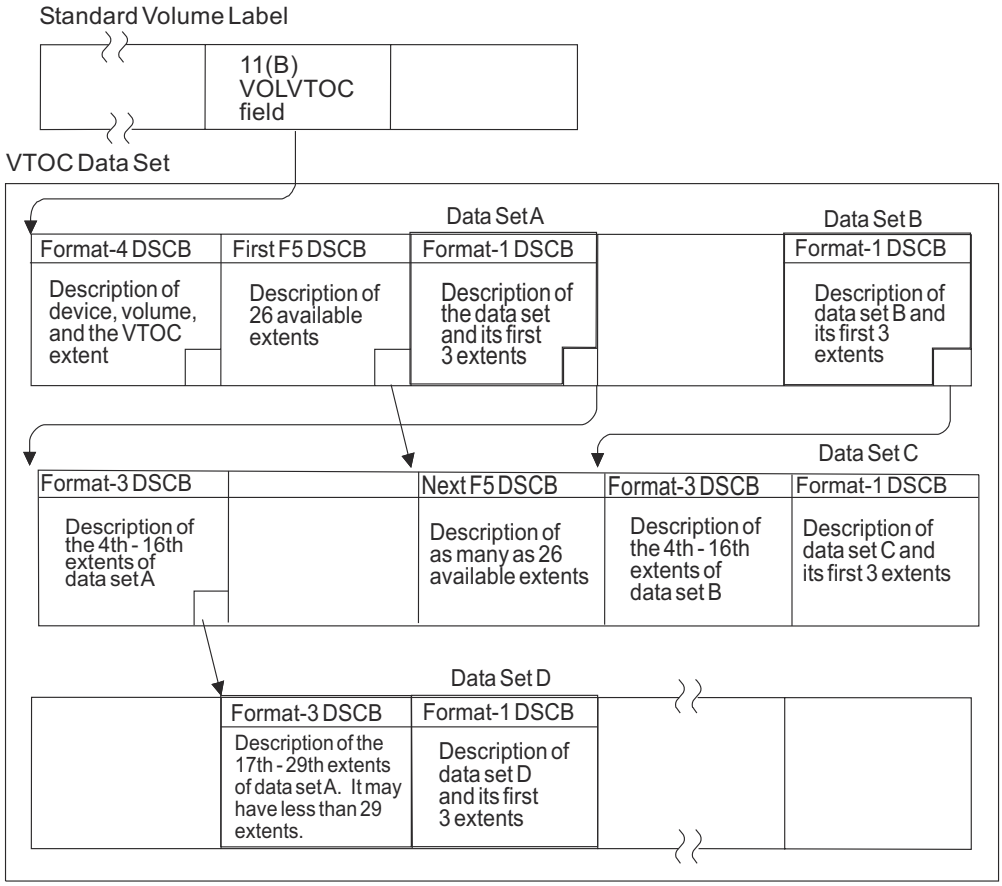
Data Set Control Block (DSCB) Types

The VTOC contains several kinds of DSCBs. This section describes what the DSCBs are used for, how many exist on a volume, and how to locate them. The DSCB layouts are shown in Table 2 on page 5 through Table 5 on page 15.

The first record in every VTOC is the VTOC DSCB (format-4). The record describes the device the volume resides on, the volume attributes, and the size and contents of the VTOC data set. The next DSCB in the VTOC data set is a free-space DSCB (format-5) even if the free space is described by format-7 DSCBs. The third and subsequent DSCBs in the VTOC can occur in any order.

One or more DSCBs in the VTOC define a data set on each volume on which the data set resides. The number of DSCBs needed to define a data set is determined by the number of extents that the data set occupies and by whether it has a format 9 DSCB. One or more format-3 DSCBs are required for data sets with more than three extents.

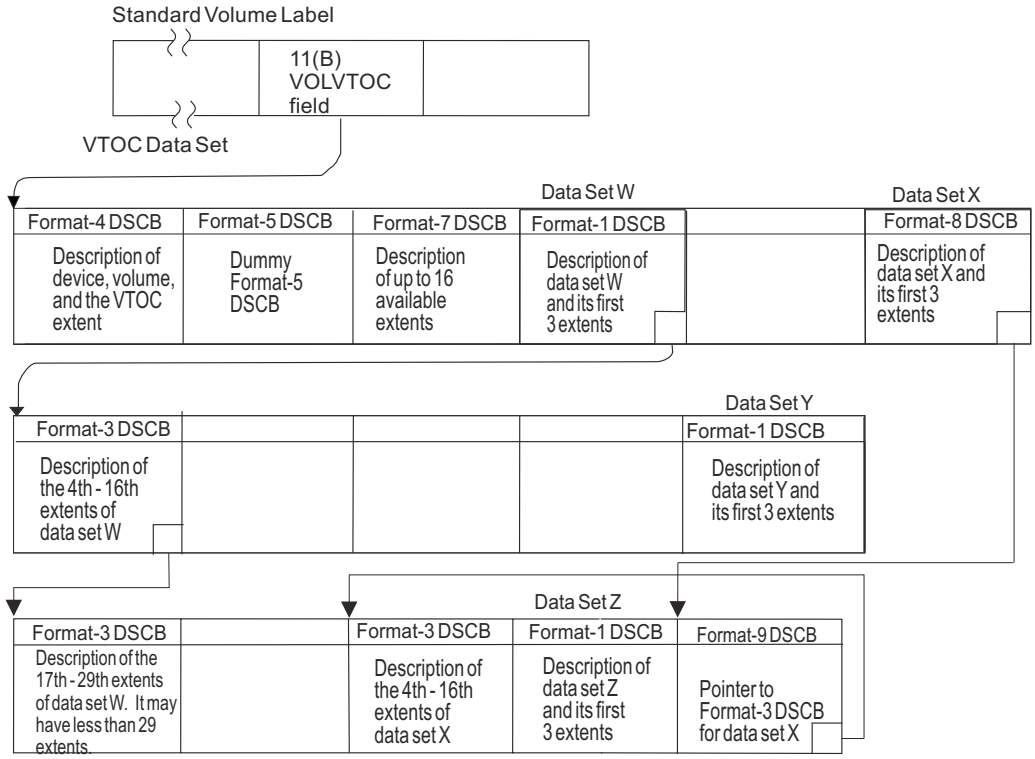
Figure 2 on page 2 shows a VTOC and the DSCBs needed to define four data sets. Data set A (in Figure 2 on page 2) has 29 extents, so it cannot be a basic format, direct or partitioned (PDS) data set. Because it has so many extents, it requires a format-1 DSCB and two format-3 DSCBs. Data set B has 16 extents and therefore requires both a format-1 and a format-3 DSCB. Data sets C and D have three or fewer extents and need only a format-1 DSCB. Data sets B, C, and D could be any type of data set.



Note: Empty boxes in the VTOC data set represent free VTOC Records (Format-0 DSCBs)

Figure 2. Contents of VTOC - DSCBs describing data sets on a volume that has no VTOC index

Figure 3 on page 3 shows a VTOC and the DSCBs needed to define four data sets on an extended address volume. Data set W (in Figure 3 on page 3) has 29 extents, so it cannot be a basic format, direct or partitioned (PDS) data set. Because it has so many extents, it requires a format-1 DSCB and two format-3 DSCBs. Data set X has 16 extents and since it is an EAS eligible data set it requires a format-8 and a format 9-DSCB, plus an additional format-3 DSCB. Data sets Y and Z have three or fewer extents and need only a format-1 DSCB. Data sets X, Y, and Z could be any type of data set.



Note: Empty boxes in the VTOC data set represent free VTOC Records (Format-0 DSCBs)

Figure 3. Contents of VTOC on an extended address volume - DSCBs that describe data sets on a volume that has no VTOC index

The mapping macro for the format-1, format-2, format-3, format-4, format-5, format-8, and format-9 DSCBs is IECSDSL1. Code positional parameters to specify which formats of DSCB to map. For example, the following maps format-1 and format-3:

```
        IECSDSL1 1,3
```

The macro does not generate a DSECT statement. This makes it easier to embed it in your own DSECT or CSECT. You can use a technique such as the following to get separate DSECT statements:

```
F1AREA   DSECT
         IECSDSL1 1
F3AREA   DSECT
         IECSDSL1 3
```

The first symbol generated for each format is of the form IECSDSLn, where n is the number of the format.

Format-0 DSCB

Name

Free VTOC Record

Function

Describes an unused record in the VTOC (contains 140 bytes of binary zeros). To delete a DSCB from the VTOC, a format-0 DSCB is written over it.

How Many

One for every unused 140-byte record on the VTOC. The DS4DSREC field of the format-4 DSCB is a count of the number of format-0 DSCBs on the VTOC. This field is not maintained for an indexed VTOC.

How Found

Search on key equal to X'00'.

Format-1 and Format-8 DSCBs

Note: The fields in the format 1 DSCB and the format 8 DSCB are almost identical. The small differences are noted in the field descriptions.

Name

Identifier

Function

Describes the first three extents and other information about a data set.

How Many

One for each data set on the volume, except the VTOC.

How Found

Use the OBTAIN macro or a CVAF macro.

Determining the Type of Data Set: To learn the type of a data set, test the DS1DSORG field. Except for the DS1DSGU bit, only one bit should be on. If no bit is on, the data set is not standard and the system generally treats it like DSORG=PS. The explanation for all DS1DSORG bits being off might be:

- The user did not specify the DSORG option and no program has written data in the data set.
- The data set was created by a program that used EXCP or EXCPVR.
- The data set was created on another operating system.

All of the above conditions are normal. Normally after a program writes data, the DS1DSORG field will have a bit on. z/OS currently supports the following types of data set:

- Physical sequential (DSORG=PS). There are three types:
  - Large format if the DS1LARGE bit is on.
  - Extended format if the DS1STRP bit is on. An indicator in the catalog entry specifies whether it is striped.
  - Basic format if neither of the above two bits is on.
- Direct (DSORG=DA).
- Partitioned (DSORG=PO). This can signify one of three types of data set:
  - PDS (partitioned data set) if the DS1PDSE and DS1PDSEX bits are off.
  - PDSE (partitioned data set extended) if the DS1PDSE bit is on and DS1PDSEX bit is off.
  - HFS (hierarchical file system) if the DS1PDSE and DS1PDSEX bits are on.

  Other combinations of these three bits are invalid.

- VSAM (AMORG). You can use CSI, catalog search interface, to determine the type of VSAM data set. If it is a linear data set, it might contain a DB2® tablespace or a z/OS file system (zFS) or something else.

Table 2 on page 5 shows the contents of a format-1 or format-8 DSCB.

The DS1REFD field:

1. For a VSAM data set, if the date in the FMT-1 DSCB's DS1REFD field is earlier than the current date, OPEN updates the field with the current date.
2. For a multivolume VSAM data set, OPEN updates the DS1REFD field only for the first volume of the data component of the base cluster.
3. For a non-VSAM multivolume data set that is not SMS managed, OPEN updates the DS1REFD field on the first volume OPEN selected to use.
4. For a non-VSAM multivolume SMS-managed data set that is not extended format, OPEN updates the DS1REFD field on the first volume OPEN selected to use, as well as the first volume of the data set.
5. For an extended format single-striped non-VSAM data set, OPEN updates the DS1REFD field on the first volume OPEN selected to use, as well as the first volume of the data set.
6. For an extended format multi-striped non-VSAM data set, OPEN updates the DS1REFD fields on all volumes of the data set.

Table 2. DSCB Format-1 or Format-8. DSCB Format-1 or Format-8

Offset Dec (Hex)	Type or Bit Mask	Length	Name	Description
0(X'0')	Character	44	DS1DSNAM	Data set name.
44(X'2C')	Character	1	DS1FMTID	Format Identifier.
			DS1IDC	X'F1'. This is a format-1 DSCB.
			DS8IDC	X'F8'. This is a format-8 DSCB.
45(X'2D')	Character	6	DS1DSSN	Data set serial number (identifies the first or only volume containing the data set/space).
51(X'33')	Unsigned	2	DS1VOLSQ	Volume sequence number.
53(X'35')	Character	3	DS1CREDT	Creation date ('YDD'), discontinuous binary. Add 1900 and the value in the Y byte to determine the year. For VSAM data sets that are not SMS-managed, the expiration date is in the catalog.
56(X'38')	Character	3	DS1EXPDT	Expiration date ('YDD'), discontinuous binary. Add 1900 and the value in the Y byte to determine the year.
59(X'3B')	Unsigned	1	DS1NOEPV	Number of extents on volume.
60(X'3C')	Unsigned	1	DS1NOBDB	Number of bytes used in last directory block.
61(X'3D')	Bitstring	1	DS1FLAG1	Flags byte
	1		DS1COMPR	Compressible format data set (DS1STRP is also 1).
	. 1		DS1CPOIT	Checkpointed data set.
	. . 1		DS1EXPBY	VSE expiration date specified by retention period (not currently used in z/OS)
	. . . 1		DS1RECAL	Data set recalled.
	. . . . 1 . . .		DS1LARGE	Large format data set.
	. . . . . 1 . .		DS1ENCRP	Access method encrypted data set.

Table 2. DSCB Format-1 or Format-8. DSCB Format-1 or Format-8 *(continued)*

Offset Dec (Hex)	Type or Bit Mask	Length	Name	Description
	11		DS1EATTR	Extended attribute setting as specified on the allocation request.(EATTR=) • If 0, EATTR has not been specified. For VSAM data sets, the default behavior is equivalent to EATTR=OPT. For non-VSAM data sets, the default behavior is equivalent to EATTR=NO. • If 1, EATTR=NO has been specified. The data set cannot have extended attributes (format 8 and 9 DSCBs) or optionally reside in EAS. • If 2, EATTR=OPT has been specified. The data set can have extended attributes and optionally reside in EAS. This is the default behavior for VSAM data sets. • If 3, Not Used, EATTR treated as not specified.
62(X'3E')	Character	13	DS1SYSCD	System code.
75(X'4B')	Character	3	DS1REFD	Date last referenced ('YDD' or zero, if not maintained). Add 1900 and the value in the Y byte to determine the year.
78(X'4E')	Bitstring	1	DS1SMSFG	System managed storage indicators.
	1.......		DS1SMSDS	System managed data set. IEHLIST displays this bit as the letter "S".
	.1......		DS1SMSUC	Uncataloged system managed data set (the VTOC index is an uncataloged system managed data set as are all temporary data sets on system managed volumes). IEHLIST displays this bit as the letter "U".
	..1.....		DS1REBLK	System determined the block size and data set can be reblocked (you or the system can reblock the data set). IEHLIST displays this bit as the letter "R".
	...1....		DS1CRSDB	DADSM created original block size and data set has not been opened for output. IEHLIST displays this bit as the letter "B".
	....1...		DS1PDSE	Data set is a PDSE or HFS data set (DS1PDSEX is also 1 for HFS). IEHLIST displays this bit as the letter "I" for a PDSE. IEHLIST displays a "?" when it finds an invalid combination of bits.
	.....1..		DS1STRP	Sequential extended-format data set. IEHLIST displays this bit as the letter "E". IEHLIST displays a "?" when it finds an invalid combination of bits.
	......1.		DS1PDSEX	HFS data set (DS1PDSE must also be 1) IEHLIST displays this bit as the letter "H".
	1		DS1DSAE	Extended attributes exist in the catalog entry.
79(X'4F')	Character	3	DS1SCEXT	Secondary space extension. Valid only when DS1EXT is on (see offset 94(X'5E')).
79(X'4F')	Bitstring	1	DS1SCXTF	Secondary space extension flag byte–only one of the first 4 bits is on.
	1.......		DS1SCAVB	If 1, DS1SCXTV is the original block length. If 0, DS1SCXTV is the average record length.
	.1......		DS1SCMB	If 1, DS1SCXTV is in megabytes.
	..1.....		DS1SCKB	If 1, DS1SCXTV is in kilobytes.

Table 2. DSCB Format-1 or Format-8. DSCB Format-1 or Format-8 *(continued)*

Offset Dec (Hex)	Type or Bit Mask	Length	Name	Description
	. . . 1		DS1SCUB	If 1, DS1SCXTV is in bytes.
	. . . . 1 . . .		DS1SCCP1	If 1, DS1SCXTV has been compacted by a factor of 256.
	. . . . . 1 . .		DS1SCCP2	If 1, DS1SCXTV has been compacted by a factor of 65,536.
80(X'50')	Unsigned	2	DS1SCXTV	Secondary space extension value for average record length or average block length.
82(X'52')	Bitstring	2	DS1DSORG	Data set organization.
			First byte of DS1DSORG	
	1000 000x		DS1DSGIS	Indexed sequential organization.
	0100 000x		DS1DSGPS	Physical sequential organization.
	0010 000x		DS1DSGDA	Direct organization.
	0001 000x		DS1DSGCX	BTAM or QTAM line group.
	. . . . xx . .			Reserved.
	0000 001x		DS1DSGPO	Partitioned organization.
	 1		DS1DSGU	Unmovable; the data contains location dependent information.
			Second byte of DS1DSORG	
	100x 00xx		DS1DSGGS	Graphics organization.
	010x 00xx		DS1DSGTX	TCAM line group (not supported)
	001x 00xx		DS1DSGTQ	TCAM message queue (not supported).
	000x 10xx		DS1ACBM	VSAM data set/space.
	000x 10xx		DS1ORGAM	VSAM data set/space.
	000x 01xx		DS1DSGTR	TCAM 3705 (not supported).
	. . . x . . xx			Reserved.
84(X'54')	Character	1	DS1RECFM	Record format.
	10		DS1RECFF	Fixed length.
	01		DS1RECFV	Variable length.
	11		DS1RECFU	Undefined length.
	. . 1		DS1RECFT	Track overflow. No longer supported by current hardware.
	. . . 1		DS1RECFB	Blocked; cannot occur with undefined.
	. . . . 1 . . .		DS1RECFS	Fixed length: standard blocks; no truncated blocks or unfilled tracks except possible the last block and track. Variable length: spanned records.
	. . . . . 00 .			No control character.
	. . . . . 10 .		DS1RECFA	ISO/ANSI control character.
	. . . . . 01 .		DS1RECMC	Machine control character.
	. . . . . 11 .			Reserved.
	 x			Reserved.

Table 2. DSCB Format-1 or Format-8. DSCB Format-1 or Format-8 *(continued)*

Offset Dec (Hex)	Type or Bit Mask	Length	Name	Description
85(X'55')	Character	1	DS1OPTCD	Option Code.
BDAM OPTCD field assignments (applies only if DS1DSGDA is on):				
	1			Write validity check.
	. 1			Track overflow.
	. . 1			Extended search.
	. . . 1			Feedback.
	. . . . 1 . . .			Actual addressing.
	. . . . . 1 . .			Dynamic buffering.
	. . . . . 1 .			Read exclusive.
	 1			Relative block addressing.
ISAM OPTCD field assignments (applies only if DS1DSGIS is on):				
	1			Write validity check.
	. 1			Accumulate track index entry.
	. . 1			Master indices.
	. . . 1			Independent overflow area.
	. . . . 1 . . .			Cylinder overflow area.
	. . . . . 1 . .			Reserved.
	. . . . . 1 .			Delete option.
	 1			Reorganization criteria.
BPAM, BSAM, QSAM OPTCD field assignments (applies only if DS1DSGPO or DS1DSGPS is on):				
	1			Write validity check.
	. 1			Allow data check (if on printer).
	. . 1			Chained scheduling.
	. . . 1			VSE/MVS interchange feature on tape.
	. . . . 1 . . .			Treat EOF as EOV (tape).
	. . . . . 1 . .			Search direct.
	. . . . . 1 .			User label totaling.
	 1			Each record contains a table reference character.
85(X'55')	Bitstring	1	DS1OPTAM	VSAM OPTCD settings.
VSAM OPTCD field assignments (applies only if DS1ORGAM is on):				
	1			Reserved.
	. 1		DS1OPTBC	Data set is an integrated catalog facility catalog.
	. . xx xxxx			Reserved.
86(X'56')	Binary	2	DS1BLKL	Block length (Type F unblocked records), or maximum block size (F blocked, U or V records).

Table 2. DSCB Format-1 or Format-8. DSCB Format-1 or Format-8 *(continued)*

Offset Dec (Hex)	Type or Bit Mask	Length	Name	Description
88(X'58')	Binary	2	DS1LRECL	Logical record length: Fixed length-record length, Undefined length-zero, Variable unspanned-maximum record length, Variable spanned and < 32757 bytes-maximum record length, Variable spanned and > 32756 bytes-X'8000'.
90(X'5A')	Binary	1	DS1KEYL	Key length (0 to 255).
91(X'5B')	Binary	2	DS1RKP	Relative key position.
93(X'5D')	Character	1	DS1DSIND	Data set indicators.
	1		DS1IND80	Last volume containing data in this data set.
	. 1		DS1IND40	Data set is RACF™, a component of the Security Server for z/OS, defined with a discrete profile.
	. . 1		DS1IND20	Block length is a multiple of 8 bytes.
	. . . 1		DS1IND10	Password is required to read or write, or both; see DS1IND04.
	. . . . 1 . . .		DS1IND08	Data set has been modified since last recall.
	. . . . . 1 . .		DS1IND04	If DS1IND10 is 1 and DS1IND04 is 1, password required to write, but not to read. If DS1IND10 is 1 and DS1IND04 is 0, password required both to write and to read.
	. . . . . 1 .		DS1IND02	Data set opened for other than input since last backup copy made.
			DS1DSCHA	Same as DS1IND02.
	 1		DS1IND01	Secure checkpoint data set.
			DS1CHKPT	Same as DS1IND01.
94(X'5E')	Binary	4	DS1SCALO	Secondary allocation space parameters.
94(X'5E')	Character	1	DS1SCAL1	Flag byte.
	xxxx xxxx		DS1DSPAC	Space request bits, defined as follows:
	00 . 0		DS1DSABS	Absolute track request.
	00 . 1		DS1EXT	Extension to secondary space exists (see DS1SCEXT at offset 79 (X'4F')).
	11		DS1CYL	Cylinder request.
	10		DS1TRK	Track request.
	01 x		DS1AVR	Average block length request.
	01 1		DS1AVRND	Average block and round request.
	. . 1		DS1MSGP	Mass storage vol group (MSVGP - no longer supported).
	. . . . 1 . . .		DS1CONTG	Contiguous request.
	. . . . . 1 . .		DS1MXIG	MXIG request.
	. . . . . 1 .		DS1ALX	ALX request.
	 1		-	Round request.
95(X'5F')	Binary	3	DS1SCAL3	Secondary allocation quantity.
98(X'62')	Binary	3	DS1LSTAR	Last used track and block on track (TTR). Not defined for VSAM, PDSE, HFS and direct (BDAM). See bit DS1LARGE at +61 and byte DS1TTTHI at +104.

Table 2. DSCB Format-1 or Format-8. DSCB Format-1 or Format-8 *(continued)*

Offset Dec (Hex)	Type or Bit Mask	Length	Name	Description
101(X'65')	Binary	2	DS1TRBAL	If not extended format, this is the value from TRKCALC indicating space remaining on last track used. For extended format data sets this is the high order two bytes (TT) of the four-byte last used track number. See DS1LSTAR. Zero for VSAM, PDSE, and HFS.
103(X'67')	Character	1		Reserved.
104(X'68')	Character	1	DS1TTTHI	High order byte of track number in DS1LSTAR. Valid if DS1Large is on.
105(X'69')	Character	30	DS1EXNTS	Three extent fields.
105(X'69')	Character	10	DS1EXT1	First extent description.
	Character	1		Extent type indicator.
	X'81'			Extent on cylinder boundaries.
	X'80'			Extent described is sharing cylinder (no longer supported).
	X'40'			First extent describes the user labels and is not counted in DS1NOEPV.
	X'04'			Index area extent (ISAM).
	X'02'			Overflow area extent (ISAM).
	X'01'			User's data block extent, or a prime area extent (ISAM).
	X'00'			This is not an extent.
		1		Extent sequence number.
		4		Lower limit (CCHH). These are the bit definitions: **0–15** Low order 16 bits of the 28-bit cylinder number. **16-27** High order 12 bits of the 28-bit cylinder number. In a format-1 DSCB, these bits always are zero. **28-31** Track number from 0 to 14. Use the TRKADDR macro or IECTRKAD routine when performing track address calculations.
		4		Upper limit (CCHH). Same format as the lower limit.
115(X'73')	Character	10	DS1EXT2	Second extent description.
125(X'7D')	Character	10	DS1EXT3	Third extent description.
135(X'87')	Character	5	DS1PTRDS	In a format-1 DSCB this can be a pointer (CCHHR) to a format-2 or format-3 DSCB or be zero. In a format-8 DSCB this always is the CCHHR of a format-9 DSCB.
140(X'8C')	Character		DS1END	-

Format-2 DSCB

This format applied only to ISAM data sets, which can no longer be created or opened.

Format-3 DSCB

Name

Extension

Function

Describes extents after the third extent of a non-VSAM data set or a VSAM data space.

How Many

One for each data set or VSAM data space on the volume that has more than three extents. There can be as many as 10 for a PDSE, HFS, extended format data set, or a VSAM data set cataloged in an integrated catalog facility catalog. PDSEs, HFS, and extended format data sets can have up to 123 extents. Each component of a VSAM data set cataloged in an integrated catalog facility catalog can have up to 123 extents per volume. All other data sets are restricted to 16 extents per volume.

How Found

Chained from a format 1, format 2 or format 9 DSCB that represents the data set. In the case of a PDSE, HFS data set, sequential extended-format data set, or VSAM data set, the chain also can be from a preceding format-3 DSCB.

Table 3 on page 11 shows the contents of a format-3 DSCB.

Table 3. DSCB Format-3

Offset Dec (Hex)	Type	Length	Name	Description
0(X'00')	Bitstring	4	-	Key identifier (X'03030303').
4(X'04')	Bitstring	40	DS3EXTNT	Four extent descriptions.
		1		Extent type indicator. (See DS1EXT1 in Table 2 on page 5.)
		1		Extent sequence number.
		4		Lower limit (CCHH).
		4		Upper limit (CCHH).
44(X'2C')	Character	1	DS3FMTID	Format identifier (X'F3').
			DS3IDC	Constant value of X'F3' in DS3FMTID.
45(X'2D')	Bitstring	90	DS3ADEXT	Nine additional extent descriptions.
135(X'87')	Bitstring	5	DS3PTRDS	Pointer (CCHHR) to next format-3 DSCB, or zero.
140(X'8C')			DS3END	-

Format-4 DSCB

Name

VTOC

Function

Describes the extent and contents of the VTOC and provides volume and device characteristics. This DSCB contains a flag indicating whether the volume is SMS managed.

How Many

One on each volume.

How Found

The VOLVTOC field of the standard volume label contains the format-4 address. The format-4 DSCB is always the first record in the VTOC.

Table 4 on page 12 shows the contents of a format-4 DSCB.

Exception: The format-4 DSCB has a 44-byte key of X'04' bytes not shown in Table 4 on page 12.

Table 4. DSCB Format-4

Offset Dec (Hex)	Type	Length	Name	Description
0(X'00')	Character	1	DS4IDFMT	Format identifier (X'F4').
			DS4IDC	Constant value of X'F4' in DS4FMT.
1(X'01')	Character	5	DS4HPCHR	Highest address (CCHHR) of a format-1 DSCB.
6(X'06')	Unsigned	2	DS4DSREC	Number of available DSCBs.
8(X'08')	Character	4	DS4HCCHH	CCHH of next available alternate track.
12(X'0C')	Unsigned	2	DS4NOATK	Number of remaining alternate tracks.
14(X'0E')	Bitstring	1	DS4VTOCI	VTOC indicators.
	1		DS4DOSBT	VSE bit. Either invalid format 5 DSCBs or indexed VTOC. Previously DOS(VSE) bit. See DS4IVTOC.
	. 1		DS4DVTOC	Index was disabled.
	. . 1		DS4EFVLD	Extended free-space management flag. When DS4EFVLD is on, the volume is in OSVTOC format with valid free space information in the format-7 DSCBs. See also DS4EFLVL and DS4EFPTR.
	. . . 1		DS4DSTKP	VSE stacked pack.
	. . . . 1 . . .		DS4DOCVT	VSE converted VTOC.
	. . . . . 1 . .		DS4DIRF	DIRF bit. A VTOC change is incomplete.
	. . . . . 1 .		DS4DICVT	DIRF reclaimed.
	 1		DS4IVTOC	Volume uses an indexed VTOC.
15(X'0F')	Unsigned	1	DS4NOEXT	Number of extents in the VTOC(X'01').
16(X'10')	Bitstring	1	DS4SMSFG	System managed storage indicators.
	00		DS4NTSMS	Non-system managed volume.
	01		DS4SMSCV	System managed volume in initial status.
	10			Reserved.
	11		DS4SMS	System managed volume.
	11		DS4SMSTS	System managed volume.
	. . xx xxxx			Reserved.
17(X'11')	Binary	1	DS4DEVAC	Number of alternate cylinders when the volume was formatted. Subtract from first 2 bytes of DS4DEVSZ to get number of useable cylinders (can be 0). Valid only if DS4DEVAV is on.
18(X'12')	Character	14	DS4DEVCT	Device constants. For currently supported devices, these fields do not provide enough information to calculate the amount of space used on a track. Use TRKCALC (see "Performing Track Calculations (TRKCALC macro)" on page 292).
18(X'12')	Character	4	DS4DEVSZ	Device size.

Table 4. DSCB Format-4 (continued)

Offset Dec (Hex)	Type	Length	Name	Description
18(X'12')	Binary	2	DS4DSCYL	Number of logical cylinders including alternates, if any exist. Unsigned number. Set to X'FFFE' for devices with more than 65,520 cylinders, indicating that cylinder-managed space exists (DS4CYLMG is set on) and extended attribute DSCBs, formats 8 and 9, are allowed.
20(X'14')	Binary	2	DS4DSTRK	Number of tracks in a logical cylinder.
22(X'16')	Unsigned	2	DS4DEVTK	Device track length.
24(X'18')	Binary	2	DS4DEVOV	Keyed record overhead.
24(X'18')	Binary	1	DS4DEVI	Non-last-keyed record overhead.
25(X'19')	Binary	1	DS4DEVL	Last keyed record overhead.
26(X'1A')	Binary	1	DS4DEVK	Non-keyed record overhead differential.
27(X'1B')	Bitstring	1	DS4DEVFG	Flag byte 1.
	. . . 1		DS4DEVAV	Value in DS4DEVAC, number of alternate cylinders is valid. Otherwise, the number is not available in this DSCB.
	. . . . 1 . . .			Keyed record overhead field (DS4DEVOV) is used as a 2-byte field to specify the overhead required for a keyed record.
	. . . . . 1 . .			The CCHH of an absolute address is used as a continuous binary value. Not implemented in the current IBM product line.
	. . . . . 1 .			The CCHH of an absolute address is used as four separate binary values. Not implemented in the current IBM product line.
	 1			A tolerance factor must be applied to all but the last block of the track. Not implemented in the current IBM product line.
	xxx.			Reserved.
28(X'1C')	Binary	2	DS4DEVTL	Device tolerance.
30(X'1E')	Binary	1	DS4DEVDT	Number of DSCBs per track.
31(X'1F')	Binary	1	DS4DEVDB	Number of PDS directory blocks per track.
32(X'20')	Binary	8	DS4AMTIM	VSAM time stamp.
40(X'28')	Character	3	DS4AMCAT	catalog indicator.
40(X'28')	Bitstring	1	DS4VSIND	VSAM indicators.
	1		DS4VSREF	A catalog references this volume.
	. 1		DS4VSBAD	The VSAM data sets on this volume are unusable because an MSS CONVERTV command has not completed successfully for the volume. (No longer set.)
	. . 1		DS4VVDSA	Bit on indicate VVDS does exist
	. . 1		DS4VVDSR	If on, VVDS data set name was scanned. It is turned on once per volume when the VVDS was scanned on the volume.
	. . . x xxxx			Reserved.
41(X'29')	Unsigned	2	DS4VSCRA	Relative track location of the CRA.
43(X'2B')	Binary	8	DS4R2TIM	VSAM volume/catalog match time stamp.

Table 4. DSCB Format-4 (continued)

Offset Dec (Hex)	Type	Length	Name	Description
51(X'33')	Character	5		Reserved.
56(X'38')	Character	5	DS4F6PTR	Pointer (CCHHR) to first format-6 DSCB, or zero. (No longer supported as non-zero).
61(X'3D')	Character	10	DS4VTOCE	VTOC extent description.
71(X'47')	Character	10	Reserved.	VTOC extent description.
81(X'51')	Character	1	DS4EFLVL	Extended free-space management level. X'00' indicates extended free-space management is not used for this volume. X'07' indicates extended free-space management is in use for this volume (see also DS4EFVLD).
82(X'52')	Character	5	DS4EFPTR	Pointer to extended free-space information. If DS4EFLVL=X'00' this is zero. If DS4EFLVL=X'07' this is the CCHHR of the first FMT-7 DSCB and no format-5 DSCBs contain free space information.
87(X'57')	Character	1	DS4MCU	Minimum allocation size in cylinders for cylinder-managed space. Each extent in this space must be a multiple of this value.
88(X'58')	Character	4	DS4DCYL	Number of logical cylinders. Valid when DS4DSCYL= X'FFFE'.
92(X'5C')	Character	2	DS4LCYL	First cylinder address/4095 where space is managed in multicylinder units. Cylinder-managed space begins at this address. Valid when DS4CYLMG is set.
94(X'5E')	Character	1	DS4DEVF2	Device Flags Byte 2
	1...		DS4CYLMG	Cylinder-managed space exists on this volume and begins at DS4LCYL in multicylinder units of DS4MCU. DS4EADSCB is also set when this flag is on.
	.1..		DS4EADSCB	Extended attribute DSCBs, Format 8 and 9 DSCBs, are allowed on this volume.
		..xx xxxx		Reserved
95(X'5F')		1		Reserved
96(X'60')			DS4END	-

Format-5 DSCB

Name

Free Space

Function

On a nonindexed VTOC, describes the space on a volume that has not been allocated to a data set (available space). For an indexed VTOC, a single empty format-5 DSCB resides in the VTOC; free space is described in the index and DS4IVTOC is normally on.

How Many

One for every 26 noncontiguous extents of available space on the volume for a nonindexed VTOC; for an indexed VTOC, there is only one.

How Found

The first format-5 DSCB on the volume is always the second DSCB of the VTOC. If there is more than one format-5 DSCB, it is chained from the previous format-5 DSCB using the DS5PTRDS field.

Table 5 on page 15 shows the contents of a format-5 DSCB.

Table 5. DSCB Format-5

Offset Dec (Hex)	Type	Length	Name	Description
0(X'00')	Bitstring	4	DS5KEYID	Key identifier (X'05050505').
4(X'04')	Bitstring	5	DS5AVEXT	Available extent.
		2		Relative track address of the first track in the extent. Relative to the beginning of the volume.
		2		Number of unused cylinders in the extent.
		1		Number of additional unused tracks.
9(X'09')	Bitstring	35	DS5EXTAV	Seven available extents.
44(X'2C')	Character	1	DS5FMTID	Format identifier (X'F5').
45(X'2D')	Bitstring	90	DS5MAVET	Eighteen available extents.
135(X'87')	Bitstring	5	DS5PTRDS	Pointer (CCHHR) to next format-5 DSCB, or zero.
140(X'8C')			DS5END	-

Format-7 DSCB

Name

Free space for certain devices

Only one field in the format-7 DSCB is an intended interface. This field indicates whether the DSCB is a format-7 DSCB. You can reference that field as DS1FMTID or DS5FMTID. A character 7 indicates that the DSCB is a format-7 DSCB, and your program should not modify it.

If you are diagnosing a problem, see *z/OS DFSMSdfp Diagnosis* for the layout of the Format-7 DSCB.

Format-9 DSCB

Name

Metadata and DSCB pointers.

Function

Contains metadata about the data set and pointers to all format 3 DSCBs for the data set.

How Many

One for each format 8 DSCB.

How Found

Chained from a format-8 DSCB that represents the data set. Data sets that have a format 1 DSCB do not have a format 9 DSCB.

Table 6 on page 16 shows the contents of a format-9 DSCB.

Table 6. Format-9 DSCB

Offset Dec (Hex)	Type	Length	Name	Description
0 (X'00')	Character	1	DS9KEYID	Key identifier.
			DS9KEY	Constant value of X'09' in DS9KEYID
1 (X'01')	Binary	1	DS9SUBTY	Subtype number for format 9.
			DS9SUBT1	Constant value of binary 1 to represent subtype 1.
				If your program uses the content of a format 9 DSCB, then it should test the subtype field to learn the format of the DSCB. Currently only one subtype is defined
Beginning of the fields that are unique to the format 9 subtype 1 DSCB.				
2 (X'02')	Binary	1	DS9NUMF9	Number of format 9 DSCB's for this data set. Valid only in the first format 9 DSCB.
3 (X'03')	Character	1	DS9FLAG1	Format 9 DSCB flag byte 1.
	1		DS9CREAT	Format 9 DSCB built by create.
4 (X'04')	Character	8	DS9JOBNAME	Job name used to create the data set on the current volume. Valid only when DS9CREAT is on (see offset 3 (X'03')).
12 (X'0C')	Character	8	DS9STEPNAME	Step name used to create the data set on the current volume. Valid only when DS9CREAT is on (see offset 3 (X'03')).
20 (X'14')	Bytes	6	DS9TIME	Number of microseconds since midnight, local time, that the data set was created. See creation date field, DS1CREDT (offset 53(X'35')), for the date. Valid only if DS1VOLSQ is 1 (first volume) and DS9CREAT is on (see offset 3 (X'03')).
26 (X'20')	Character	18	*	Reserved.
44 (X'2C')	Character	1	DS9FMTID	Format identifier.
			DS9IDC	Constant value of X'F9' in DS9FMTID.
45 (X'2D')	Binary	1	DS9NUMF3	Number of format 3 pointers that follow
46 (X'2E')	Binary	50	DS9F3	Pointers to first to tenth format-3 DSCBs
The following five bytes occur ten times.				
46 (X'2E')	Binary	5	DS9F3P	First pointer to a format-3 DSCB
46 (X'2E')	Binary	2	DS9F3CC	Cylinder number in pointer to format-3
48 (X'30')	Binary	2	DS9F3HH	Track number in pointer to format-3
50 (X'32')	Binary	1	DS9F3R	Record number in pointer to format-3
96 (X'60')	Character	20	DS9ATRV1	Attribute bytes available for vendor use. See Vendor fields in DS9ATRV!.
116 (X'74')	Character	19	DS9ATRI2	Attribute fields, for future IBM definition
End of the fields that are unique to the format 9 subtype 1 DSCB.				
135 (X'87')	Bitstring	5	DS9PTRDS	Pointer (CCHHR) to next format-9 DSCB, the first format-3 DSCB or zero.
135 (X'87')	Binary	2	DS9CCPTR	Cylinder number in DSCB pointer
138 (X'89')	Binary	2	DS9HHPTR	Track number in DSCB pointer
140 (X'8B')	Binary	1	DS9RPTR	Record number in DSCB pointer
140 (X'8C')			DS9END	

Table 6. Format-9 DSCB (continued)

Offset Dec (Hex)	Type	Length	Name	Description
Vendor fields in DS9ATRV1 in the format 9 DSCB				
z/OS does not enforce any rules on the content of the DS9ATRV1 field. It is available for vendor products to set. IBM provides the following recommendations:				
Each vendor should store subfields in the following format beginning at the leftmost byte:				
0 (X'0')	Character	1		
	xxxx			Reserved
	 xxxx			Number of bytes that follow this two-byte header. Minimum value is 0000.
1 (X'1')	Character	1		One-byte field containing a vendor identification issued by IBM.
2 (X'2')	Character			Beginning of variable-length field up to 15 bytes.

Allocating and Releasing DASD Space

DADSM allocate and extend routines assign tracks and cylinders on direct access volumes for data sets. The DADSM extend routine gets additional space for a data set that has exceeded its primary allocation. The DADSM scratch and partial release routines release space that is no longer needed on a direct access volume.

The DADSM routines allocate and release space by adding, deleting, and modifying the DSCBs and updating the VTOC index. When space is released, the scratch routines free the DSCBs of the deleted data set or data space.

Every data set occupies an integral number of tracks. Unused space on a track cannot be used for another data set unless the whole track is released. The minimum amount of space allocated to a data set is zero tracks but a PDS with no tracks can never contain a member.

After a certain place on the volume, the system rounds up each primary or secondary space request so it is a multiple of 21 cylinders. Currently that place is after the first 65520 cylinders. The space within the first 65520 cylinders is called the track-managed space, even if data sets are allocated in units of cylinders. The space after the first 65520 cylinders is called the cylinder-managed space. If the volume capacity is 65520 cylinders or less, then all of the space is track-managed.

Note: When a user program needs to determine the characteristics of a given volume, it must separately check each of the following characteristics and not assume that the value obtained for one characteristic implies the values for the other three:

- Whether space beyond a certain place on the volume is managed in multicylinder units.
- The location on the volume where the multicylinder units begin.
- The number of cylinders in a multicylinder unit.
- Whether the VTOC contains format 8 and 9 DSCBs.

The VTOC Index

The VTOC index enhances the performance of VTOC access. The VTOC index is a physical-sequential data set on the same volume as the related VTOC. It consists of an index of data set names in DSCBs contained in the VTOC and volume free space information. The data set names are in format-1 and format-8 DSCBs. The free space information describes available tracks on the volume, available DSCBs in the VTOC and available records in the index.

An SMS-managed volume *requires* an indexed VTOC; otherwise, the VTOC index is highly recommended. For additional information about SMS-managed volumes, see *z/OS DFSMS Implementing System-Managed Storage*.

Note: You can use the ICKDSF REFORMAT REFVTOC command to rebuild a VTOC index to reclaim any no longer needed index space and to possibly improve access times.

z/OS does support sharing a non-SMS-managed volume that contains a VTOC index with a non-z/OS system. If the other system updates the VTOC and turns on the DS4DOSBT, then later when z/OS is used to modify the VTOC, DADSM can detect that the index is no longer valid. The z/VSE® operating system sets this bit on.

Device Support Facilities (ICKDSF) initializes a VTOC index into 2048-byte physical blocks, or 8192-byte physical blocks on an extended address volume, named VTOC index records (VIRs). The DEVTYPE INFO=DASD macro can be used to return the actual block size or it can be determined from examining the format-1 DSCB of the index data set. VIRs are used in several ways. A VTOC index contains the following kinds of VIRs:

VTOC index entry record (VIER) identifies the location of format-1 and format-8 DSCBs and the format-4 DSCB.
VTOC pack space map (VPSM) identifies the free and allocated space on a volume.
VTOC index map (VIXM) identifies the VIRs that have been allocated in the VTOC index.
VTOC map of DSCBs (VMDS) identifies the DSCBs that have been allocated in the VTOC.

A format-1 DSCB in the VTOC contains the name and extent information of the VTOC index. The name of the index must be 'SYS1.VTOCIX.*volser*', where '*volser*' is the serial number of the volume containing the VTOC and its index. The name must be unique within the system to avoid ENQ contention and must conform to standard data set naming conventions.

Note:

1. If first character of *volser* is numeric, you must either precede or replace the first character with the letter "V" (for example, if *volser* is 12345, then you should use either V12345 or V2345).
2. RMF statistics on the VTOC index will always show the data set name as SYS1.VTOCIX.V*volser* regardless of what the actual name is on the volume.

You can only create (allocate) one data set whose name begins with 'SYS1.VTOCIX.' on a volume. To rename a VTOC index data set when the VTOC index is active, use a name beginning with 'SYS1.VTOCIX.'. If a 'SYS1.VTOCIX.' data set already exists on a volume, you cannot rename another data set on the volume to a name with those qualifiers. If the VTOC index is active, you cannot scratch the VTOC index data set.

The relationship of a VTOC to its index is shown in Figure 4 on page 18.

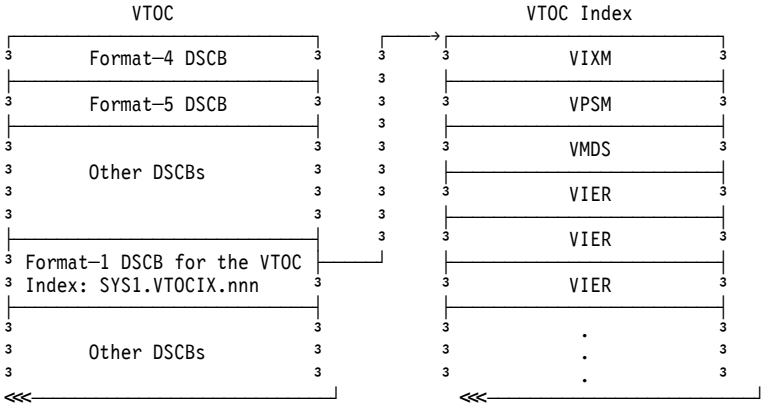


Figure 4. Example of the relationship of a VTOC to its index

VTOC Index Records

VTOC index records consist of the following types:

VTOC Index Entry Record

The first level-one VIER is created with the VTOC index. Subsequent VIERs are created whenever a VIER is too full to allow a data set name to be added to the VTOC index. VIERs have the following characteristics:

- A VIER uses one VIR and contains variable-length index entries. The number of VIERs in an index depends upon the number of data sets on the volume.
- All index entries within a VIER are at the same index level. VIERs in a VTOC index can be on several levels and have a hierarchic relationship. Index entries in higher-level VIERs point to lower-level VIERs. Index entries in level-one VIERs (those at the lowest level) point to format-1 or format-8 DSCBs for data sets on the volume.
- Whenever a fourth VIER is created on the same level, a VIER at a higher level is created. Once a higher-level VIER is filled with pointers to lower-level VIERs, another VIER at the same level is created.

VTOC Pack Space Map

The VPSM shows the allocated space on a volume and the space that remains free. The space within the first 65,520 cylinders is called the track-managed space. The VPSMs for this area contains bit maps of the cylinders and tracks on the volume. A value of one indicates that the cylinder or track has been allocated; zero, that it is unallocated. The space after the first 65,520 cylinders is called the cylinder-managed space. The VPSMs for this area contains bit maps of only multicylinder units. Individual cylinders and tracks are not mapped in this type of VPSM. A value of one indicates that the multicylinder unit is allocated; zero, that it is unallocated.

VTOC Index Map

The VIXM contains a bit map in which each bit represents one VTOC index record. The status of the bit indicates whether the VIR is allocated (1), or unallocated (0). On an extended address volume, the VIXM record header is enlarged to accommodate the volume size, free space statistics for the VTOC and index, free space statistics for the entire volume and from track-managed space. Fields in this expanded header define the RBA of the first VPSM record that maps multicylinder units, along with the minimum allocation unit in cylinders for the VPSMs that map cylinder-managed space.

An area of the VIXM is reserved for VTOC recording facility (VRF) data. (This facility allows detection of and recovery from some errors in an indexed VTOC.)

VTOC map of DSCBs

The VMDS shows the DSCBs that have been allocated in the VTOC. The map contains a bit map of DSCBs corresponding the relative DSCB record in the VTOC. A value of one indicates that the DSCB has been has been allocated; zero indicates that it is unallocated.

Structure of an Indexed VTOC

Indexed and nonindexed VTOCs have similar structures with the following differences for an indexed VTOC:

- Only a single, empty format-5 DSCB exists.
- Some format-4 DSCB data (the number of format-0 DSCBs and the CCHHR of the highest format-1 DSCB) is not maintained by DADSM. The VSE bit (bit 0 in field DS4VTOCI), set to 1 in the format-4 DSCB, indicates that the contents of these fields (and the format-5 DSCB) are not valid. The index bit (bit 7 in field DS4VTOCI) is set in the format-4 DSCB; it indicates that a VTOC index exists.

Accessing the VTOC with DADSM Macros

You can use either DADSM or common VTOC access facility (CVAF) macros to access a VTOC and its index. (CVAF access is described in “Accessing the VTOC with CVAF Macros” on page 55.) The DADSM macros and tasks covered here include:

- LSPACE provides information on volume size, free space on the volume, free space on the VTOC and INDEX, volume fragmentation, and VTOC status. Also provided is information on the size of the track-managed space and its free space statistics.
- OBTAIN reads one or more DSCBs from the VTOC.
- PARTREL releases unused space from a sequential or partitioned data set or a PDSE.
- REALLOC allocates DASD space.

To read one or more DSCBs into virtual storage, use the OBTAIN and CAMLST macro instructions. Identify the DSCB to be read using the name of the data set associated with the DSCB, or the absolute track address of the DSCB. Provide a 140-byte data area in virtual storage to contain the DSCB. On a request to read multiple DSCBs specify the NUMBERDSCB= parameter on the OBTAIN or CAMLST macro and provide consecutive 140-byte return areas in virtual storage to contain this number of DSCBs. When you specify the name of the data set, an identifier (format-1, format-4, or format-8) DSCB is read into virtual storage. To read a DSCB other than a format-1, format-4, or format-8 DSCB, specify an absolute track address (see the example on page “Example” on page 41). Code the EADSCB=OK on the OBTAIN or CAMLST macro when your program supports DSCBs that describe data sets with format-8 and format-9 DSCBs. The extent descriptors in DSCBs for a data set described with these formats may have 28-bit cylinder track addresses. Use the TRKADDR macro or IECTRKAD service to manipulate 16-bit or 28-bit cylinder track addresses.

Restriction: You cannot use the OBTAIN macro instruction with either a SYSIN or SYSOUT data set.

To release unused space from a sequential, partitioned, or key sequenced data set or a PDSE, use the PARTREL macro instruction. Your program must be APF authorized.

Another way is to code the RLSE option on the SPACE keyword on the DD statement or the dynamic allocation equivalent. This technique does not require APF authorization. It requires that your program open the data set with the OUTPUT, EXTEND, OUTIN, OUTINX or INOUT option and the last operation before closing the data set not be a read or POINT macro."

The following macro instruction descriptions include coding examples, programming notes, and exception return code descriptions.

Requesting DASD Volume Information Using LSPACE

LSPACE provides information on volume size, free space on the volume, free space on the VTOC and INDEX, volume fragmentation, and VTOC status. Also provided is information on the size of the track-managed space and its free space statistics. The LSPACE macro returns status information (such as LSPACE subfunction, return code, and reason code) in the parameter list. The LSPACE macro also returns the return code in register 15. For volumes that are configured with more than 9999 cylinders, you can use the EXPMSG option to create an expanded message return area that the LSPACE macro needs. For volumes that are configured with cylinder-managed space, you can use the XEXPMSG option to create an extended expanded message return area that the LSPACE macro needs. The use of XEXPMSG is recommended for all requests to return message data. The expanded data return area (EXPDATA) will return binary data of free space and total volume space information for volumes. For volumes with cylinder-managed space, this will be returned as free space for the entire volume and free space for the track-managed space. The two sets of free space data will be the same for a volume that does not have cylinder-managed space. The use of EXPDATA is recommended for all requests to return binary data. You can have LSPACE return additional information such as the format 4 DSCB, the total number of free extents on the volume or the fragmentation index. This information can be returned in the:

- Message return area
- Expanded message return area
- Extended expanded message return area
- Data return area
- Expanded data return area
- Format-4 DSCB return area

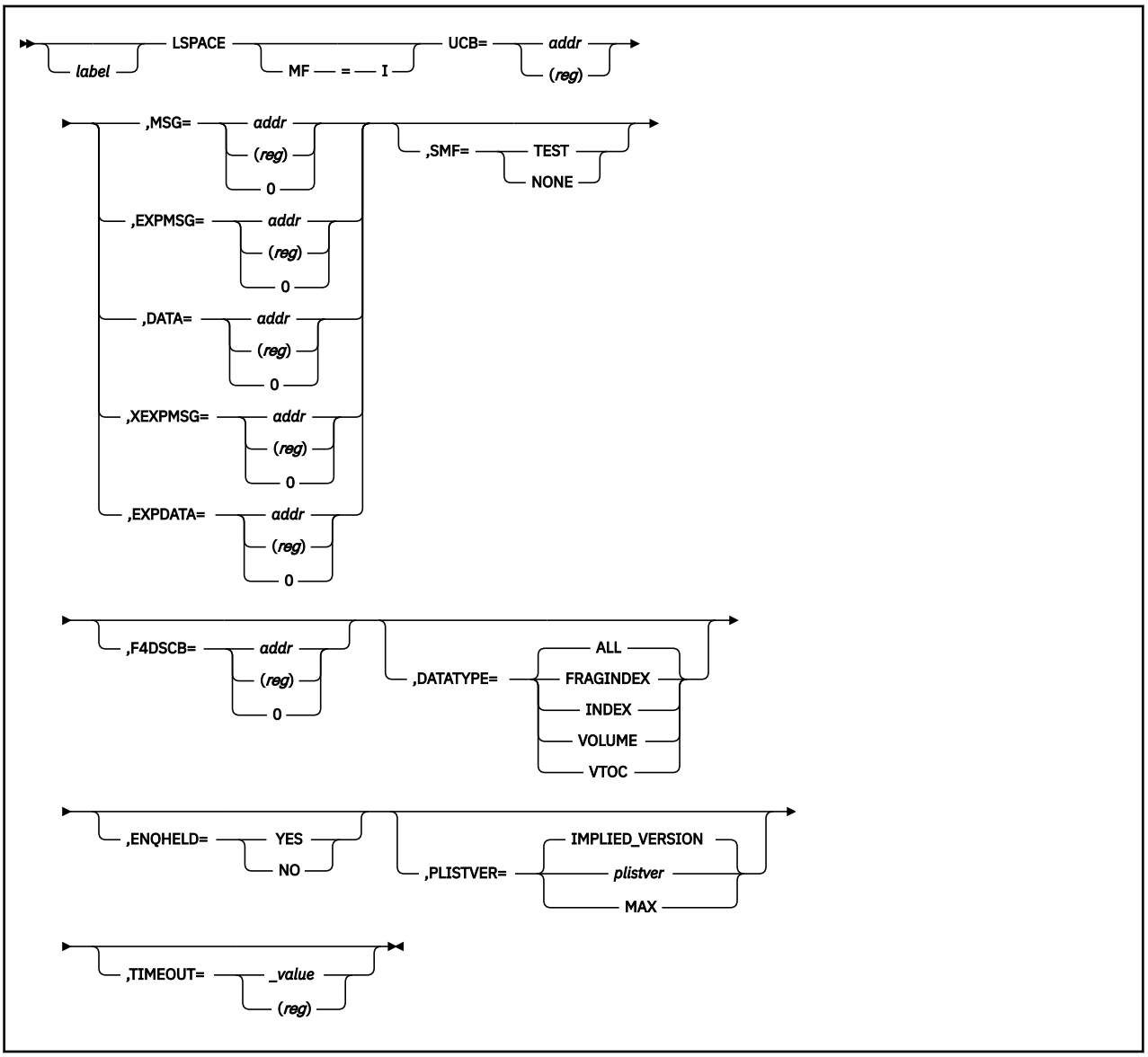
The calling program must ensure that the volume to be processed remains mounted during LSPACE processing. The volume need not be allocated.

If the device is not ready when you issue LSPACE and remains not ready, LSPACE eventually gives return code 4 with a timeout message. (See Table 8 on page 34.) In the current level of the system, LSPACE defaults to waiting as long as 240 seconds. You can change this amount of time by setting the byte at offset 7 in the parameter list. (See Table 7 on page 30.) You must use the list and execute forms of the macro because the macro has no parameter for this.

For more information about the LSPACE return code, subfunction code, and the subfunction return and reason codes, see Table 8 on page 34 and *z/OS DFSMSdfp Diagnosis*.

LSPACE—Standard Form

The format of the standard form of the LSPACE macro is:



The keywords are the same as described in the execute form of the LSPACE macro. See the execute form of the LSPACE macro for the descriptions.

MF=I
Specifies the standard form of the LSPACE macro.

I

Specifies the standard form of the macro. This generates a standard parameter list containing the required variables, loads the address of the parameter list in register 1, and issues a supervisor call. The standard form is the default.

Requirement: UCB must be specified when MF=I is used.

PLISTVER=plistver | IMPLIED_VERSION | MAX

This keyword defines the version of the LSPACE parameter list that should be generated for the MF=I form of the LSPACE macro.

PLISTVER=*plistver* specifies the version of the LSPACE parameter list that should be generated, where *plistver* is either 1 or 2. This PLISTVER= keyword is required for any macro keys associated with version 2 or larger to be specified. The macro keys associated with each supported version of the macro are listed below:

PLISTVER=1

PLISTVER=1 is associated with the following macro keys:

DATA
EXPMSG
F4DSCB
MSG
SMF

PLISTVER=2

PLISTVER=2 is associated with the following macro keys:

XEXPMSG
EXPDATA
DATATYPE
TIMEOUT
ENQHELD

When PLISTVER= IMPLIED_VERSION is specified the generated parameter list is the lowest version that allows all of the parameters on the invocation to be processed. When PLISTVER is omitted, the default is the lowest version of the parameter list, which is version 1.

When PLISTVER= MAX is specified, the generated parameter list is the largest size currently supported. This size may grow from release to release thus possibly affecting the amount of storage needed by your program. If your program can tolerate this, IBM recommends that you always specify MAX when creating the list form parameter list as that will ensure that the list form parameter list is always long enough to hold whatever parameters might be specified on the execute form.

TIMEOUT=__ value |_(reg)

Specifies the elapsed-time value, in units of seconds, that the caller is willing to wait for the LSPACE function to complete. Note that TIMEOUT is the elapsed-time value, not the address-of-value. This timeout value includes the time it takes for LSPACE to obtain the SYSVTOC ENQ resource. The value must be a positive number between 1 and 86400, inclusive.

value

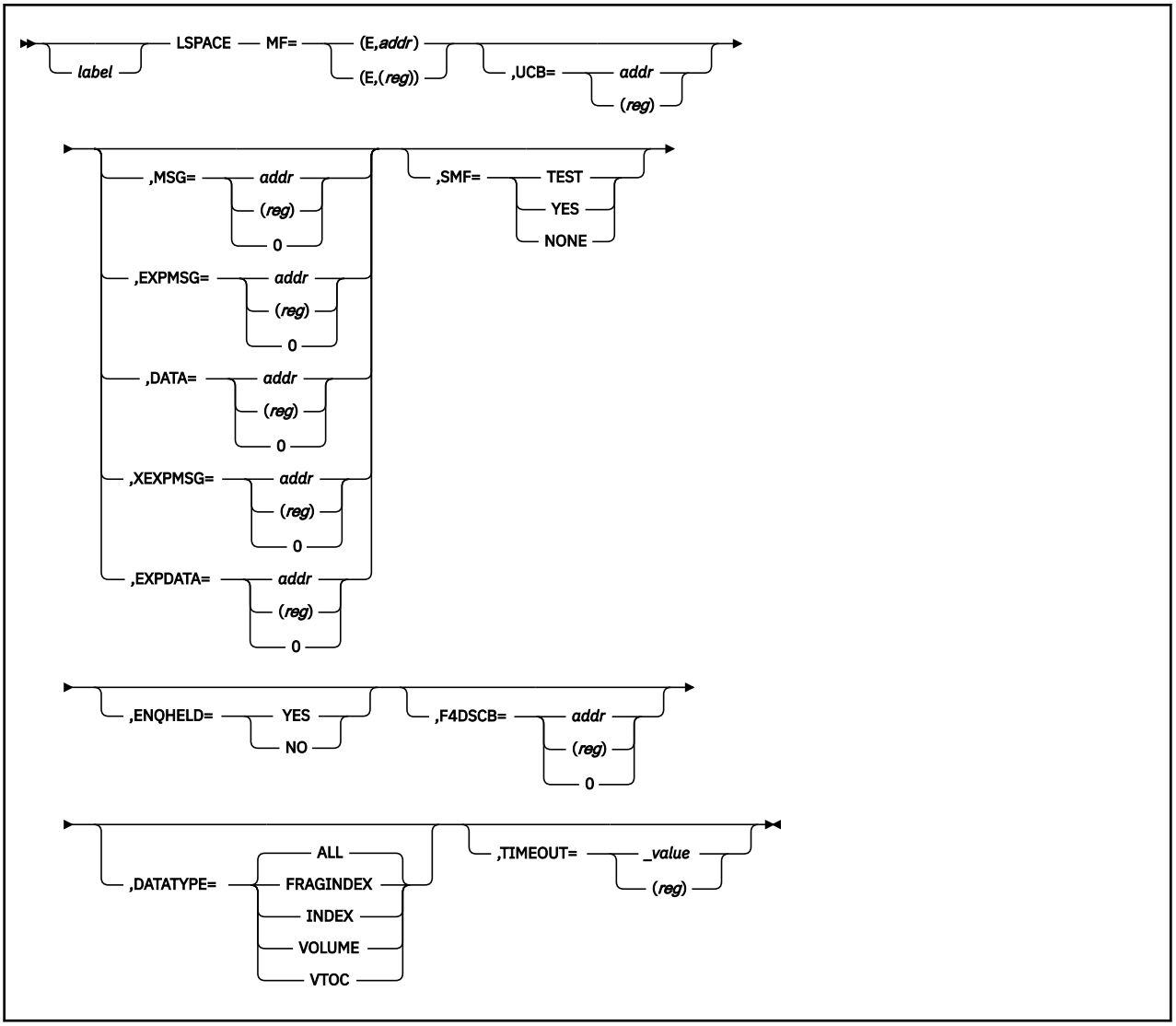
The value can be an integer number or a symbol.

(reg) (2-12)

Specifies a register containing the elapsed-time value, in units of seconds, that the caller is willing to wait for the LSPACE function to complete.

LSPACE-Execute Form

The format of the execute form of the LSPACE macro is:



MF=(E,*addr*) or (E,(*reg*))
Specifies the execute form of the LSPACE macro.

The MF=L form is usually issued before the execute form to create the parameter list.

(E,*addr*)
Loads the address of the parameter list specified by *addr* into register 1 and then issues a supervisor call.

(E,(*reg*))
Loads the address of the parameter list specified by (*reg*) into register 1 and then issues a supervisor call.

UCB=*addr* or (*reg*)
Specifies the address of the UCB or the address of a word in storage that contains the UCB address, depending on how the parameter is specified. The address can be for a captured UCB, or for an actual UCB above or below the 16MB line. For 31-bit callers, the high-order byte is part of the UCB address and must be cleared to zeros if a 24-bit UCB address is being passed.

***addr*—RX-type address**
Specifies the address of a fullword containing the UCB address.

(*reg*)—(2-12)
Specifies a register containing the UCB address for the device. Note that this differs from the *addr* form of the parameter.

When using the standard (MF=I) form of the macro, you must provide a UCB address.

Restrictions:

1. The LSPACE macro will accept the address of a UCB or UCB copy. Unauthorized programs can get a copy of the UCB by using the UCBSCAN macro and specifying the COPY and UCBAREA keywords. The UCB copy can be above or below the 16MB line and on a word boundary. Refer to *z/OS HCD Planning* for details.
2. LSPACE does not support VIO UCBs.

MSG=*addr or (reg)or* 0 or EXPMSG=*addr or (reg) or* 0 or DATA= *addr or (reg*) or 0
Specifies the way LSPACE is to return free space information. (optional)

Restriction: The MSG, EXPMSG, and DATA parameters are mutually exclusive.

MSG=*addr* or (*2-12*) or 0
Specifies the address of a caller-provided 30-byte message return area into which LSPACE returns either a free space message or, for unsuccessful requests, status information. For a description of this area, see "Message Return Area" on page 33.

***addr*–RX-type address**
Specifies the address of the message return area.

(*reg*)–(2-12)
Specifies a register containing the address of the message return area.

0
Specifies that you do not want the free space message. This is the default for all forms of the macro except execute.

EXPMSG=*addr* or (*reg*) or 0
Specifies the address of a caller-provided 40-byte expanded message return area into which LSPACE returns either a free space message or, for unsuccessful requests, status information. For a description of this area, see "Expanded Message Return Area" on page 35.

***addr*–RX-type address**
Specifies the address of the message return area.

(*reg*)–(2-12)
Specifies a register containing the address of the message return area.

0
Specifies that you do not want the free space message. This is the default for all forms of the macro except execute.

DATA=*addr* or (*reg*) or 0
Specifies the address of a caller-provided data return area into which LSPACE returns free space and volume information. For a description of this area, see "Data Return Area" on page 35.

***addr*–RX-type address**
Specifies the address of the data return area.

(*reg*)–(2-12)
Specifies a register containing the address of the data return area.

0
Specifies that you do not want the free space and volume information.

SMF=TEST or YES or NONE
Specifies the type of SMF processing.

TEST
Specifies that LSPACE is to test for an active SMF system and whether SMF volume information is desired. If these conditions are met, an SMF record is written.*Only programs executing in supervisor state, protect key 0-7, or APF authorized can specify this operand*.

YES
Specifies that you want an SMF record containing volume information to be written. *Only programs executing in supervisor state, protect key 0-7, or APF authorized can specify this operand*.

NONE
Specifies that you do not want an SMF record containing volume information to be written. This is the default for all forms of the macro except execute.

F4DSCB=*addr* or (*reg*) or 0
Specifies the address of a 96-byte DSCB return area provided by the calling program, into which LSPACE returns the volume's format-4 DSCB. For a description of the format-4 DSCB fields, see Table 4 on page 12.

***addr*– RX-type address**
Specifies the address of the format-4 DSCB return area.

(*reg*)– (2-12)
Specifies a register containing the address of the format-4 DSCB return area.

0
Specifies that you do not want the data portion of the format-4 DSCB for the volume. This is the default for all forms of the macro except execute.

XEXPMSG=addr or (reg) or 0
Specifies the address of a caller-provided 95-byte extended expanded message return area into which LSPACE returns either a free space message or, for unsuccessful requests, status information. Specify this keyword if you wish to obtain free space information in the message return area for volumes that are configured with cylinder-managed space. The returned free space will include space for the total volume and space from the track-managed space on a volume. The two sets of free space message data will be the same for a volume that does not have cylinder-managed space. The use of XEXPMSG is recommended for all requests to return message data. See "LSPACE Information Return Areas" on page 32 for a description of the message return area.

***addr*– RX-type address**
Specifies the address of the message return area.

(*reg*)– (2-12)
Specifies a register containing the address of the message return area.

0
Specifies that you do not want the free space message. This is the default for all forms of the macro except execute.

EXPDATA=addr or (reg) or 0
Specifies the address of a caller-provided EXPANDED data return area into which LSPACE returns expanded free space and volume information. Specify this keyword if you wish to obtain free space information and total volume space in the LSPACE data return area for volumes. The returned free space will include space for the total volume and space from the track managed space on a volume. For volumes with cylinder-managed space this data will be returned as free space for the entire volume and free space for the track-managed space. The two sets of free space data will be the same for a volume that does not have cylinder-managed space. The use of EXPDATA is recommended for all requests to return binary data. See Table 9 on page 35 for a description of the expanded data return area.

***addr*– RX-type address**
Specifies the address of the expanded data return area.

(*reg*)– (2-12)
Specifies a register containing the address of the expanded data return area.

0
Specifies that you do not want the expanded free space and volume information.

ENQHELD=YES or NO
Specifies whether the LSPACE address space of the caller has already obtained the SYSVTOC resource.

YES
Specifies that the LSPACE address space of the caller (which can be the TCB calling LSPACE, or another task within the address space of the caller), has issued a RESERVE or ISGENQ

REQUEST=OBTAIN,RESERVEVOLUME=YES for the SYSVTOC resource of the volume for which LSPACE is being called. When ENQHELD=YES is specified, the caller must obtain the SYSVTOC resource before invoking LSPACE (LSPACE verifies that the address space of the caller holds the SYSVTOC resource). The address space can obtain the SYSVTOC resource *shared* or *exclusive*, but the SYSVTOC must be a SYSTEMS ENQ of the RESERVE type. Specifying ENQHELD=YES on the execute form will override the specifications or the defaulting of ENQHELD=NO on the list form. Note that APF-authorization is required for the caller to obtain the SYSVTOC resource.

NO
Specifies that the LSPACE address space of the caller has not obtained the SYSVTOC resource before calling LSPACE. This is the default. If the caller holds SYSVTOC but has specified or defaulted to ENQHELD=NO, the LSPACE request is failed. Specifying ENQHELD=NO on the execute form will override the specification of ENQHELD=YES on the list form.

DATATYPE= ALL or FRAGINDEX or INDEX orVOLUME orVTOC
This keyword is allowed only when the DATA or EXPDATA keyword is specified. Only the information specified will be returned to the caller. DATATYPE is valid for both non-EAV and EAV. This keyword will eliminate unnecessary I/O required to retrieve free space information that is not be required by the caller. DATATYPE=ALL is the default.

ALL
Provide all available LSPACE statistics. This is the default

FRAGINDEX
Provide the fragmentation index

INDEX
Provide free space information related to the VTOC INDEX

VOLUME
Provide free space information for the VOLUME

VTOC
Provide free space information related to the VTOC

PLISTVER
This keyword, if specified on the MF=E form of the LSPACE macro, will generate an MNOTE.

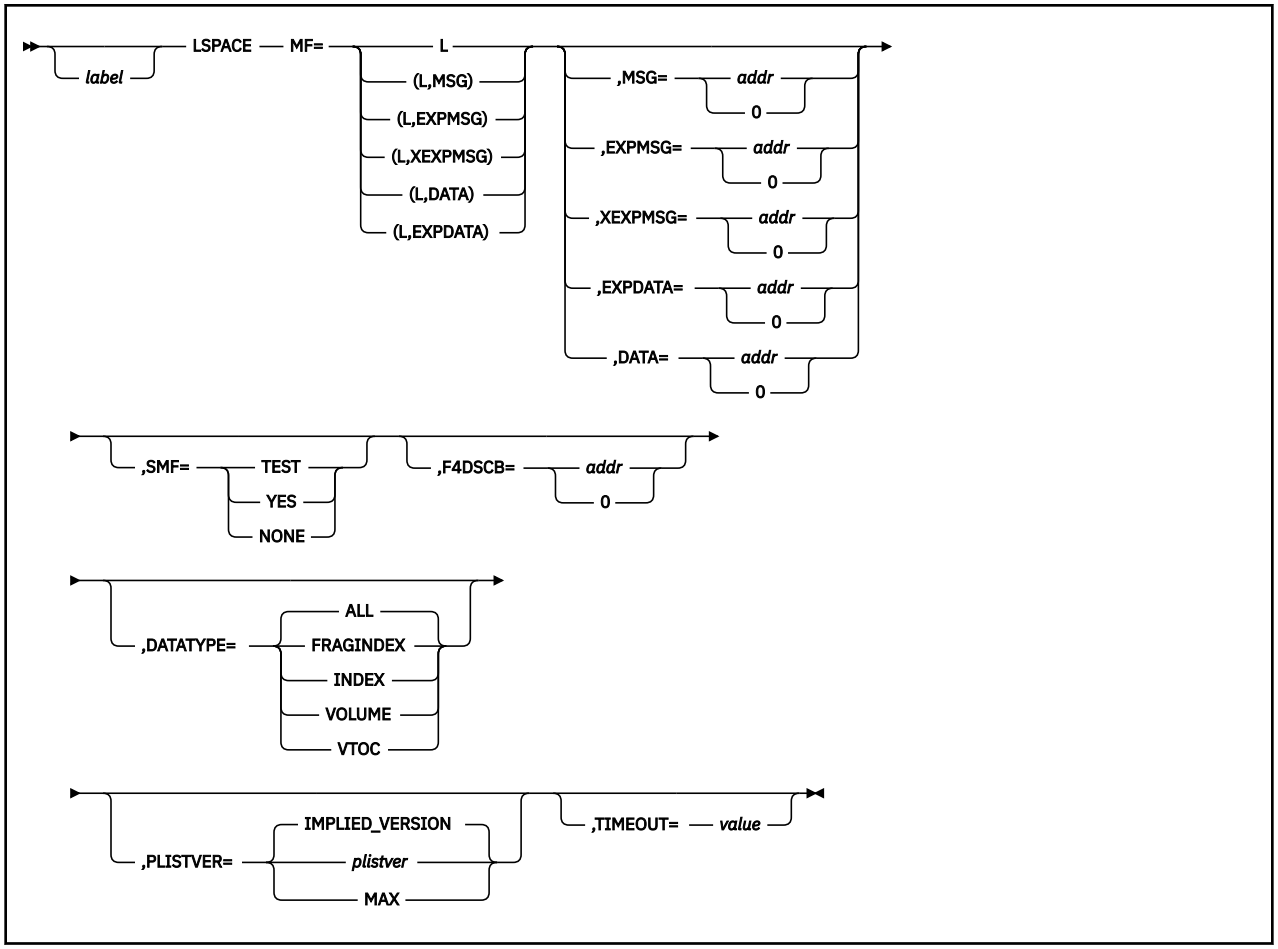
TIMEOUT=value or (reg)
Specifies the elapsed-time value, in units of seconds, that the caller is willing to wait for the LSPACE function to complete. Note that TIMEOUT is the elapsed-time value, not the address-of-value. This timeout value includes the time it takes for LSPACE to obtain the SYSVTOC ENQ resource. The value must be a positive number between 1 and 86400, inclusive.

value
The value can be an integer number or a symbol.

(reg) (2-12)
Specifies a register containing the elapsed-time value, in units of seconds, that the caller is willing to wait for the LSPACE function to complete.

LSPACE—List Form

The format of the list form of the LSPACE macro is:



MF=L or (L,MSG) or (L,EXPMSG) or (L,EXPMSG) or (L,DATA) or (L,EXPDATA)
Specifies the list form of the LSPACE macro.

L
Generates the required constants in the calling program. You can then issue the execute form of the macro, which uses these constants.

(L,MSG)
Generates the required message return area constants in the calling program. No other parameters are allowed.

(L,EXPMSG)
Generates the required expanded message return area constants in the calling program. No other parameters are allowed.

(L,XEXPMSG)
Generates the required extended expanded message return area constants (95-bytes) in the calling program. No other parameters are allowed. Use this keyword to obtain the message data area for all volume sizes including ones with cylinder-managed space.

(L,DATA)
Generates the required data return area constants in the calling program. No other parameters are allowed.

(L,EXPDATA)
Generates the required expanded data return area constants in the calling program. No other parameters are allowed. Use the keyword to obtain the data area for all volume sizes including ones with cylinder-managed space.

DATATYPE=ALL or FRAGINDEX or INDEX orVOLUME orVTOC
This keyword is allowed only when the DATA or EXPDATA keyword is specified. Only the information specified will be returned to the caller. DATATYPE is valid for both non-EAV and EAV. This keyword will

eliminate unnecessary I/O required to retrieve free space information that is not be required by the caller. DATATYPE=ALL is the default.

ALL
Provide all available LSPACE statistics. This is the default

FRAGINDEX
Provide the fragmentation index

INDEX
Provide free space information related to the VTOC INDEX

VOLUME
Provide free space information for the VOLUME

VTOC
Provide free space information related to the VTOC

PLISTVER=plistver | IMPLIED_VERSION | MAX
This keyword defines the version of the LSPACE parameter list that should be generated for the MF=I form of the LSPACE macro.

PLISTVER=*plistver* specifies the version of the LSPACE parameter list that should be generated, where *plistver* is either 1 or 2. This PLISTVER= keyword is required for any macro keys associated with version 2 or larger to be specified, The macro keys associated with each supported version of the macro are listed below:

PLISTVER=1
PLISTVER=1 is associated with the following macro keys:

DATA
EXPMSG
F4DSCB
MSG
SMF

PLISTVER=2
PLISTVER=2 is associated with the following macro keys:

XEXPMSG
EXPDATA
DATATYPE
ENQHELD
TIMEOUT

When PLISTVER= IMPLIED_VERSION is specified the generated parameter list is the lowest version that allows all of the parameters on the invocation to be processed. When PLISTVER is omitted, the default is the lowest version of the parameter list, which is version 1.

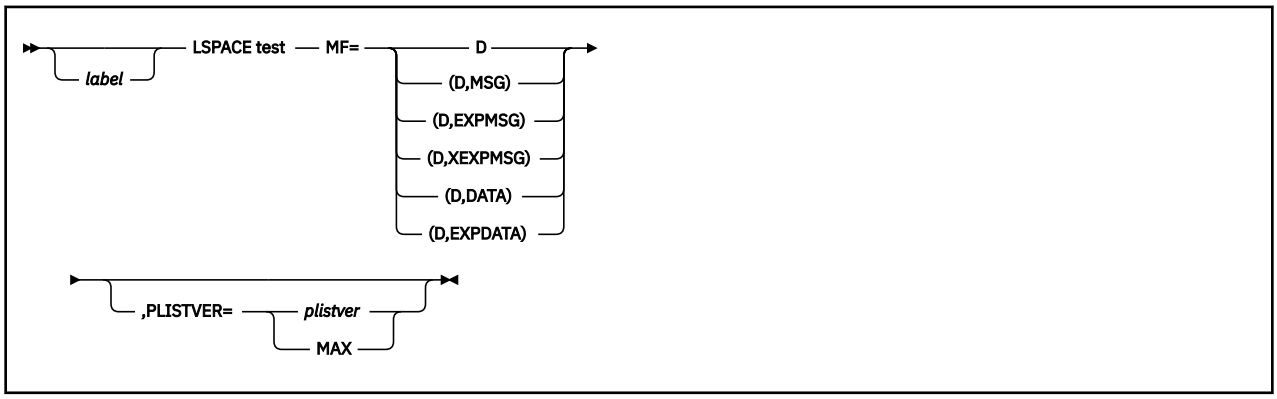
When PLISTVER= MAX is specified, the generated parameter list is the largest size currently supported. This size may grow from release to release thus possibly affecting the amount of storage needed by your program. If your program can tolerate this, IBM recommends that you always specify MAX when creating the list form parameter list as that will ensure that the list form parameter list is always long enough to hold whatever parameters might be specified on the execute form.

TIMEOUT=value
Specifies the elapsed-time value, in units of seconds, that the caller can wait for the LSPACE function to complete. Note that this is the elapsed-time value, not the address-of-value. This timeout value includes the time it takes for LSPACE to obtain the SYSVTOC ENQ resource. The value must be a positive number between 1 and 86400, inclusive.

value
The value can be an integer number or a symbol.

LSPACE–DSECT Form

The format of the DSECT form of the LSPACE macro is:



MF=D or (D,MSG) or (D,EXPMSG) or (D,XEXPMSG) or (D,DATA) or (D,EXPDATA)
Specifies the DSECT form of the LSPACE macro.

D
Generates a DSECT that maps the LSPACE parameter list. No other parameters are allowed. See Table 7 on page 30 for the format of the LSPACE parameter list.

(D,MSG)
Generates a DSECT that maps the message return area. No other parameters are allowed. For the format of the area, see "Message Return Area" on page 33.

(D,EXPMSG)
Generates a DSECT that maps the expanded message return area. No other parameters are allowed. For the format of the area, see "Expanded Message Return Area" on page 35.

(D,XEXPMSG)
Generates a DSECT that maps the extended expanded message return area (95-bytes). No other parameters are allowed. Use this keyword to obtain the message data area for all volume sizes including ones with cylinder-managed space. For the format of the area, see "Expanded Message Return Area" on page 35.

(D,DATA)
Generates a DSECT that maps the data return area. No other parameters are allowed. For the format of the area, see "Data Return Area" on page 35.

(D,EXPDATA)
Generates a DSECT that maps the base and expanded data return area. No other parameters are allowed. Use the keyword to obtain the data area for all volume sizes including ones with cylinder-managed space. For the format of the area, see "Data Return Area" on page 35.

PLISTVER=plistver | MAX
This keyword defines the version of the LSPACE parameter list that should be generated for the MF=D form of the LSPACE macro.

PLISTVER=*plistver* specifies the version of the LSPACE parameter list that should be generated, where *plistver* is either 1 or 2.This PLISTVER= keyword is required for any macro keys associated with version 2 or larger to be specified, The macro keys associated with each supported version of the macro are listed below:

PLISTVER=1
PLISTVER=1 is associated with the following macro keys:

DATA
EXPMSG
F4DSCB
MSG
SMF

PLISTVER=2

PLISTVER=2 is associated with the following macro keys:

XEXPMSG
EXPDATA
DATATYPE
ENQHELD
TIMEOUT

When PLISTVER= MAX is specified, the generated parameter list is the largest size currently supported. This size may grow from release to release thus possibly affecting the amount of storage needed by your program. If your program can tolerate this, IBM recommends that you always specify MAX when creating the list form parameter list as that will ensure that the list form parameter list is always long enough to hold whatever parameters might be specified on the execute form.

When PLISTVER is omitted, the default is the lowest version of the parameter list mapping.

Table 7. Format of the LSPACE Parameter List (MF=D)

Name	Offset	Bytes	Description
LSPAPL			
LSPAPLID	0(X'00')	4	EBCDIC 'LSPA'.
LSPANGTH	4(X'04')	2	Length of parameter list.
LSPAFLAG	6(X'06')	1	Parameter flag byte.
LSPASMFY		10	SMF=YES.
LSPASMFT		01	SMF=TEST.
LSPADATA		. . 1 0 0 . . .	Free space data request.
LSPAMSG		. . 0 1 0 . . .	Message data returned.
LSPAEMSG		. . 0 0 1 . . .	Expanded message data requested.
LSPAEPLP		. . 0 0 0 1 . .	Expanded LSPACE parameter input list provided
LSPAXINF		. . 0 0 0 0 1 .	Set on when expanded data (EXPDATA) or extended expanded message (XEXPMSG) information is returned by LSPACE processing. Systems prior to z/OS V1R10 do not set this flag. A program assembled (on a z/OS V1R10 or later system) with the LSPACE macro using the extended EXPMSG information respectively when the program is run on a system prior to z/OS V1R10.
LSPAFRES		 x	Reserved.
LSPAXTIM	7(X'07')	1	The maximum amount of time (seconds) LSPACE is allowed to run. Default is 240 seconds.
LSPAERCD	8(X'08')	1	LSPACE return code. See “Message Return Area” on page 33.
LSPASFID	9(X'09')	1	LSPACE subfunction code to further describe the LSPACE result.
LSPASFPC		X'00'	Processing complete.
LSPASF01		X'01'	Validate parameters.
LSPASF02		X'02'	Check UCB status.
LSPASF06		X'06'	Read volume label (EXCP).
LSPASF07		X'07'	Read volume label with Timeout.
LSPASF08		X'08'	ISGENQ OBTAIN RESERVEVOLUME on SYSVTOC

Table 7. Format of the LSPACE Parameter List (MF=D) (continued)

Name	Offset	Bytes	Description
LSPASF09		X'09'	STIMERM SET
LSPASF0A		X'0A'	ISGENQ RELEASE SYSVTOC.
LSPASF0B		X'0B'	ISGENQ RELEASE (conditional).
LSPASF0C		X'0C'	ISGQUERY for ENQHELD=YES
LSPASF0D		X'0D'	Validate UCB status immediately-prior to ISGQUERY or ENQ-SYSVTOC.
LSPASF0E		X'0E'	Examine ISGQUERY results.
LSPASF4X		X'80'	Read F4 and maps (CVAFDIR).
LSPASFEX		X'81'	Get free extents (CVAFDSM).
LSPASFF0		X'82'	Get F0 count (CVAFDSM).
LSPASFVR		X'83'	Get VIR count (CVAFDSM).
LSPASFVD		X'84'	Check for VRF (CVAFVRF).
LSPASF85		X'85'	ESTAE routine entered. Processing error in LSPACE.
LSPASF86		X'86'	LSPACE STIMERM timeout
LSPASFRT	10(X'0A')	1	Subfunction return code.
LSPASFRS	11(X'0B')	1	Subfunction reason code.
LSPARS01		X'01'	Invalid parameter list storage key.
LSPARS02		X'02'	Invalid parameter list ID.
LSPARS03		X'03'	Invalid LSPACE flag.
LSPARS04		X'04'	Invalid authorization for System Management Facility (SMF) flag.
LSPARS05		X'05'	Invalid message or data return area storage key.
LSPARS06		X'06'	Invalid format-4 DSCB return area storage key.
LSPARS07		X'07'	Invalid UCB address.
LSPARS08		X'08'	Invalid virtual UCB address.
LSPARS09		X'09'	Invalid VTOC pointer (UCBVTOC). Either UCBVTOC is zero, or it does not match the volume label.
LSPARS0A		X'0A'	Check for parm list length
LSPARS0B		X'0B'	ESTAE return code nonzero.
LSPARS0C		X'0C'	Timeout of ISGENQ OBTAIN for the SYSVTOC resource.
LSPARS0D		X'0D'	Internal error in ECBs.
LSPARS0E		X'0E'	UCBVOLI is binary-zeroes (volume has gone offline)
LSPARS0F		X'0F'	
LSPARS10		X'10'	ISGQUERY-results indicates that pointer-to the first RS (resource) record is zero
LSPARS11		X'11'	ISGQUERY-results indicates that pointer-to the first RQ (requester) record is zero.
LSPARS12		X'12'	ISGQUERY-results indicates that no RQ (requester) is the *owner* of the SYSVTOC resource.
LSPARS13		X'13'	The required RESERVE of SYSVTOC was not performed by the address space of the caller

Table 7. Format of the LSPACE Parameter List (MF=D) (continued)

Name	Offset	Bytes	Description
LSPARS14		X'14'	The TIMEOUT value is zero or negative
LSPARS15		X'15'	The TIMEOUT value is not within allowable range
LSPAUCB	12(X'0C')	4	UCB address.
LSPAFRSP	16(X'10')	4	Address of message or data return area.
LSPAFMT4	20(X'14')	4	Address of format-4 DSCB.
LSPAEXPL	24 (X'18')	24	Expanded parm list area
LSPAFLAG2		1	Parameter flag byte 2
LSPAXMSG		1...	Extended expanded message area requested (XEXPMSG)
LSPAEDAT		.1..	Expanded output data area requested (EXPDATA)
*		..x.	Unused
The next five flags pertain to the DATATYPE keyword values.:			
LSPAFRSI		...1	Volume is specified. the return of volume free space information is requested
LSPAFRVT		.... 1...	Vtoc is specified. the return of vtoc free space information is requested
LSPAFRVX		.... .1..	Index is specified. the return of index free space information is requested
LSPAFRFI		.... ..1.	Fragindex is specified. the return of the fragmentation index is requested
LSPAFALL		1	All is specified or defaulted. the return of all free space information is requested, including the frag index
LSPAFLAG3	25 (X'19')	1	Parameter flag byte 3
LSPAENQH		1...	ENQHELD=YES was specified by the caller
LSPATIMS		.1..	TIMEOUT= was specified by the caller
LSPARSV2	25 (X'19')	..xx xxxx	Reserved (unused) bits in LSPAFLAG3
LSPARES2	26 (X'1A')	2	Reserved
LSPATIMV	28(X'1C')	4	TIMEOUT value in seconds
LSPARES3	32(x'20')	16	Reserved

Return Codes from LSPACE

Control returns to the instruction following the instructions generated by the LSPACE macro.

See Table 8 on page 34 for a description of the LSPACE return codes.

LSPACE returns four bytes of diagnostic information in register 0. See the "DADSM/CVAF Diagnostic Aids" section of *z/OS DFSMSdfp Diagnosis* for a description of this information.

LSPACE Information Return Areas

The LSPACE macro returns status information to the parameter list and, optionally, returns volume information to any of the four following caller requested return areas.

Requests for the MSG, EXPMSG, XEXPMSG, DATA, and EXPDATA areas are mutually exclusive. LSPACE checks to ensure that the storage key of each information return area is equal to the caller's key or that the caller is authorized prior to its use.

Message Return Area

LSPACE returns information to a 30-byte message return area. If you provide a message return area with the MSG option, LSPACE returns EBCDIC text, qualified by return codes as shown in Table 8 on page 34.

```
LSPMSG   DSECT           Message Area
LSPMTEXT DS    CL30      Message Text
```

Table 8. DADSM LSPACE Message Return Area Contents

Return Code	Description
0(X'00')	For MSG keyword (30 byte area): Text: SPACE=*aaaa*,*bbbb*, *cccc*/*dddd*,*eeee* where: Free space statistics from the entire volume: ***aaaa*** = Total number of free cylinders ***bbbb*** = Total number of additional free tracks ***cccc*** = Total number of free extents ***dddd*** = Number of cylinders in largest free extent ***eeee*** = Number of additional tracks in largest free extent For *aaaa*, *bbbb*, and *cccc*, the maximum value indicated is 9999 even if the actual value is greater. Use EXPMSG to avoid use of the maximum 9999. **For EXPMSG keyword (40 byte area)**: Text: SPACE=*aaaaaa*,*bbbbbb*, *cccccc*/*dddddd*,*eeeeee* where: Free space statistics from the entire volume: ***aaaaaa*** = Total number of free cylinders ***bbbbbb*** = Total number of additional free tracks ***cccccc*** = Total number of free extents ***dddddd*** = Number of cylinders in largest free extent ***eeeeee*** = Number of additional tracks in largest free extent For *aaaaaa*, *bbbbbb*, *cccccc*, and *dddddd*, the maximum value indicated is 999999 even if the actual value is greater. Use XEXPMSG to avoid use of the maximum 999999. **For XEXPMSG keyword (95 byte area)**: Text: SPACE=*aaaaaaaaa*,*bbbbbbbbb*, *ccccccccc*/*ddddddddd*,*ee*, *fffffffff*,*ggggggggg*,*hhhhhhhhh*/*iiiiiiiii*,*jj* where: Free space statistics from the entire volume: ***aaaaaaaaa*** = Total number of free cylinders ***bbbbbbbbb*** = Total number of additional free tracks ***ccccccccc*** = Total number of free extents ***ddddddddd*** = Number of cylinders in largest free extent ***ee*** = Number of additional tracks in largest free extent Free space statistics from the track-managed space of the volume. For a volume without cylinder-managed space, these statistics will be equivalent to the total volume statistics above: ***fffffffff*** = Total number of free cylinders ***ggggggggg*** = Total number of additional free tracks ***hhhhhhhhh*** = Total number of free extents ***iiiiiiiii*** = Number of cylinders in largest free extent ***jj*** = Number of additional tracks in largest free extent Leading zeroes for each value will be set to 'blanks' in the returned message area.
4(X'04')	Text: LSPACE—PERMANENT I/O ERROR
4(X'04')	Text: LSPACE—I/O TIMEOUT ERROR

Table 8. DADSM LSPACE Message Return Area Contents (continued)

Return Code	Description
8(X'08')	Text: LSPACE—NON-STANDARD OS VOLUME. The volume does not have a VTOC index and free space information is not available. Either an operating system other than z/OS has allocated or freed space on the volume or the volume does not have a format 5 or 7 DSCB chain.
12(X'0C')	Text: LSPACE—UCB NOT READY
	Text: LSPACE—UCBVTOC IS ZERO
	Text: LSPACE—INVALID PARAMETER
	Text: LSPACE—NOT A DIRECT ACCESS VOL
16(X'10')	No text returned (invalid parameter list or SMF indicator) This return code indicates a parameter list error, which can be a bad parameter list storage key, parameter list ID is invalid (not set to 'LSPA'), or the parameter list size is not sufficient.
20 (X'14')	No text returned (processing error in LSPACE)

Expanded Message Return Area

LSPACE returns information to a 40-byte expanded message return area (Figure 5 on page 35). By providing an expanded message return area with the EXPMSG option, LSPACE returns EBCDIC text, qualified by return codes as shown in Table 8 on page 34.

Even though the expanded message return area displays the same return code information as the standard message return area, the space information provided for return code zero consists of six-digit values (*aaaaaa* instead of *aaaa*).

```
LSPMSG   DSECT          Expanded Message Area
LSPETEXT DS    CL40     Expanded Message Text
```

Figure 5. DADSM LSPACE free space information format, MF=(D,EXPMSG)

Data Return Area

If you provide a data return area with the DATA or EXPDATA keywords, LSPACE returns the information shown below. The expanded data return area is provided only when EXPDATA is specified. EXPDATA is the recommended keyword for returning binary data from LSPACE.

Table 9. LSPACE Data Return Area Format

Name	Offset	Bytes	Description
LSPDRETN	0(X'00')	1	Return area status byte
LSPDSPAC		1	Returned space information
LSPDF0CN		. 1	Returned F0 DSCB count
LSPDVRCN		. . 1	Returned free VIR count
LSPDFRGI		. . . 1 . . .	Returned fragmentation index
LSPDCYLM		. . . . 1 . .	Returned data is for a volume with cyl-managed space
LSPDRRES		. . . . . x x x	Reserved
LSPDSTAT	1(X'01')	1	Volume status byte
LSPDIXDS		1	Index exists for VTOC
LSPDIXAC		. 1	Index VTOC active
LSPDSRES		. . x x x x x x	Reserved
LSPDRSV1	2(X'02')	2	Reserved

Beginning of the base data return area: For DATA and EXPDATA requests, the following 32 bytes if requested or defaulted by the DATATYPE keyword will be returned. They represent statistics that describe the entire volume. LSPDFOS and LSPDVIRS are not applicable to volume statistics.

Table 9. LSPACE Data Return Area Format (continued)

Name	Offset	Bytes	Description
LSPDEVFS: Free space statistics from the entire volume. For volumes with cylinder-managed space (LSPDCYLM = '1') these statistics represent space from both the track and cylinder- managed space on the volume. See LSPDTMFS for statistics from the track-managed space on the volume.			
LSPDEVFS	4(X'04')	20	Total volume free space
LSPDNEXT	4(X'04')	4	Number of free extents
LSPDTCYL	8(X'08')	4	Total free cylinders
LSPDTTRK	12(X'0C')	4	Total addition free tracks
LSPDLCYL	16(X'10')	4	Number of cylinders in largest free extent
LSPDLTRK	20(X'14)'	4	Number of additional tracks in largest free extent
LSPDF0S	24(X'18')	4	Format 0 count
LSPDVIRS	28(X'1C')	4	Free VIR count
LSPDFRAG	32(X'20')	4	Fragmentation count
End of the base data return area.			
Beginning of the expanded data return area: For EXPDATA requests, the following 32 bytes of statistics if requested or defaulted by the DATATYPE keyword will be returned.			
LSPDTMFS: Free space statistics from track-managed space on a volume for a volume with cylinder-managed space (LSPDCYLM ='1'). For volumes with no cylinder-managed space (LSPDCYLM ='0') than these statistics are equivalent to the total volume statistics described above.			
LSPDTMFS	36(X'24')	24	Track-managed free space
LSPDVNXT	36(X'24')	4	Number of free extents
LSPDVTCL	40(X'28')	4	Total free cylinders
LSPDVTTK	44(X'2C'	4	Total additional free cylinders
LSPDVLCL	48(X'30')	4	Number of cylinders in the largest free extent
LSPDVLTK	52(X'34')	4	Number of additional free tracks in the largest free extent
LSPDVFRG	56(X'38')	4	Fragmentation index
Volume size information			
LSPDVOLI	60(X'3C')	8	Volume size information
LSPDTRKS	60(X'3C')	4	Total number of tracks on volume
LSPDTRKM	64(X'40'	4	Total number of tracks in track-managed space when LSPDCYLM='1'. Set to the value of LSPDTRKS otherwise. When LSPDCYLM='1', this value is also the first track address where cylinder managed space begins.
LSPDRSV8	68(X'44')	60	Reserved

Format-4 DSCB Return Area

If you provide a format-4 DSCB return area with the F4DSCB option, LSPACE returns information to it as described in Table 4 on page 12.

LSPACE Examples

The following example returns free space information in the message return area.

```
LSPAMFIM LSPACE MSG=MYMSG,UCB=(R10),MF=I
```

The following example returns free space information in the data return area.

```
LSPAMFID LSPACE DATA=MYDATA,UCB=(R10),MF=I
```

The following example uses the list form of the macro to define the parameter list, and the execute form to refer to the same parameter list.

```
LSPALIST LSPACE MSG=MYDATA,MF=L
         .
         .
         .
LSPAEX   LSPACE MF=(E,LSPALIST),UCB=(R10)
```

Note: The LSPACE macro is in SYS1.MODGEN

Reading DSCBs from the VTOC Using OBTAIN

The following section discusses using the OBTAIN routine to read a DSCB. You can specify either the data set name or the absolute device address.

OBTAIN does not support z/OS UNIX files. You will receive unpredictable results if you issue OBTAIN for an z/OS UNIX file.

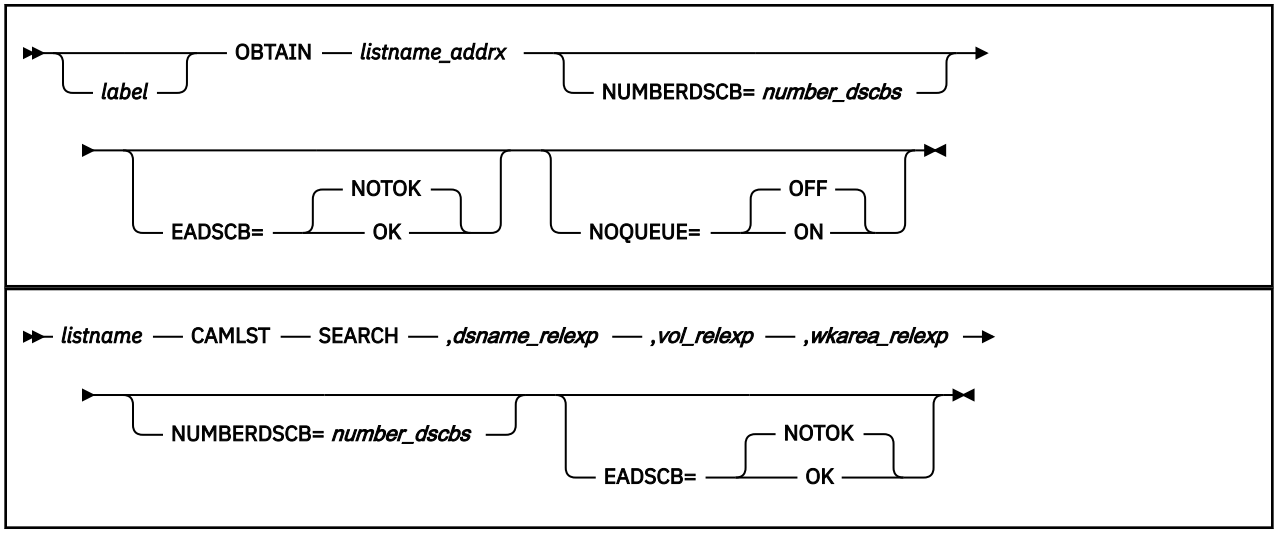
Reading a DSCB by Data Set Name

If you specify a data set name using OBTAIN and the CAMLST SEARCH option, the OBTAIN routine reads the 96-byte data portion of the format-1 DSCB and the absolute track address of the DSCB into virtual storage. The absolute track address is a 5-byte field in the form CCHHR that contains zeros for VIO data sets.

If the data set name is the name of a VSAM cluster that has a different name in the VTOC, then the system returns an emulated format-1 or 8 DSCB. It does not describe more than three extents and the system does not return a second DSCB. It is a format-8 DSCB only if any of the first three extents has a cylinder number greater than 32752. That is possible only on an EAV (extended addressing volume). If the emulated DSCB is format-8, then your program must specify EADSCB=OK, or OBTAIN gives return code 24.

Note: A format-1 or format-8 DSCB holds a volume serial number. The field name is DS1DSSN. This field identifies the volume during data set creation. The value in this field (that is, the volume serial number) is not updated if the volume is relabeled after data set creation, and can thus hold an invalid value. In addition, some data set copying programs propagate this field to the new data set, where it no longer identifies the real volume. Another case where this field does not identify the current volume is in a multi-volume data set. Normally the system propagates this field from the first volume to the subsequent volumes. If the data set is cataloged, you can use the LOCATE macro or Catalog Search Interface (CSI) to get a correct list of volumes that the data set resides on. All data sets that are managed by SMS are cataloged and most data sets on most systems are SMS-managed. Data sets that are not SMS-managed generally are cataloged but it is not required.

The format of the OBTAIN and CAMLST macros is:



listname_addrx
Points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

SEARCH
Code this operand as shown.

dsname_relexp
Specifies the virtual storage location of a fully-qualified data set name. The area that contains the name must be 44 bytes long.

Tip: A DSNAME of 44 bytes of X'04' (X'040404...04') can be used to read a format-4 DSCB.

vol_relexp
Specifies the virtual storage location of the 6-byte volume serial number on which the DSCB is located.

wkarea_relexp
Specifies the virtual storage location of a 140-byte work area that you must define.

NUMBERDSCB
Specifies a value between 0 and 255 that designates the number of consecutive 140-byte return areas that are provided in wkarea_relexp. The system treats a value of 0 as a 1. Currently, the system does not support a chain of more than 12 DSCBs for one data set, but it is valid for you to provide an area that is longer than currently needed. The system verifies that the provided area is valid. When you provide an area that is long enough to contain more than one DSCB, obtain processing returns DSCBs for the requested data set name in logical VTOC order until all the 140-byte return areas are used. The logical VTOC order is a format-1 DSCB, followed by zero or more format-3 DSCBs or a format-8 DSCB, followed by one or more format-9 DSCBs, followed by zero or more format-3 DSCBs. No absolute maximum number of DSCBs for a data set should be assumed. The actual number of DSCBs are returned in a field located in the first 140-byte return area.

On the OBTAIN macro, you can code a register number or symbol for a register number in parentheses. It means that the specified register contains the number of DSCBs that can fit in the return area. If you code the NUMBERDSCB parameter on OBTAIN, the macro execution stores the value in the CAMLST area. You cannot code a register on the CAMLST macro.

Note that for programs run on a pre-z/OS R10 system that do not support this keyword, the NUMBERDSCB value is treated as if it were 1.

EADSCB
Specifies whether this program supports data sets with format-8 and format-9 DSCBs. Such data sets can appear on extended address volumes.

EADSCB=OK
Code EADSCB=N0T0K when your program supports data sets that have format-8 and format-9 DSCBs. The extent descriptors in DSCBs for a data set described with these formats might have

track addresses that contain cylinder addresses 65,520 or larger. EADSCB=OK is accepted for data sets described by all DSCB types, including format-1 DSCBs, regardless of the volume size where the data set resides. Your program can also run on an older level of the system that does not support this keyword. In these cases, EADSCB=OK is ignored. EADSCB=OK sets byte 2 bit 4 in the OBTAIN parameter list to on.

EADSCB=NOTOK
Code EADSCB=NOTOK when your program does ***not*** support DSCBs that describe data sets with format-8 and format-9 DSCBs. EADSCB=NOTOK is the default when the EADSCB keyword is not specified.

When EADSCB=NOTOK is coded or assumed by default, OBTAIN sets return code 24 if the target of the OBTAIN request has a format-8 DSCB.

EADSCB=NOTOK Sets byte 2 bit 4 in the OBTAIN parameter list to zero.

NOQUEUE=OFF
Specifies that the caller wants to wait on a resource (volume) if no paths are available.

NOQUEUE=ON
Specifies that the caller wants to have a no path situation terminated instead of waiting for the resource (volume) to become available.

Example

In the following example, the format-1 DSCB for data set A.B.C is read into virtual storage using the SEARCH option. The serial number of the volume containing the DSCB is 770655.

```
          OBTAIN    DSCBABC           READ DSCB FOR DATA
*                                     SET A.B.C INTO DATA
*                                     AREA NAMED WORKAREA

DSCBABC   CAMLST    SEARCH,DSABC,VOLNUM,WORKAREA
DSABC     DC        CL44'A.B.C'       DATA SET NAME
VOLNUM    DC        CL6'770655'       VOLUME SERIAL NUMBER
WORKAREA  DS        140C              140-BYTE WORK AREA
```

Recommendation: Check the return codes.

The OBTAIN macro instruction points to the CAMLST parameter list. SEARCH, the first operand of CAMLST, specifies that a DSCB be read into virtual storage using the data set name at the address indicated in the second operand. DSABC specifies the virtual storage location of a 44-byte area containing the fully-qualified name of the data set whose format-1 DSCB is to be read. VOLNUM specifies the virtual storage location of a 6-byte area in which you have placed the serial number of the volume containing the required DSCB. WORKAREA specifies the virtual storage location of a 140-byte work area into which the DSCB is to be returned.

Control is returned to your program at the next executable instruction following the OBTAIN macro instruction. If the DSCB has been successfully read into your work area, register 15 contains zeros. Otherwise, register 15 contains one of the return codes shown in Table 10 on page 39.

Return Codes from OBTAIN (Reading by Data Set Name)

Table 10. DADSM OBTAIN SEARCH Return Codes

Return Code	Description
0(X'00')	Successful completion of OBTAIN routine.
4(X'04')	The required volume was not mounted.
8(X'08')	The format-1 DSCB was not found in the VTOC of the specified volume.
12(X'0C')	A permanent I/O error was encountered, or an invalid format-1 DSCB was found when processing the specified volume, or an unexpected error return code was received from CVAF (common VTOC access facility).
16(X'10')	Invalid work area pointer.

Table 10. DADSM OBTAIN SEARCH Return Codes (continued)

Return Code	Description
24(X'18')	Data set has a format-8 DSCB and EADSCB=NOTOK is in effect
28(X'1C')	Internal error with EADSCB=NOTOK in effect.

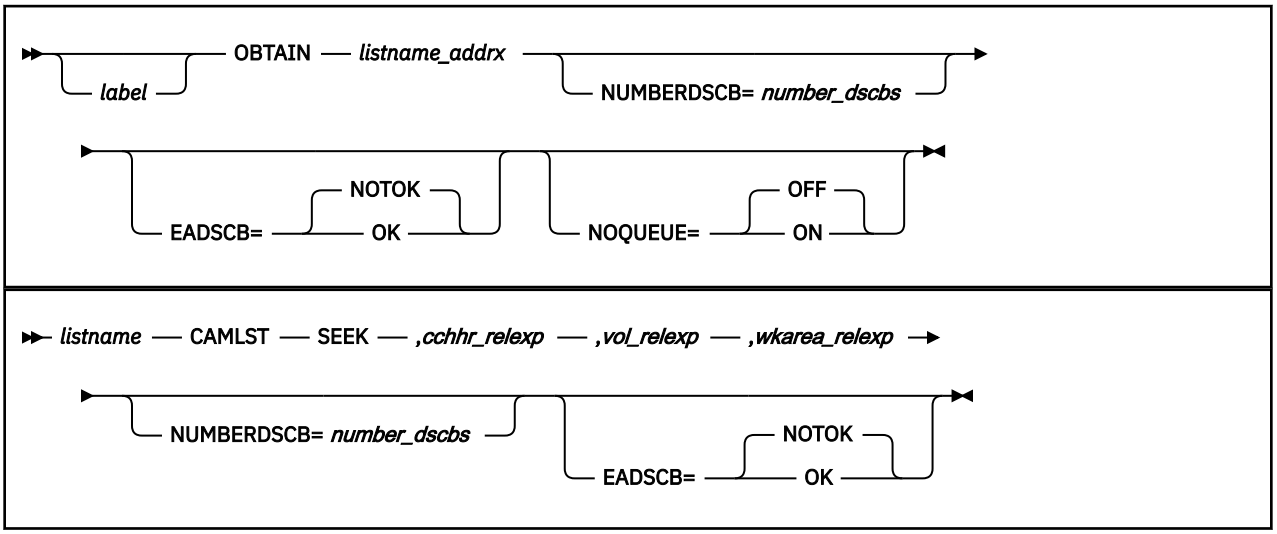
On return from SEARCH requests, the first 103 bytes of the first 140 byte return area will contain:

- The 96 byte data portion of the format-1, format-4, or format-8 DSCB
- Followed by 5 bytes that contain the absolute track address (CCHHR) of this DSCB. For VSAM object names that appear in the catalog, but not in the VTOC or VIO data sets, these 5 bytes contain zeros.
- Followed by a 2 byte count of the total number of DSCBs associated with the data set DSCB, even if there are insufficient return areas into which to read them all. This count includes all DSCB types that could describe a data set. Set to zero for DSCBs constructed from catalog when DSCBs are not present in the VTOC.

Reading a DSCB by Absolute Device Address

You can read a DSCB from a VTOC using OBTAIN and the CAMLST SEEK option. Specify the SEEK option by coding SEEK as the first operand of the CAMLST macro and by providing the absolute device address of the DSCB you want to read, unless the DSCB is for a VIO data set. Only the SEARCH option can be used to read the DSCB of a VIO data set.

The format of the OBTAIN and CAMLST macros is:



listname
Label of the CAMLST macro instruction.

SEEK
Code this operand as shown when calling obtain to read by the passed CCHHR device address..

cchhr_relexp
For SEEK requests, specifies the virtual storage location of the 5-byte absolute device address (CCHHR) of a DSCB.

vol_relexp
Specifies the virtual storage location of the 6-byte volume serial number on which the DSCB is located.

wkarea_relexp
For all requests, specifies the virtual storage location of a 140-byte work area, or larger, that must be defined.

NUMBERDSCB
Specifies a value between 0 and 255 that designates the number of consecutive 140-byte return areas that are provided in *wkarea_relexp*. The system treats a value of 0 as a 1. Currently, the system does not support a chain of more than 12 DSCBs for one data set, but it is valid for you to provide an area that is longer than currently needed. The system verifies that the provided area is valid. When you provide an area that is long enough to contain more than one DSCB, OBTAIN processing returns DSCBs for the requested data set name in logical VTOC order until all the 140-byte return areas are used. The logical VTOC order is a format-1 DSCB, followed by zero or more format-3 DSCBs or a format-8 DSCB, followed by one or more format-9 DSCBs, followed by zero or more format-3 DSCBs. No absolute maximum number of DSCBs for a data set should be assumed. When the target of the seek operation is not a format-1 or format-8 DSCB, the NUMBERDSCB value is treated as if it were 1 and only that single DSCB is returned.

On the OBTAIN macro, you can code a register number or symbol for a register number in parentheses. This means that the specified register contains the number of DSCBs that can fit in the return area. If you code the NUMBERDSCB parameter on OBTAIN, the macro execution stores the value in the CAMLST area. You cannot code a register on the CAMLST macro.

Note that for programs run on a pre-z/OS R10 system that do not support this keyword, the NUMBERDSCB value is treated as if it were 1.

EADSCB
Specifies whether this program supports data sets with format-8 and format-9 DSCBs. Such data sets can appear on extended address volumes.

EADSCB=OK
Code EADSCB=NOTOK when your program does not support data sets that have format-8 and format-9 DSCBs. The extent descriptors in DSCBs for a data set described with these formats might have track addresses that contain 28-bit cylinder numbers. EADSCB=OK is accepted for data sets described by all DSCB types, including format-1 DSCBs, regardless of the volume size where the data set resides. Your program can also run on an older level of the system that does not support this keyword. In these cases, EADSCB=OK is ignored. EADSCB=OK sets byte 2 bit 4 in the OBTAIN parameter list to on.

EADSCB=NOTOK
Code EADSCB=NOTOK when your program does not support DSCBs that describe data sets with format-8 and format-9 DSCBs. EADSCB=NOTOK is the default when the EADSCB keyword is not specified.

When EADSCB=NOTOK is coded or assumed by default, OBTAIN sets return code 24 if the target of the OBTAIN request has a format-8 or format-9 DSCB. OBTAIN does not check format-3 DSCB extent ranges for track addresses that contain 28-bit cylinder numbers.

EADSCB=NOTOK Sets byte 2 bit 4 in the OBTAIN parameter list to zero.

NOQUEUE=OFF
Specifies that the I/O to read DSCBs does not queue on the resource and wait if that resource is not available.

Example

In the following example, the DSCB at actual-device address X'00 00 00 01 07' is returned in the virtual storage location READAREA, using the SEEK option. The DSCB resides on the volume with the volume serial number 108745.

```
          OBTAIN   ACTADDR         READ DSCB FROM
*                                  LOCATION SHOWN IN CCHHR
*                                  INTO STORAGE AT LOCATION
*                                  NAMED READAREA

ACTADDR   CAMLST   SEEK,CCHHR,VOLSER,READAREA
CCHHR     DC       XL5'0000000107' ABSOLUTE TRACK ADDRESS
VOLSER    DC       CL6'108745'     VOLUME SERIAL NUMBER
READAREA  DS       140C            140-BYTE WORK AREA
```

Recommendation: Check the return codes.

The OBTAIN macro points to the CAMLST parameter list. SEEK, the first operand of CAMLST, specifies that a DSCB be read into virtual storage. CCHHR specifies the storage location containing the 5-byte actual-device address of the DSCB. VOLSER specifies the storage location containing the serial number of the volume where the DSCB resides. READAREA specifies the storage location to which the 140-byte DSCB is to be returned.

Control is returned to your program at the next executable instruction following the OBTAIN macro instruction. If the DSCB has been read into your work area, register 15 contains zeros. Otherwise, register 15 contains one of the return codes shown in Table 11 on page 42.

Return Codes from OBTAIN (Reading by Absolute Device Address)

Table 11. DADSM OBTAIN SEEK Return Codes

Return Code	Description
0(X'00')	Successful completion of OBTAIN routine.
4(X'04')	The required volume was not mounted.
12(X'0C')	A permanent I/O error was encountered or an unexpected error return code was received from CVAF.
16(X'10')	Invalid work area pointer.
20(X'14')	The SEEK option was specified and the absolute track address (CCHHR) is not within the boundaries of the VTOC.
24(X'18')	Data set has a format-8 or format-9 DSCB and EADSCB=NOTOK is in effect
28(X'1C')	Internal error with EADSCB=NOTOK in effect.

On return from SEEK requests with multiple DSCBs requested, the entire 140 bytes of the first return area will contain the 140 byte key and data portions of the DSCBs read from the volume. The remaining 140 byte return areas will contain the entire 140 byte key and data portions of the DSCBs chained from this first DSCB until the chain of DSCBs has ended or until the 140 byte return areas are exhausted.

On return from SEEK requests without multiple DSCBs specified, the first return area will contain the entire 140 byte key and data portions of the DSCB read from the user's passed device address.

Releasing Unused Space from a DASD Data Set Using PARTREL

DADSM supports the release of unused space that is allocated to sequential or partitioned data sets, PDSEs, sequential extended-format data sets, and VSAM extended format data sets.The partial release function is called when:

- The data set is closed (if the RLSE subparameter of SPACE was specified on its DD statement or the storage administrator specified an appropriate value for the partial release option in the management class definition).
- A restart is processing from a checkpoint taken before the data set was extended.
- DFSMShsm performs a space management cycle and an appropriate value is specified in the management class definition.
- A PARTREL macro is issued.

For an extended address volume, partial release processing will release space on multicylinder unit boundaries when the last used track is in an extent in cylinder-managed space. For VSAM striped data sets where at least one stripe is on an extended address volume, partial release processing will ensure the stripes remain the same size. For these reasons, it is possible that the high allocated track after the release may be larger than the last used track or that no space could be released.

The PARTREL Macro

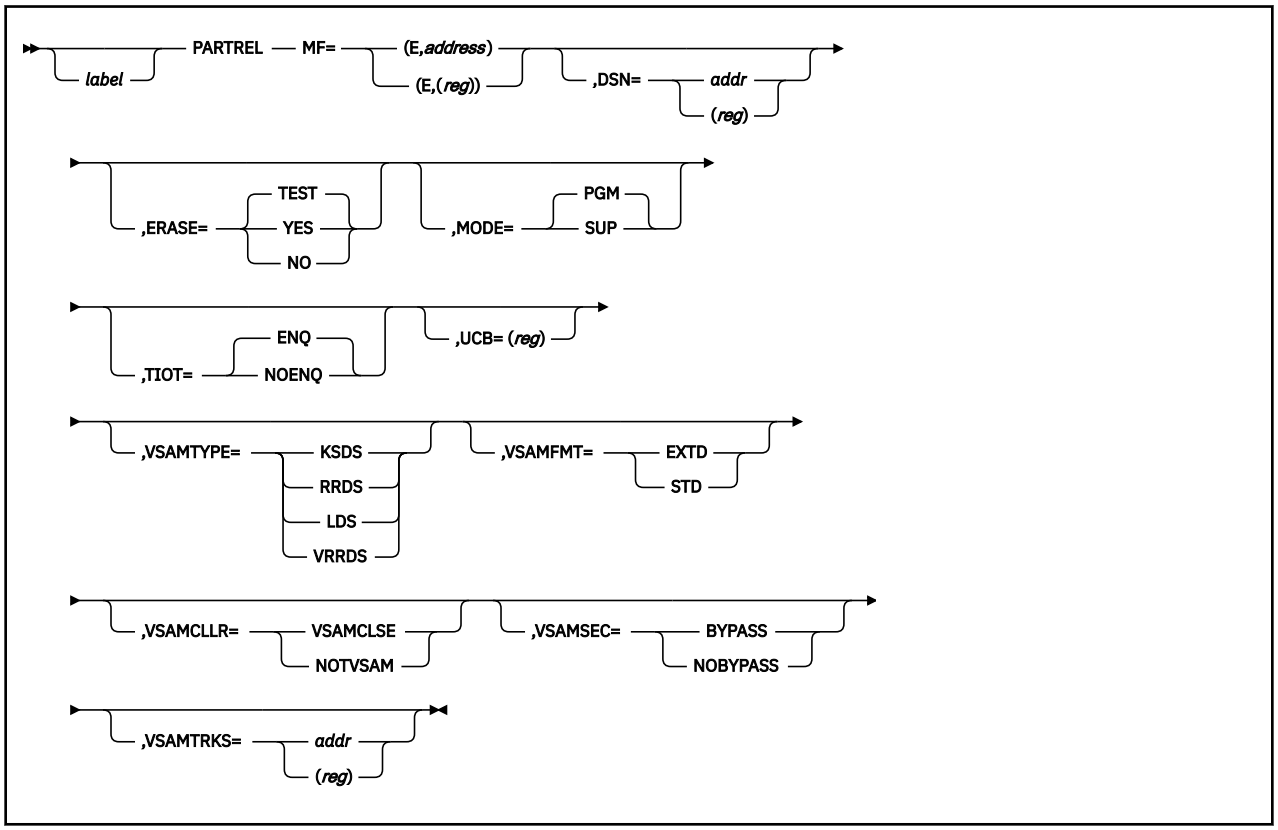
The PARTREL macro supports sequential and partitioned data sets on volumes with or without indexed VTOCs. It supports PDSEs and extended format data sets on SMS-managed volumes for which indexed VTOCs are required. PARTREL does not support z/OS UNIX files. You will receive unpredictable results if you issue PARTREL for z/OS UNIX files.

PARTREL can be coded in the execute, DSECT, and list forms, but not the standard form. The calling program:

- Must be APF authorized.
- Must have allocated the volume to this task and must ensure it stays mounted during the PARTREL function.
- Must ensure that the data set is not open.
- Must not hold any locks or an ENQ on the VTOC.
- Must provide the address of an available standard register save area in general register 13.
- Can provide the associated parameter list and parameters in storage either above or below 16MB virtual.
- Can be in any storage key.
- Can run in either supervisor or problem program state.
- Can be in either 24 or 31-bit addressing mode.
- Requires the address of a UCB, not a UCB copy.

PARTREL–Execute Form

The format of the execute form of the PARTREL macro is:



Except for MODE, all parameters default to the current contents of the parameter list. The MODE parameter defaults to PGM.

Descriptions of these parameters include information about DADSM processing. The descriptions use the term *value* to designate the parameter value passed to DADSM. The value can be:

- Specified as a parameter on the PARTREL macro
- Provided as the parameter's associated value in the parameter list
- Defined by DADSM from the information provided in the request.

MF=(E,*addr*) or (E,(*reg*))
Specifies the execute form of the macro and the address of an existing PARTREL parameter list.

***addr*—RX-type address, (*reg*)—(0-12)**
Specifies the PARTREL parameter list address.

DSN=*addr* or (*reg*)
Specifies the address of a 44-byte area that contains the data set name. The name must be left-justified, with any unused bytes defined as blanks. For a VSAM file, you must specify the component name, not the cluster name.

***addr*—RX-type address, (*reg*)—(0), (2-12)**
You must provide a value for DSN.

ERASE=YES or NO or TEST
Specifies a residual data erase attribute (see "Deleting a Data Set from the VTOC" on page 133 for a description of erase attributes). ERASE=YES and ERASE=NO are mutually exclusive. The default is ERASE=TEST. If you specify VSAMTYPE, ERASE is ignored.

ERASE=YES
Specifies that the area being released should be erased (overwritten with zeros) before it is made available for new allocations.

ERASE=NO
Specifies that the area should not be erased. This specification overrides the RACF® erase attribute.

ERASE=TEST
Specifies that the associated RACF erase attribute is to be used.

MODE=PGM or SUP
Specifies that PARTREL is requested by a caller in problem program state (MODE=PGM) or in supervisor state (MODE=SUP). MODE=PGM is the default.

If the calling program is in supervisor state (and wants control returned in supervisor state), the value of MODE must be SUP. If the calling program is in problem program state, the value of MODE must be PGM.

TIOT=ENQ or NOENQ
Specifies the desired SYSZTIOT and SYSDSN ENQ processing within partial release. The default is ENQ. If you specify VSAMTYPE, TIOT is ignored.

TIOT=ENQ
Specifies that partial release is to do its normal, exclusive ENQ on SYSZTIOT and SYSDSN. If either of these ENQ requests fails, PARTREL terminates the request with a return code of X'08'.

TIOT=NOENQ
Specifies that the caller has provided the necessary serialization. If partial release finds that the caller does not have exclusive use of SYSDSN, PARTREL terminates the request with a return code of X'24'.

When TIOT=NOENQ is specified and flag PRLTIOTX is set on, the ENQ TEST is not executed. This is a dangerous option and can cause data corruption. IBM recommends that you not set PRLTIOTX.

UCB=(*reg*)
Specifies the address of the UCB for the volume on which the subject data set resides. The UCB address can be for a captured UCB, or for an actual UCB above or below the 16MB line. For 31-bit callers, the high-order byte is part of the UCB address and must be cleared to zeros if a 24-bit

UCB address is being passed. The volume must be mounted, and you must ensure that it remains mounted. Use the address of a UCB, not a UCB copy.

(*reg*)–(0), (2-12)
specifies a register containing the UCB address for the device.

VSAMTYPE=KSDS or ESDS or RRDS or LDS or VRRDS
Specifies the type of VSAM data set.

Currently, only extended format VSAM data sets are supported.

If you specify VSAMTYPE, you *must* also specify VSAMFMT, VSAMCLLR, VSAMSEC, and VSAMTRKS.

If you specify VSAMTYPE, the ERASE= parameter is ignored.

VSAMTYPE=KSDS
Specifies a VSAM key-sequenced data set.

VSAMTYPE=ESDS
Specifies a VSAM entry-sequenced data set.

VSAMTYPE=RRDS
Specifies a VSAM fixed-length relative record data set.

VSAMTYPE=LDS
Specifies a VSAM linear data set.

VSAMTYPE=VRRDS
Specifies a VSAM variable-length relative record data set.

VSAMFMT=EXTD or STD
Specifies the format of the data set.

VSAMFMT=EXTD
Specifies that this is an extended format data set.

VSAMFMT=STD
Specifies that this is *not* an extended format data set. If you specify STD catalog services returns an error to partial release.

VSAMCLLR=VSAMCLSE or NOTVSAM
Specifies the caller.

VSAMCLLR=VSAMCLSE
Specifies that the caller is VSAM CLOSE.

VSAMCLLR=NOTVSAM
Specifies that the caller is other than VSAM CLOSE.

VSAMSEC=BYPASS or NOBYPASS
Specifies if security checking is to be performed

VSAMSEC=BYPASS
Specifies that security checking is *not* to be performed.

VSAMSEC=NOBYPASS
Specifies that security checking is to be performed.

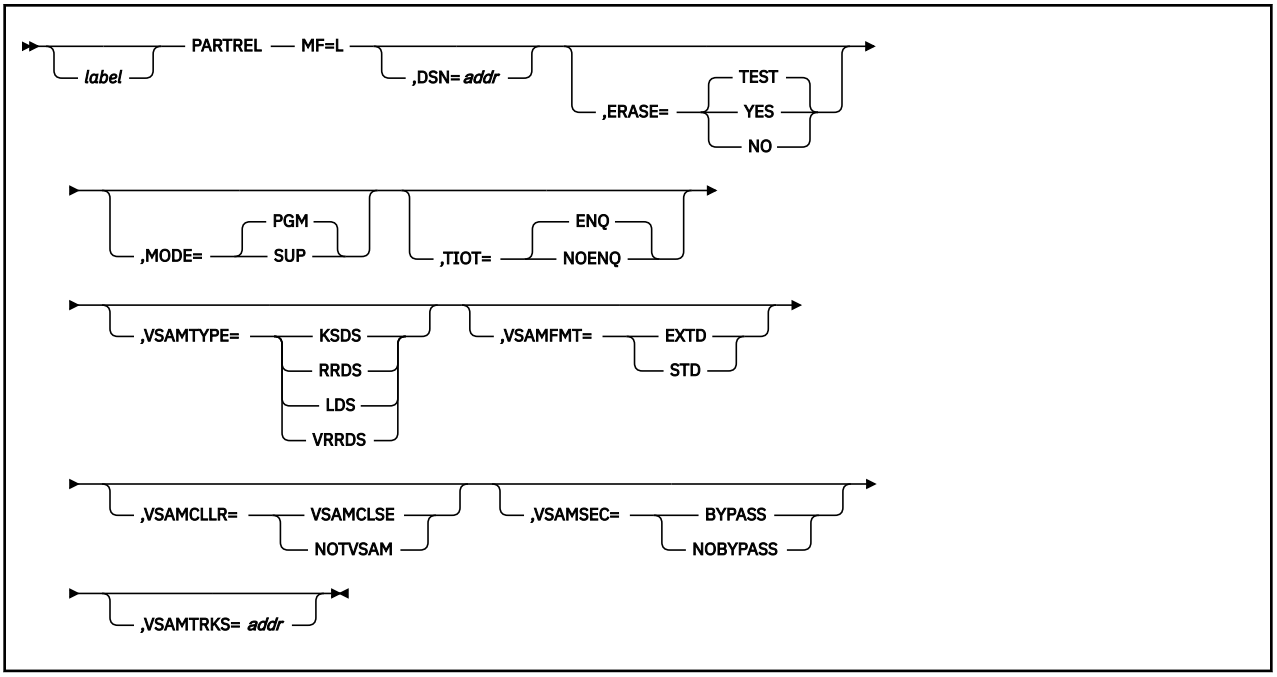
VSAMTRKS=*addr* or (*reg*)
Specifies the address of a 4-byte area that is used to contain the number of tracks released for this data set.

You must provide an address for VSAMTRKS only if you code VSAMTYPE.

***addr*—RX-type address, (*reg*)—(0), (2-12)**
For a VSAM extended format data set, PARTREL returns a value which is the sum of all the space released for all the parts of the data set.

PARTREL—List Form

The format of the list form of the PARTREL macro is:



Restrictions:

1. The execute form of the UCB parameter cannot be specified on the list form.
2. The list form MODE parameter is for documentation only. The value of MODE is as specified or defaulted on the execute form.

For an explanation of the parameters, see the execute form.

PARTREL–DSECT Form

The format of the DSECT form of the PARTREL macro is:

Description

The following example provides a description of the parameter list:

```
++PRLPLID  DS    CL4     EBCDIC 'PREL' FOR PARTREL
+PRLNGTH  DS    AL2     LENGTH OF PARAMETER LIST
+PRERRCDE DS    H       ERROR CODE RETURNED FROM
++                        PARTIAL RELEASE
+PRLFLAG  DS    XL1     PARAMETER FLAG BYTE
+PRLPGM   EQU   X'00'   MODE=PGM (PROBLEM PROGRAM)
+PRLSUP   EQU   X'80'   MODE=SUP (SUPERVISOR STATE)
+PRLTIOT  EQU   X'40'   TIOT=NOENQ
+PRLNERAS EQU   X'20'   ERASE=NO
+PRLERASE EQU   X'10'   ERASE=YES
+PRLVCLOS EQU X'08'     VSAMCLLR=VSAMCLSE
+PRLVCNV  EQU X'04'     VSAMCLLR=NOTVSAM
+PRLBYPS  EQU X'02'     VSAMSEC=BYPASS
+PRLNBYPS EQU X'01'     VSAMSEC=NOBYPASS
+PRLFLAG2 DS XL1
+PRLVKSDS EQU X'80'     VSAMTYPE=KSDS
+PRLVESDS EQU X'40'     VSAMTYPE=ESDS
+PRLVRRDS EQU X'20'     VSAMTYPE=RRDS
+PRLVVRRD EQU X'10'     VSAMTYPE=VRRDS
+PRLVLDS  EQU X'08'     VSAMTYPE=LDS
+PRLEXTD  EQU X'04'     VSAMFMT=EXTD
+PRLSTD   EQU X'02'     VSAMFMT=STD
+PRLTIOTX EQU X'01'     No ENQ TEST when TIOT=NOENQ is specified
+PRLRSVD  DS XL2        RESERVED
```

```
+PRLDSN   DS A          DATA SET NAME POINTER
+PRLUCB   DS A          UCB POINTER
+PRLTRKS  DS A          ADDRESS OF NUMBER OF TRACKS RETURNED
                         (ONLY VALID FOR VSAM REQUESTS)
+PRLCTGR  DS F          CATALOG REASON CODE
+PRLDDIG  DS F          DADSM DIAGNOSTIC INFORMATION
+PRLEND   EQU   *       END OF PARAMETER LIST
+PRLENGTH EQU   PRLEND-PRELPL       LENGTH OF PARAMETER LIST
```

Return Codes From PARTREL

Control returns to the instruction following the last instruction generated by the PARTREL macro. Register 15 contains the applicable PARTREL return code.

If an error occurs, PARTREL issues message IEC614I, consisting of failure-related status information.

- If an error results from a CVAF function, the subfunction return code field contains the CVAF return code, and the subfunction reason code field contains the CVAF status code (CVSTAT).
- If an error results from the execution of an EXCP channel program, the subfunction return code and reason code fields contain either:
  - The ECB completion code and the CSW channel status, if the ECB completion code is not X'41' and the channel status is not zero.
  - The sense bytes (two) from the IOB, if the ECB completion code is X'41' and there is no channel status.
- If an error results from an RACF invocation, the subfunction return code and reason code fields contain the RACF return code and reason code.

Table 12 on page 47 describes the conditions indicated by the PARTREL return code.

Exception: This is a cumulative list of DADSM partial release return codes. Some of these codes might not apply to the PARTREL macro.

Table 12. DADSM PARTREL Return Codes

Return Code	Description
0(X'00')	Sucessful
2(X'02')	Unable to find extent in format-1 or format-8 DSCB.
3(X'03')	Exceeded maximum number of format-3 pointers in format-9 DSCB.
4(X'04')	Unable to find extent in format-3 DSCB.
8(X'08')	Either the required SYSZTIOT or SYSDSN ENQ failed, or an unrelated DEB indicates that another DCB is open to the data set.
12(X'0C')	Invalid parameter list.
16(X'10')	One of the following conditions occurred: • A permanent I/O error occurred. • CVAF provided an unexpected return code. • An installation exit rejected the request. • An I/O error occurred while the tracks to be released were being erased (for ERASE-on-SCRATCH).
20(X'14')	DSN, or DSN pointer is invalid.
24(X'18')	Invalid UCB pointer.
28(X'1C')	Specified DSORG is not supported.
32(X'20')	No room in the VTOC.
36(X'24')	Invalid TIOT=NOENQ request; caller does not have exclusive use of SYSDSN.
40(X'28')	An error occurred while SMS was processing the request.

Table 12. DADSM PARTREL Return Codes (continued)

Return Code	Description
44(X'2C')	CLOSE is the caller, user rejected the partial release using the PREEXIT routine.
48(X'30')	An error occurred during conversion from CCHH to relative track address.
52(X'34')	An error occurred during conversion from relative track address to CCHH.
56(X'38')	An error occurred during conversion from new extent descriptor table to old extent descriptor table.
60(X'3C')	An error occurred during sort conversion of the extent descriptor table.
64(X'40')	An error occurred during conversion from format-7 to extent descriptor table.
68(X'44')	An error occurred during conversion from format-5 to extent descriptor table.
72(X'48')	Input DSCB is not a format-5 or a format-7 DSCB.
76(X'4C')	An error occurred during conversion from extent descriptor table to format-7.
80(X'50')	An error occurred during conversion from extent descriptor table to format-5.
84(X'54')	VSAM standard format data set incorrectly specified. If the data set is VSAM, it must be extended format.
88(X'58')	VSAMTRKS address parameter must be specified when VSAMTYPE is specified.
92(X'5C')	The name of the specified VSAM data set is not a VSAM cluster data component.
96(X'60')	Catalog call to partial release failed.
100(X'64')	Request was directed to a VSAM data set defined with guaranteed space.
104(X'68')	Unexpected reason code received back from a call to catalog services.
108X'6C')	Unable to find enough F0s to complete the PARTREL request.
112(X'70')	DAPCCHH passed is in cylinder-managed space, but is not on a multi-cylinder unit boundary.
178 (X'B2')	Refer to message IDC3009I, return code 178 for possible causes. See *z/OS MVS System Messages, Vol 6 (GOS-IEA)* for more information.
184 (X'B8')	Refer to message IDC3009I, return code 184 for possible causes. See *z/OS MVS System Messages, Vol 6 (GOS-IEA)* for more information.

Creating (Allocating) a DASD Data Set Using REALLOC

The REALLOC macro builds a parameter list to allocate a new data set. You can code the macro in the execute, DSECT, and list forms, but not in the standard form. The calling program includes the following requirements and options:

- Must be APF authorized
- Must allocate the volume to this address space and must ensure it stays mounted during the REALLOC function
- Must not hold any locks
- Can provide the associated parameter list and parameters in storage either above or below 16MB
- Can use any storage key
- Can run in either supervisor or problem program state
- Can be in either 24 or 31-bit addressing mode
- Must note that REALLOC does not call RACF or catalog management
- Must note that REALLOC cannot create data sets on SMS-managed volumes (SMS-managed data sets can be created through JCL or dynamic allocation)
- Must note that REALLOC cannot create PDSEs, HFS data sets, or extended format data sets

- Requires the address of a UCB, not a UCB copy.

In addition, the calling program must provide the REALLOC macro with one or more model DSCBs. You can use the OBTAIN macro to get the DSCBs from other data sets and modify them for the request. DADSM uses these model DSCBs to validate the allocation request, and to construct the DSCBs written to the VTOC for the requested allocation.

The ALLOC parameter for the REALLOC macro defines the allocation request as either absolute (ABS) or movable (MOV).

The requested data set's allocation is not sensitive to its placement on the volume. This is not a reference to the format-1 DSCB bit DS1DSGU (unmovable bit), that can be either on or off in an ALLOC=MOV request's partial DSCB. That is, the data set can subsequently contain location-dependent information.

An absolute request is limited to a single volume with indexed VTOC support. An absolute request provides a set of allocation parameters, a full format-1 or format-8 DSCB, and an optional format-3 DSCB, that describe the space and attributes of the desired data set. An optional format-9 DSCBs can also be provided to pass additional attributes:

- Support is provided for, but not limited to, data sets with a user label extent.
- The number of extents to be allocated, and their absolute placement on the volume, are defined by the format-1 or format-8 DSCB and one (optional) format-3 DSCB.

For an absolute request, the following checks are performed prior to writing the passed DSCBs. The type of checks depend on the volume for which the REALLOC macro is issued.

- For any volume, a passed DSCB format must not describe a format that is not supported. The supported formats in REALLOC processing are 1, 3, 8, or 9.
- For an extended address volume, a passed format-1 DSCB must not describe extents that contain cylinder addresses larger than 65,519.
- A passed format-1, format-3, or format-8 DSCB must not describe extents that contain cylinder addresses larger than the highest cylinder address of the volume.
- For an extended address volume, a passed format-8 DSCB must not describe a data set organization that is not EAS eligible.
- For an extended address volume, a passed format-3 or format-8 DSCB must not describe an extent that begins on cylinder address 65519 or lower and that ends on cylinder address 65520 or higher.
- For a volume that is not an extended address volume, a passed format-8 or format-9 DSCB is not allowed.

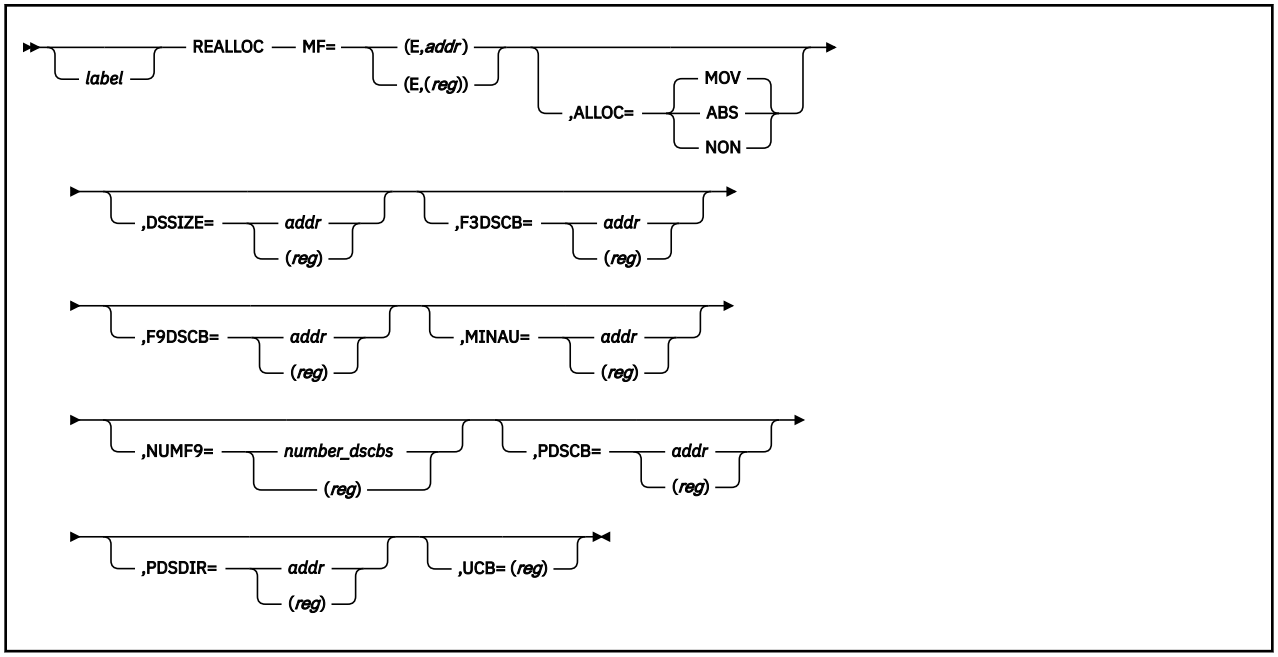
The partial DSCB (mapped by the IECPDSCB macro) consists of the first 106 bytes of a format-1 DSCB followed by 10 bytes in which the primary space request and number of directory blocks are specified.

A movable request is limited to a single volume with or without indexed VTOC support. A movable request provides a set of allocation parameters and a partial DSCB that describe the attributes of the desired data set:

- Absolute track allocated data sets are not supported.
- The maximum number of extents that can be allocated is determined by the data set organization (PD1DSORG) and the system managed storage indicators (PD1SMSFG) bytes in the partial DSCB.If PD1DSORG indicates a VSAM data set organization, the maximum number of extents is 7257 across multiple volumes, 123 on a single volume.

REALLOC–Execute Form

The format of the execute form of the REALLOC macro is:



All parameters except ALLOC default to the current contents of the referenced parameter list. The ALLOC parameter defaults to MOV.

MF=(E,*addr*) or (E,(*reg*))
Specifies the execute form of the macro and the address of a REALLOC parameter list.

***addr*–RX-type address, (*reg*)—(0-12)**
Specifies the address of the REALLOC parameter list.

ALLOC=ABS or MOV or NON
Specifies one of the following:

ALLOC=ABS
Specifies that the REALLOC request is for absolute extents.

ALLOC=MOV
Specifies that the REALLOC request is for a movable allocation. ALLOC=MOV is the default.

ALLOC=NON
Specifies that the REALLOC request is to rebuild the free space chain on an unindexed VTOC without allocating a data set. Expect a DADSM create return code X'3D'.

DSSIZE=*addr* or (*reg*)
Specifies the size of the data set to be allocated in tracks. The DSSIZE parameter is invalid for an ALLOC=ABS request.

***addr*–RX-type address**
Specifies the address of a word that contains the data set size.

(*reg*)–(0), (2-12)
Specifies a register that contains the size of the data set.

Provide a value for DSSIZE for an ALLOC=MOV request.The PDPRIQTY field of the partial DSCB is ignored.

REALLOC assumes that you have provided the value of DSSIZE in tracks even if the PD1SCALO flag byte of the partial DSCB indicates a cylinder request (X'C0'), or an average block request, (X'40').

If the PD1SCALO flag byte of the partial DSCB indicates a cylinder request (X'C0'), or an average block with round request (X'41'), the value of DSSIZE is rounded up to the next full cylinder.

F2DSCB
This parameter still is supported for assembly purposes, but the system no longer supports it when your program executes. This because you can no longer create indexed sequential data sets

F3DSCB=*addr* or (*reg*)
Specifies the in-storage address of a format-3 DSCB. This DSCB is used as a model to construct the allocated data set's format-3 DSCB.

The F3DSCB parameter is invalid for an ALLOC=MOV request.

addr–RX-type address, (*reg*)–(0), (2-12)

Provide a value for F3DSCB in an ALLOC=ABS request when the DS1NOEPV byte of the format-1 DSCB indicates more than three extents (or when the DS1NOEPV byte indicates more than two extents and the DS1EXT1 extent type indicator is X'40', a user label extent).

The REALLOC request is limited to a maximum of 16 extents when the F3DSCB keyword is specified. No more than one format-3 DSCB can be specified.

Enter a value of zero for the F3DSCB in an ALLOC=ABS request when the DS1NOEPV byte of the format-1 DSCB indicates that there are less than four extents (or when the DS1NOEPV byte indicates that there are less than three extents and the DS1EXT1 extent type indicator is X'40', a user label extent).

F9DSCB=*addr* or (*reg*) or 0
Specifies the address of a caller-provided contiguous partial format-9 DSCB data area where attribute information from it is to be used when creating a format-9 DSCB. Specify this keyword if you wish to pass format-9 DSCB attribute information to REALLOC processing. Only attribute information in the format-9 DSCB will be processed. Format-9 DSCBs with a subtype field with a value other than 1 will be ignored. See mapping macro, IECSDSL1. The number of contiguous partial format-9 DSCBs defined in this data area is defined in the NUMF9=*number_dscb* keyword or is defaulted to one.

addr-RX-type address
Specifies the address of the partial format-9 DSCB data area.

(reg)-(2-12)
Specifies a register containing the address of the partial format-9 DSCB data area.

0
Specifies that you do not want to pass a partial format-9 DSCB data area.

MINAU=*addr* or (*reg*)
Specifies the size of the minimum allocation unit in tracks. All primary extents for this data set are in multiples of this minimum allocation unit. This minimum does not apply to subsequent extensions of the data set.

The MINAU parameter is invalid on an ALLOC=ABS request.

***addr*–RX-type address**
Specifies the address of a word containing the minimum allocation unit.

(*reg*)–(0), (2-12)
Specifies a register containing the minimum allocation unit.

The MINAU parameter has no effect on the requested allocation if:

- You provide a value of zero.
- The PD1SCALO flag byte of the partial DSCB indicates either a cylinder request (X'C0') or an average block with round request (X'41').

Otherwise, the value of DSSIZE must be a multiple of the value of MINAU.

NUMF9=*number_dscbs* or (*reg*)
For REALLOC with F9DSCB requests, *number_dscbs* is an absolute expression or (register) with a value between 0 and 255 that designates the number of consecutive 140-byte partial format-9 DSCB areas that are provided in the F9DSCB=*addr*. The system treats a value of 0 as a 1 when F9DSCB=*addr* is specified. Format-9 DSCBs with a subtype field with a value other than 1 are ignored.

PDSCB=*addr* or (*reg*)
Specifies the address of a partial DSCB (for ALLOC=MOV) or the in-storage address of a full format-1 or format-8 DSCB (for ALLOC=ABS). This DSCB is used as a model to construct the allocated data set's format-1 or format-8 DSCB.

addr–RX-type address, (*reg*)–(0), (2-12)

Provide a value for the PDSCB parameter and initialize the PD1FMTID field to X'F1' for a format-1 DSCB or to X'F8' for a format-8 DSCB. However, the PDSCB attribute information determines EAS eligibility and whether the actual allocated DSCB allocated is a format-1 or format-8 DSCB.

PDSDIR=*addr* or (*reg*)
Specifies the number of 256-byte directory blocks for a partitioned data set (PDS).

***addr*–RX-type address**
Specifies an in-storage address of a full word containing the number of 256-byte PDS directory blocks.

(*reg*)–(0), (2-12)
Specifies a register containing the number of 256-byte PDS directory blocks.

Provide a value for PDSDIR when partitioned organization is indicated:

- The DS1DSORG flag byte of the format-1 DSCB is X'02' (ALLOC=ABS).
- The PD1DSORG flag byte of the partial DSCB is X'02' (ALLOC=MOV).

For an ALLOC=MOV request, you can specify the value of PDSDIR in the PDDIRQTY field of the partial DSCB. The PDDIRQTY field is used only if the REALLOC parameter list value of PDSDIR is zero.

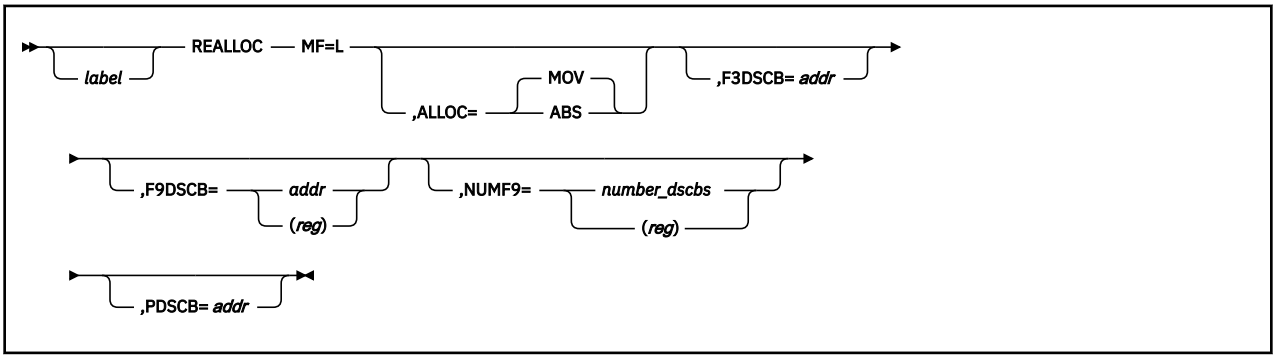
Do not specify a value for PDSDIR if a PDS is not indicated.

UCB=(*reg*)
Specifies the address of the UCB for the volume on which the data set is to be allocated. The UCB address can be for a captured UCB, or for an actual UCB above or below the 16MB line. For 31-bit callers, the high-order byte is part of the UCB address and must be cleared to zeros if a 24-bit UCB address is being passed. The volume must be mounted, and remain mounted. Use the address of a UCB, not a UCB copy.

(*reg*)–(0), (2-12)
Specifies a register containing the UCB address for the device.

REALLOC–List Form

The format of the list form of the REALLOC macro follows. See the execute form for an explanation of the parameters.



Restrictions:

1. The execute form parameters DSSIZE, MINAU, PDSDIR, and UCB cannot be specified on the list form.
2. The list form's ALLOC parameter affects the tests made by the REALLOC macro during assembly, and the contents of the parameter list.
3. The value of ALLOC is as specified or defaulted on the execute form.

REALLOC–DSECT Form

The format of the DSECT form of the REALLOC macro is:

REALLOC Parameter List

The parameter list for the DSECT form expansion is:

Table 13. REALLOC Parameter List

Offset	Length or Bit Pattern	Field Name	Description
0(X'00')		REALPL	DSECT name.
0(X'00')	4	RALPLID	EBCDIC "REAL" for "REALLOC"
4(X'04')	2	RALNGTH	Length of parameter list
6(X'06')	2	RAERRCDE	Error code from DADSM create
8(X'08')	1	RALFLAG	Flag byte
	1...	RALABS	0 means ALLOC=MOV. 1 means ALLOC=ABS.
	.1..	RALVTOCE	VTOCENQD=YES
9(X'09')	1	RALPFLGS	Processing flag byte.
	1...	RALDUMMY	Dummy REALLOC parameter list is passed. Only the processing flag byte (byte 9), minimum allocation unit (bytes 16-19), and UCB address (bytes 24-27) are used. Ignore all other bytes and use the values in the JFCB or Partial DSCB interface as passed in register 0.
	.1..	RALTRKAL	Space must be allocated from track-managed space.
	..1.	RALEXREQ	The exact amount of space must be allocated. Applicable to EAV. The request is to be allocated using a combination of the track-managed and/or the cylinder-managed spaces. If the exact space is not available, then the request is failed.
	...1 11..	*	Reserved.
	11	RALEATTR	The extended attribute (EATTR=) value to be used. Valid when RALDUMMY is set and when the JFCB is passed in register 0. • If 0, EATTR has not been specified. For VSAM data sets, the default behavior is equivalent to EATTR=OPT. For non-VSAM data sets the default behavior is equivalent to EATTR=NO. • If 1, EATTR=NO has been specified. The data set cannot have extended attributes (format 8 and 9 DSCBs) or optionally reside in EAS. • If 2, EATTR=OPT has been specified. The data set can have extended attributes and optionally reside in EAS. This is the default behavior for VSAM data sets. • If 3, Not Used, EATTR treated as not specified.
10(X'0A')	1	RALNUMF9	Number of contiguous partial format 9 DSCBs that are located at the address in bytes 32-35. The default is one.
11(X'0B')	1		Reserved
12(X'0C')	4	RALDSSZ	Data set size
16(X'10')	4	RALMAU	Minimum allocation unit
20(X'14')	4	RALPDSCB	Address of partial DSCB
24(X'18')	4	RALUCB	Address of UCB

Table 13. REALLOC Parameter List (continued)			
Offset	**Length or Bit Pattern**	**Field Name**	**Description**
28(X'1C')	4	RALDQTY	PDS directory quantity
32(X'20')	4	RAL2DSCB	Address of format 2 DSCB. Mutually exclusive with RAL9DSCB.
32(X'20')	4	RAL9DSCB	Contiguous partial format 9 DSCBs pointer. RALNUMF9 defines the number of partial format 9 DSCBs. Format 9 DSCBs with a subtype field with a value other than 1 are ignored. Only format 9 DSCB attribute data in this model will be used. Mutually exclusive with RAL2DSCB.
36(X'24')	4	RAL3DSCB	Address of format 3 DSCB
40(X'28')	0	RALEND	Byte after end of list
	40	RALENGTH	Symbolic length of parameter list

Return Codes from REALLOC

Control returns to the instruction following the instructions generated by the REALLOC macro. Register 15 contains the applicable REALLOC return code as shown in Table 14 on page 54.

REALLOC returns 4 bytes of diagnostic information in register 0. See *z/OS DFSMSdfp Diagnosis*.

If an error occurs, REALLOC issues message IEC614I, consisting of failure-related status information. See Table 14 on page 54 for descriptions of the conditions indicated by the REALLOC return code.

Exception: This is a cumulative list of DADSM create return codes. Some of these codes might not apply to the REALLOC macro.

Table 14. DADSM CREATE Return Codes

Return Code	Description
00(X'00')	If the 4 bytes of diagnostic information returned in register 0 are all zeros, this indicates successful data set creation. If they are nonzero, see the DADSM/CVAF Diagnostic Aids section of *z/OS DFSMSdfp Diagnosis* to determine the failure-related conditions.
04(X'04')	Duplicate data set name.
08(X'08')	No room in VTOC or VTOC index.
12(X'0C')	Permanent I/O error or CVAF error.
16(X'10')	Requested absolute track not available.
20(X'14')	Requested quantity not available.
24(X'18')	Average record length exceeds 65,535 bytes.
28(X'1C')	ISAM is no longer supported and the request has been failed.
32(X'20')	SMS configuration parmlib field USESLV was set to NO. The request to allocate on a volume with more than 65,520 cylinders is not allowed.
40(X'28')	The create request specified DACEXREQ, but the exact amount of space could not be returned. Because the exact amount was required, no space is returned.
48(X'30')	Invalid DADSM REALLOC parameter list.
52(X'34')	Invalid JFCB, partial DSCB pointer, or ineligible DSORG specified with F8 ID in the DSCB for an EAS request on EAV.
56(X'38')	Not enough space on volume for directory.
60(X'3C')	REALLOC ALLOC=ABS is not supported on unindexed VTOCs.
61(X'3D')	REALLOC ALLOC=NON requested the free space chain to be rebuilt. The data set was not created.
64(X'40')	Invalid user label request.

Table 14. DADSM CREATE Return Codes (continued)

Return Code	Description
68(X'44')	Invalid UCB pointer. Requires the address of a UCB, not a UCB copy.
72(X'48')	VSE VTOC cannot be converted to an unindexed VTOC.
76(X'4C')	No space parameter given for a new data set or zero space requested at absolute track zero.
104(X'68')	Invalid space subparameter.
108(X'6C')	ABSTR request.
116(X'74')	User labels not supported.
120(X'78')	Invalid combination of DSSIZE and MINAU in REALLOC parameter.
124(X'7C')	DSSIZE not a multiple of MINAU.
128(X'80')	Directory space requested is larger than primary space.
136(X'88')	Invalid FMT3 DSCB pointer.
144(X'90')	EAV extent above 65520 cylinders is not an even integral of the multicylinder unit.
148(X'94')	Overlapping extents in the VTOC.
156(X'9C')	DADSM CREATE terminated because of possible VTOC errors.
164(X'A4')	Allocation terminated because of VSE stacked pack format.
168(X'A8')	RACDEF failed, data set already defined.
172(X'AC')	User not authorized to define data set.
176(X'B0')	Installation exit rejected this request with return code 8.
180(X'B4')	Installation exit rejected the request with return code 4.
184(X'B8')	RACF define with modeling specified and model not found.
188(X'BC')	Invalid FMT2 DSCB pointer.
192(X'C0')	Requested data set creation was not allowed by SMS.
196(X'C4')	Requested data set creation was not possible. Register 0 contains additional diagnostic information.
200(X'C8')	The PDSE directory could not be built.
204(X'CC')	VTOC ENQ related failure.
208(X'D0')	I/O error occurred during the allocation of a data set. The data set will be deleted.
212(X'D4')	Request failed due to a presence of split cylinder data sets on the volumes.
216(X'D8')	For this type of data set, the primary quantity requested cannot exceed 65,535 tracks.
217(X'D9')	Data set could not be created DSNTYPE=LARGE not valid for this data set.
220(X'DC')	VTOC conversion failed because the VTOC was full.

Accessing the VTOC with CVAF Macros

The common VTOC access facility (CVAF) macros and tasks discussed in this section consist of the following:

- CVAFDIR directly accesses one or more DSCBs.
- CVAFDSM obtains volume free space information.
- CVAFFILT reads sets of DSCBs for one or more DASD data sets.
- CVAFSEQ retrieves the following:
  - Data set names from an active VTOC index

  - DSCBs in physical-sequential order
  - DSCBs in data set name order (index required).
- CVAFTST determines if a DASD volume has an active VTOC index.

"Coding CVAF VTOC Access Macros" on page 71, contains descriptions of these macros and examples of their use.

When calling CVAF, your program can be in either 24-bit or 31-bit addressing mode. If it is in 31-bit mode, the control blocks shown in Figure 6 on page 66 might reside above the 16 MB line. All these areas must be accessible in your program's storage key.

Note: You must supply a UCB address that matches the caller's AMODE. That is, AMODE=24 requires a 24 bit UCB address, while AMODE=31 requires a 31 bit UCB address.

Serializing and Updating

CVAF requires that you provide all necessary system resource serialization for your request. You can ensure the integrity of multiple data elements (sets of DSCBs or VIRs) returned by CVAF only if you adequately serialize system resources and avoid multiple CVAFFILT requests for a set of DSCBs or VIRs. Weigh possible system performance loss because of serialization against the potential loss of data integrity.

Updating without adequate serialization might compromise the integrity of the volume's VTOC, the VTOC index, or any associated data set.

CVAF only complies with requests to modify the volume's VTOC or index from authorized programs.

CVAF assumes that an authorized program holds an exclusive RESERVE (or ENQ) on the *qname* (major name) of SYSVTOC, and the rname (minor name) of the volume's serial number, with the scope of SYSTEMS. This RESERVE can be made more efficient if Global Resource Serialization or a functional equivalent is active.

The SYSVTOC *qname* does not serialize access to the format-1 or the format-8 DSCB for a data set. You can provide serialization by allocating the data set with disposition OLD, MOD, or NEW (not SHR). This causes the proper ENQ, ensuring that no other job can update that data set's format-1 or format-8 DSCB.

If your program holds the enqueue for the SYSVTOC resource, then no other program can later start and complete a DADSM request. This includes extending or creating a data set on that volume in your own address space. If you try to extend a data set under the same task that holds SYSVTOC, it will be abnormally terminated. If a different task requests SYSVTOC, that task will wait. If your program then requests a resource that is held by that task, the two tasks will be in a deadlock. To prevent deadlocks between tasks in the same address space when either of the tasks has previously enqueued on SYSVTOC, the secondary caller of CVAF must ensure the enqueue bit is on (CV3ENQD) in the CVPL. Other options to get around this limitation are either:

- Allocating a data set so that it does not require secondary extents.
- Requesting that the output data set be on a volume other than the one where the application holds enqueue for the SYSVTOC resource.

Identifying the Volume

If authorized, you can identify the volume to the CVAFDIR, CVAFDSM, CVAFFILT, and CVAFSEQ macros by specifying the address of the UCB. These macros do not accept the address of a UCB copy. If your program is not authorized, specify the address of a SAM or EXCP DEB opened to the volume's VTOC.

The data extent block (DEB) can be obtained by opening a DCB for INPUT, using the RDJFCB and OPEN TYPE=J macros. (After issuing an OPEN TYPE=J macro, the DCB's DCBDEBA field contains the DEB's address.) The DCB's DDNAME must identify a DD statement allocated to the unit whose VTOC is to be accessed. Once your program issues the RDJFCB macro, it must initialize the JFCBDSNM field with the data set name of the format-4 DSCB: 44 bytes of X'04'. The RDJFCB macro is described under "RDJFCB Macro Specification" on page 275; the OPEN macro is described under "OPEN - Initialize Data Control Block for Processing the JFCB" on page 287. For an extended address volume the DCB macro must point

to a DCBE where the EADSCB=OK keyword is specified. If you do not code this option, the OPEN function will issue ABEND 113-48 and message IEC142I.

If a CVAF macro call specifies IOAREA=KEEP, a subsequent CVAF call using a different CVAF parameter list (CVPL) might omit the UCB and DEB keywords and supply the IOAREA address from the other CVPL by using the IOAREA keyword.

Generating a CVPL (CVAF Parameter List)

The CVAFDIR, CVAFDSM, CVAFFILT, and CVAFSEQ macros use the CVPL to pass parameters to CVAF. The CVAFTST macro expands to provide its only parameter (UCB address) in register 1, and calls the applicable CVAF module. CVAF returns information related to the CVAF request in the CVPL. The CVAFTST macro will accept the address of a UCB or UCB copy. Unauthorized programs can get a copy of the UCB by using the UCBSCAN macro and specifying the COPY, UCBAREA, CMXTAREA, and DCEAREA keywords. The UCB copy, including the extension, must be below the 16MB line and on a word boundary. Data accessed with the DCEAREA can be above the 16MB line. Refer to *z/OS HCD Planning* for details.

Specifying the CVAFDIR, CVAFDSM, CVAFFILT, or CVAFSEQ macro with MF=L or MF=I (the default) as a subparameter generates a CVPL. Upon return the CV1IVT bit in the CVPL indicates whether the accessed VTOC was indexed or nonindexed. If an error occurs, the CVSTAT field contains feedback data. The CVAF I/O area address is returned in the CVIOAR field and the CVAF filter save area address is returned in the CVFSA field. To use the generated CVPL to execute a different function than was originally specified, use the MF=E keyword.

To specify a CVAF filter request, use a CVPL generated by the CVAFFILT macro. The CVAFFILT macro generates a CVPL 4 bytes longer (total length = X'44') than that generated by the other CVAF macros (total length = X'40'). CVAFFILT request need a longer parm list to accommodate one extra file.

- To get the longer parameter list (X'44' bytes), specify CVPLFSA=YES on the inclusion of the ICVAFPL mapping macro.
- To get the shorter parameter list (X'40' bytes), specify the CVPLX=YES on the inclusion of the ICVAFPL mapping macro.

The ICVAFPL macro maps the CVPL. The format of the CVPL is shown in Table 15 on page 57.

Note: The area starting at CVCTAR5 is generated only when the CVPLX=YES macro variable is specified on the invocation of ICVAFPL.

Table 15. CVAF Parameter List - ICVAFPL

Offset	Type	Length	Name	Description
0(X'00')	STRUCTURE	100	CVPL	CVAF parameter list
0(X'00')	CHARACTER	4	CVLBL	EBCDIC 'CVPL'
4(X'04')	SIGNED	2	CVLTH	Length of CVPL
6(X'06')	BIT(8)	1	CVFCTN	Function byte (see values below)
7(X'07')	UNSIGNED	1	CVSTAT	Status information (see values below)
8(X'08')	BIT(8)	1	CVFL1	First flag byte
	1...		CV1IVT	Indexed VTOC accessed
	.1..		CV1IOAR	IOAREA=KEEP
	..1.		CV1PGM	BRANCH=(YES,PGM)
	...1		CV1MRCDS	MAPRCDS=YES
	.... 1...		CV1IRCDS	IXRCDS=KEEP
	.... .111		CV1MAP	Map bits (see next three bits)

Table 15. CVAF Parameter List - ICVAFPL (continued)

Offset	Type	Length	Name	Description
	.... .1..		CV1MAPIX	MAP=INDEX
	.... ..1.		CV1MAPVT	MAP=VTOC
	1		CV1MAPVL	MAP=VOLUME
9(X'09')	BIT(8)	1	CVFL2	Second flag byte
	1...		CV2HIVIE	HIVIER=YES
	.1..		CV2VRF	VRF data exists
	..1.		CV2CNT	COUNT=YES
	...1		CV2RCVR	RECOVER=YES
	.... 1...		CV2SRCH	SEARCH=YES
	.... .1..		CV2DSNLY	DSNONLY=YES
	.... ..1.		CV2VER	VERIFY=YES
	1		CV2NLEVL	Output - new highest level VIER created
10(X'0A')	BIT(8)	1	CVFL3	Third flag byte
	1...		CV3FILT	FLTAREA=KEEP
	.1..		CV3IXERR	Index error found
	..1.		CV3PTR31	Address words in CVPL are valid in AMODE 31
	...1		CV3AVT	Support bypass write inhibit error.
	.... 1...		CV3ENQD	CVAF caller has serialized with another task that has reserved the VTOC
	.... .1..		CV3NOPIN	Caller requested no UCBPIN
	.... ..1.		CV3HLIXP	Higher level IX pruning done
	1		CV3RTA4B	CVAFDSM RTA4BYTE=YES
11(X'0B')	BIT(8)	1	CVFL4	Fourth flag byte
	1...		CV4UCBCA	UCB has been captured
	.1..		CV4OBTRQ	Request came from OBTAIN
	..1.		CV4NMHID	Data set name hiding for authorized caller.
	...1		CV4EADOK	EADSCB=OK specified
	.... 1...		CV4MULTD	Processing of multiple DSCBs on CVAFDIR is requested (MULTIPLEDSCBS=YES)
	.... .1..		CV4NOQUE	Do not queue the OBTAIN I/O
	.... ..1.		CV4VALID	Validate fields when CVAFDIR ACCESS=WRITE
	x		CV4RSVD	Reserved
12(X'0C')	ADDRESS	4	CVUCB	UCB address
16(X'10')	ADDRESS	4	CVDSN	Data set name address
16(X'10')	ADDRESS	4	CVFCL	Filter criteria list address

Table 15. CVAF Parameter List - ICVAFPL (continued)

Offset	Type	Length	Name	Description
20(X'14')	ADDRESS	4	CVBUFL	Buffer list address
24(X'18')	ADDRESS	4	CVIRCDS	Index VIR's buffer list pointer
28(X'1C')	ADDRESS	4	CVMRCDS	MAP VIR's buffer list pointer
32(X'20')	ADDRESS	4	CVIOAR	I/O area address
36(X'24')	ADDRESS	4	CVDEB	DEB address
40(X'28')	ADDRESS	4	CVARG	Argument address
44(X'2C')	ADDRESS	4	CVSPACE	Space parameter list address
48(X'30')	ADDRESS	4	CVEXTS	Extent table address
52(X'34')	ADDRESS	4	CVBUFL2	New VRF VIXM Buffer list pointer
56(X'38')	ADDRESS	4	CVVRFDA	VRF data address
60(X'3C')	ADDRESS	4	CVCTAR	Count area address
64(X'40')	ADDRESS	4	CVFSA	Filter save area address
68(X'44')	ADDRESS	4	CVCTAR5	CCHHR AREA RETURN ADDRESS
72(X'48')	UNSIGNED	1	CVNMDSCB	NUM OF DSCBs
73(X'49')	CHARACTER	4	CVCLID	ID of CVAFDIR WRITE caller
77(x'4D')	CHARACTER	23	*	Reserved

The possible contents of the CVFCTN field in the CVPL and their meanings are as follows:

Table 16. CVFCTN Field of CVPL—Contents and Definitions

Value	Name	Description	
X'01'	CVDIRD	CVAFDIR	ACCESS=READ
X'02'	CVDIWR	CVAFDIR	ACCESS=WRITE
X'03'	CVDIRLS	CVAFDIR	ACCESS=RLSE
X'04'	CVSEQGT	CVAFSEQ	ACCESS=GT
X'05'	CVSEQGTE	CVAFSEQ	ACCESS=GTEQ
X'06'	CVDMIXA	CVAFDSM	ACCESS=IXADD
X'07'	CVDMIXD	CVAFDSM	ACCESS=IXDLT
X'08'	CVDMALC	CVAFDSM	ACCESS=ALLOC
X'09'	CVDMRLS	CVAFDSM	ACCESS=RLSE
X'0A'	CVDMMAP	CVAFDSM	ACCESS=MAPDATA
X'0E'	CVFIRD	CVAFFILT	ACCESS=READ
X'0F'	CVFIRES	CVAFFILT	ACCESS=RESUME
X'10'	CVFIRLS	CVAFFILT	ACCESS=RLSE
X'AA'	CVDMMAPX	CVAFDSM	ACCESS=MAPDATA RTA4BYTE=YES

Using Buffer Lists

A buffer list consists of one or more chained control blocks, each with a header and buffer list entries, obtained and initialized by your program before calling CVAF. The header indicates whether the buffer list is for DSCBs. The entries point to and describe the buffers. You can create buffer lists in two ways:

- Directly, when you fill in the arguments and buffer addresses of DSCBs to be read or written
- Indirectly (by CVAF), when you code the IXRCDS=KEEP and/or MAPRCDS=YES keywords.

The ICVAFBFL macro maps CVAF buffer lists. Table 17 on page 60 shows the format of a buffer list header. Table 18 on page 61 shows the format of a buffer list entry.

Buffer List Header

The buffer list header indicates whether the buffer list describes buffers for DSCBs. The DSCB bit must be set to 1 and the VIR bit to zero for CVAF to process a request to read or write a DSCB. Provide buffer lists and buffers in your program's protect key. CVAF uses the protect key and subpool fields in the buffer list header only if you code ACCESS=RLSE.

Each buffer list header contains a count of the number of entries in the buffer list that directly follows the header.

The forward chain address chains buffer lists together. The format of the buffer list header is shown in Table 17 on page 60.

Table 17. Format of a Buffer List Header

Offset	Length or Bit Pattern	Name	Description
0 (X'00')	8	BFLHDR	Buffer list header.
0 (X'00')	1	BFLHNOE	Number of entries.
1 (X'01')	1	BFLHFL	Key and flag byte.
	xxxx	BFLHKEY	Protect key of buffer list and buffers.
	. . . . 1 . . .	BFLHVIR	Reserved.
	. . . . . 1 . .	BFLHDSCB	Buffer list entries describe DSCBs.
	. . . . . 1 .	BFLHWREV	Write multiple DSCBs in reverse order.
	x		Reserved.
2 (X'02')	1	BFLHNOEN	Number of entries needed to read all DSCBs describing the DSN. Set by CVAFDIR READ when processing a READ of a format-1 or format-8 DSCB. Applicable to all volume types and set regardless of the value provided in BFLHNOE.
3 (X'03')	1	BFLHSP	Identifies the subpool of buffer list and buffers.
4 (X'04')	4	BFLHFCHN	Forward chain address of next buffer list.

Buffer List Entry

A buffer list contains one or more entries in contiguous storage. Each entry provides the buffer address, the length of the DSCB buffer, the argument, and indicates whether the argument is an RBA, a TTR, or a CCHHR. The fields and bit uses are listed below.

- For a DSCB buffer, the RBA bit must be 0, and either the TTR or CCHHR bits must be set to 1 (they must not both be 1).
- The BFLESKIP bit causes an entry to be ignored.
- The BFLEIOER bit is an output indicator from CVAF that indicates an I/O error occurred during reading or writing of the DSCB.

- The BFLELTH field is the length of the buffer; for a DSCB buffer, the length must be 96 or 140.
- The BFLEARG field is the argument (address) of the DSCB. Specify the format of the 5-byte field by setting the BFLECHR, or BFLETTR bit to 1. The respective BFLEARG values and formats are as follows:

Value	Format
CCHHR	5-byte CCHHR
TTR	0TTR0

The values for BFLEARG depend on the variables associated with a given request. These are described in the following request-oriented topics.

The format of the buffer list entry is shown in Table 18 on page 61.

Table 18. Format of a Buffer List Entry

Offset	Length or Bit Pattern	Name	Description
0 (X'00')	12	BFLE	Buffer list entry.
0 (X'00')	1	BFLEFL	Flag byte.
	100	BFLERBA	Argument is RBA.
	010	BFLECHR	Argument is CCHHR.
	001	BFLETTR	Argument is TTR.
	. . . 1	BFLEAUPD	CVAF updated argument field.
	. . . . 1 . . .	BFLEMOD	Data in buffer has been modified.
	. . . . . 1 . .	BFLESKIP	Skip this entry.
	. . . . . 1 .	BFLEIOER	I/O error.
	 1	BFLENOVR	If using CVAFDIR to write a 96-byte DSCB, bypass comparing the DSCB key to the data set name.
1 (X'01')	1	BFLEFLG2	Flag byte 2.
	1	BFLENVER	When using CVAFDIR to write multiple DSCBs and this flag is set, CVAF will not verify that the existing DSCB is format 0 prior to writing this DSCB. This overrides VERIFY=YES on CVAFDIR (CV2VER) for this buffer list entry.
	. 1	BFLEVLER	This buffer cannot be written due to an error in validation. (VALIDATE=YES) on CVAFDIR write.
			Reserved.
2 (X'02')	1	BFLELTH	Length of DSCB buffer.
3 (X'03')	5	BFLEARG	Argument of DSCB.
4 (X'04')	3	BFLEATTR	TTR of DSCB.
4 (X'04')	4	BFLEARBA	Reserved.
8 (X'08')	4	BFLEBUF	Buffer address.

Using Macro ICVEDT02 to Map the Extents Area

The ICVEDT02 mapping macro is used to map the extent area when you use the CVAFDSM macro and specify RTA4BYTE=YES for indexed VTOCs or nonindexed VTOCs.

The format of the ICVEDT02 mapping macro follows.

Table 19. Format of ICVEDT02 Mapping Macro

Offset	Bytes	Name	Description
0 (X'00')	8	DT2X7EYE	Identifier="ICVEDT02"
8 (X'08')	4	DT2X7LEN	Length of ICVEDT02.
12 (X'0C')	1	DT2X7LEV	Level number.
13 (X'0D')	1	DT2X7FLG	Flag byte.
14 (X'0E')	2	DT2X7NF0	Number of format-0 DSCBs created.
16 (X'10')	5	DT2X7CSR	CCHHR cursor field. Used by the system. Initialized to zero by caller —then do NOT modify.
21 (X'15')	3	DT2X7RES	Reserved.
24 (X'18')	4	DT2X7RTA	Relative Track Address (RTA) cursor field. Used by the system. Initialized to zero by caller—then do NOT modify.
28 (X'1C')	4	DT2X7RE2	Reserved.
32 (X'20')	4	DT2X7ENT	Number of extent descriptor entries.
-	-	-	Extent descriptor array.
36 (X'24')	8	DT2ENTRY	One entry for each extent.
-	4	DT2RTAST	RTA of start of extent
-	4	DT2RTAED	RTA+1 of end of extent

Accessing the DSCB Directly

The CVAFDIR macro can be used to read or write one or more DSCBs and is described in “CVAFDIR Macro Overview and Specification” on page 71. After a CVAFDIR call, you can test the CV1IVT bit in the CVPL to determine whether the VTOC is indexed or nonindexed.

If the first buffer is 96 bytes, CVAF issues a channel program to verify that the key in the DSCB matches the 44-byte data set name you entered, unless the operation is a write and the BFLENOVR bit is on.

If the first buffer is for a 96-byte write and the BFLENOVR bit in the BFLEFL is set to 1, CVAF skips the key verification, improving performance. If you are not certain that the data set name you provide is correct, set the BFLENOVR bit to 0. If the BFLENOVR bit is set to 0, CVAF does not execute the write unless the keys match.

If CVAF is performing key verification, and the DSCB key does not match the data set name you supply, CVAF ignores any specified BFLEARG and writes the first DSCB using the rules described in the following section, “Specifying a Data Set Name to Read or Write a DSCB” on page 62.

Specifying a Data Set Name to Read or Write a DSCB

To read or write one or more DSCBs by specifying only the data set name (that is, BFLEARG is zero), specify either ACCESS=READ or ACCESS=WRITE.

Specify the address of the data set name in the DSN keyword. Specify the address of the buffer list in the BUFLIST keyword. Each of these areas and the associated buffers must be in your program's protect key.

The buffer list must contain at least one buffer list entry with the skip bit off and a pointer to a 96-byte first buffer. Do not use a 140-byte buffer as the first buffer. You can chain buffer list headers together, but at this time, CVAF only uses the first buffer list.

For an indexed VTOC, CVAF searches the index for the data set name. If the data set name is found, the DSCB argument is put into the buffer list entry and used to read or write the DSCB. If the data set name cannot be found in the index, CVAF performs a key search of the VTOC.

For a nonindexed VTOC, CVAF uses a channel program to perform a key search of the VTOC to locate the data set name and read or write the DSCBs. If the data set name is found, CVAF puts the DSCB argument into the buffer list entry.

The DSCB argument returned in the buffer list entry is in the format determined by the BFLECHR or BFLETTR bits in the buffer list entry.

If CVAF does not find the data set name in the VTOC, a return code of 4 is indicated in register 15, and an error code of 1 in the CVSTAT field.

For CVAF calls that pass a data set name, a CVAFDIR request will fail if the EADSCB=OK indicator is not set and the DSCB associated with this data set name is a format-8 DSCB. CVAF return code of 4 with a CVAF status code (CVSTAT) of STAT082 will be set.

If you use CVAFDIR MULTIPLEDSCBS=YES ACCESS=READ, all DSCBs associated with this data set name are read and returned in the buffer list buffer (BFLEBUF) for entries without the skip bit on in logical VTOC order. The associated address of each DSCB is put in the corresponding BFLEARG field. The total number of DSCBs associated with this data set name is also returned in the BFLHNOEN field of the buffer list header. If there are not enough buffers to house all the DSCBs associated with this data set name, only those DSCBs that have BFLEs are processed. The buffer list header field BFLHNOEN will then have a number greater than the BFLHNOE field.

If you use CVAFDIR MULTIPLEDSCBS=YES ACCESS=WRITE, all BFLEs without the skip bit are used to write all DSCBs with one call. BFLARG must be correct for all entries (except the first 96-byte DSCB) and BFLEBUF must point to the correct corresponding DSCB to be written.

Specifying the DSCB Location

To read or write one or more DSCBs by specifying the DSCB's location (that is, BFLEARG), specify either ACCESS=READ or ACCESS=WRITE.

Specify the address of the data set name in the DSN keyword and the address of the buffer list in the BUFLIST keyword. Each of these areas and the associated buffers must be in your program's protect key.

The buffer list must have at least one buffer list entry with the skip bit off and a pointer to a 96-byte or 140-byte buffer. You can chain buffer lists together, but at this time CVAF uses only the first buffer list.

If the first buffer is for a 140-byte read or write, CVAF issues a channel program to read or write the DSCB at the location specified in the buffer list entry. CVAF ignores the specified data set name. If you specify VERIFY=YES, CVAF verifies that the designated DSCB is a format-0 DSCB before issuing the write channel program.

For CVAF calls that specify the DSCB location, a CVAFDIR request will fail if the EADSCB=OK indicator is not set and the DSCB associated with this location is a format-8 DSCB. CVAF return code of 4 with a CVAF status code (CVSTAT) of STAT082 will be set.

If you use CVAFDIR MULTIPLEDSCBS=YES ACCESS=READ, all DSCBs associated with the data set whose format-1 or format-8 DSCB is pointed to by the BFLEARG of the first BFLE entry are read and returned in the buffer list buffer (BFLEBUF) for entries without the skip bit on in logical VTOC order. The associated address of each DSCB is put in the corresponding BFLEARG field. The total number of DSCBs associated with this data set is also returned in the BFLHNOEN field of the buffer list header. If there are not enough buffers to house all the DSCBs associated with this data set name, only those DSCBs that have BFLEs are processed. The buffer list header field BFLHNOEN will then have a number greater than the BFLHNOE field.

If you use CVAFDIR MULTIPLEDSCBS=YES ACCESS=WRITE, all BFLEs without the skip bit are used to write all DSCBs with one call. BFLARG must be correct for all entries and BFLEBUF must point to the correct corresponding DSCB to be written.

Releasing Buffers and Buffer Lists Obtained by CVAF

You can release buffers and buffer lists acquired by CVAF in the following ways:

- Issue a CVAF call with ACCESS=RLSE, and specify a buffer list address with the BUFLIST keyword.

- Free a MAP records buffer list by coding MAPRCDS=NO or MAPRCDS=(NO,*addr*) and specifying any ACCESS.
- Free an index records buffer list by coding IXRCDS=NOKEEP or IXRCDS=(NOKEEP,*addr*) and specifying any ACCESS.

CVAF frees all eligible buffers and any buffer lists that become empty. Eligible buffers are those pointed to by buffer list entries with the skip bit off. CVAF frees a buffer list if none of its buffer list entries have the skip bit on. If buffer lists are chained together, CVAF checks and frees all appropriate buffer lists. Do not request CVAF to release the same buffer list twice by specifying its address in more than one place.

Accessing DSNs or DSCBs in Sequential Order

You can use the CVAFSEQ macro to request the return of information about the data sets:

- DSCBs in indexed (data set name) order (either format-1 and format-8 DSCBs or a format-4 DSCB)
- One or more DSCBs in physical-sequential order. (If you are unauthorized, you can request only one DSCB.)
- The next data set name in the index.

CVAF reads the DSCBs into buffers identified by the BUFLIST keyword. See "CVAFSEQ Macro Overview and Specification" on page 111 for additional information about the macro.

Use the buffer list to specify the argument of each DSCB to be read. For indexed access, request 96-byte DSCBs in the buffer list. For physical-sequential access, request 140-byte DSCBs.

If you select indexed order, CVAF returns each format-1, format-4, or format-8 DSCB pointed to by the index. To return only the data set names in the index (not the DSCBs), specify DSNONLY=YES. CVAF updates the DSN area specified in the CVAFSEQ macro with the data set name of each DSCB read, every time you issue CVAFSEQ. CVAF also returns the CCHHR of the DSCB in the argument area supplied with the ARG keyword.

Note: The returned DSCB from CVAFSEQ always includes the key portion of the DSCB when the supplied length is 140 bytes for the buffer (field BFLELTH). The mapping for structure IECSDSF4 in macro IECSDSL1, however, does **not** include the key portion of the DSCB. Therefore, if you use the format-4 DSCB fields defined in this structure (all fields starting with DS4), you must adjust the starting point of the first field in the IECSDSF4 structure to correspond to 44 bytes into the returned buffer.

For indexed access, a CVAFSEQ request will fail if the EADSCB=OK indicator is not set and the DSCB attempted to be read is a format-8 DSCB. CVAF return code of 4 with a CVAF status code (CVSTAT) of STAT082 will be set.

Initiating Indexed Access (DSN Order)

To initiate indexed access (DSN order), either supply 44 bytes of binary zeros in the DSN area (to indicate the first data set name in the index) or specify the data set name that is the starting point for the index search.

The name returned in the DSN area is equal to or greater than the DSN supplied, depending on the ACCESS keyword selection.

If you specify DSNONLY=NO, CVAF returns the DSCB and argument using the buffer list provided with the BUFLIST keyword. CVAF uses the first entry in the buffer list with the skip bit set to 0 and a nonzero buffer address. You can specify the argument format by setting either the TTR or CCHHR bit in the buffer list entry to 1. If neither bit is set, CVAF returns a CCHHR argument. For indexed access, the DSCB size in the buffer list entry must be 96 bytes.

If you specify DSNONLY=YES, specify the CCHHR argument in the ARG area.

The data set name of the format-4 DSCB is in the index and CVAF might return its name (44 bytes of X'04'). The format-4 DSCB's name is likely to be the first data set name in the VTOC index.

Initiating Physical-Sequential Access

To initiate physical-sequential access, either specify DSN=0, or do not specify the DSN parameter at all. To begin the read, initialize the argument field in the first buffer list entry to zero or to the argument of the DSCB. If the argument is zero, CVAF uses the argument of the start of the VTOC.

The ACCESS keyword determines whether CVAF reads the DSCB whose argument is supplied or the DSCB following it. For example, to read the first DSCB (the format-4 DSCB) in the VTOC, you can set the BFLEARG in the first buffer list entry to zero and specify ACCESS=GTEQ in the CVAFSEQ macro. If you subsequently specify ACCESS=GT, CVAF reads the second DSCB (the first format-5 DSCB). Set the DSCB size to 140 in buffer list entries.

If your program is authorized, CVAF reads as many DSCBs as there are entries in the buffer list for a single CVAF call; if it is not authorized, CVAF reads only one DSCB.

CVAF uses one buffer list and does not inspect a second buffer list chained from the first. If your program is authorized, CVAF uses all entries in the buffer list; if it is not authorized, CVAF uses only the first entry. CVAF does not inspect the skip bit. Each entry must contain a buffer address and have the length field set to 140. CVAF updates the argument field of each buffer list entry with the argument of the DSCB. You can specify the argument format by setting either the TTR or CCHHR bit in the buffer list entry to 1. If neither bit is set, CVAF returns a CCHHR argument.

CVAF uses only the argument in the first entry to begin the search and does not inspect arguments in subsequent entries. If you specify a nonzero argument value in the first entry, there must be a DSCB with that argument.

CVAF indicates an end-of-data condition by providing return code 4 in register 15, and a value of X'20' in the CVSTAT field. CVAF sets the argument fields of all buffer list entries following the last DSCB read, to zero (the first entry is zero if CVAF does not read any DSCBs).

CVAF reads all DSCBs, including format-0 DSCBs. You cannot be certain that you have read all DSCBs until CVAF has read the entire VTOC. For a nonindexed VTOC, the DS4HPCHR field of the format-4 DSCB contains the CCHHR of the last format-1 DSCB. DSCBs other than format-1 can reside beyond that location. For an indexed VTOC, the VMDS contains information about which DSCBs are format-0 DSCBs.

For physical-sequential access, a CVAFSEQ request to an extended address volume will fail if the EADSCB=OK indicator is not set. CVAF return code of 4 with a CVAF status code (CVSTAT) of STAT082 will be set.

Reading Sets of DSCBs with CVAF Filter

You can use the CVAFFILT macro to retrieve sets of DSCBs into buffers that are provided by the calling program. CVAF filter service supports both indexed and nonindexed VTOCs. “CVAFFILT Macro Overview and Specification” on page 94 describes the macro's format and parameters. The following section summarizes this service and its requirements.

- Request DSCBs by specifying either one or more fully qualified data set names, or one partially qualified name. See “Filter Criteria List (FCL)” on page 67 and “Partially-Qualified Names for CVAFFILT” on page 99 for further information.
- Identify a single DASD volume in the CVPL.
- For each data set that has a format-1 DSCB, the VTOC order that is returned by CVAFFILT is the format-1 DSCB, followed by any format 3 DSCBs.

  For each data set that has a format-8 DSCB, the VTOC order that is returned by CVAFFILT is the format-8 DSCB, one or more format-9 DSCBs, followed by any format-3 DSCBs. Currently, the minimum buffer list entry size that is needed to read all DSCBs associated with a data set is 12. This is one format-8 DSCB, one format-9 DSCB, and 10 format-3 DSCBs to reach the 123-extent limit.
- CVAF filter service returns DSCB information for one or more qualifying data sets into caller-provided buffers. See “Example of CVAFFILT Macro Sequences” on page 69 and “Example of Using the CVAFFILT Macro” on page 99 for further information. CVAF filter service does not return a partial DSCB chain in the following cases:

- If you do not provide enough buffers to hold the requested DSCBs, CVAF filter service returns one or more complete DSCB chains and/or a status code (CVSTAT in the CVPL). The status code indicates if a RESUME CVAF call can be used to retrieve the rest (or more) of the DSCBs. See "RESUME Capability" on page 66 for specific information.
- If the total number of buffers is insufficient to contain a data set's complete DSCB chain, CVAF filter service sets the FCLDSNST byte in the FCL, ignores the data set, and processes the next qualifying data set. To avoid this situation, provide a minimum of eleven DSCB buffers (enough for a data set at the 123 extent limit).

- For fully qualified data set names in the Filter Criteria List, a CVAFFILT request that is issued to an extended address volume will fail if the EADSCB=OK indicator is not set and the DSCB associated with the fully qualified data set name is described by a format-8 DSCB. For partially qualified data set names in the Filter Criteria List, a CVAFFILT request issued to an extended address volume will fail if the EADSCB=OK indicator is not set and a DSCB associated with a data set that matches the Filter Criteria List is described by a format- 8 DSCB. In both these cases, the data set name status in the FCL (FCLDSNST) will be set to a status value of (x'06'). This status code indicates that a data set name is described by a format-8 DSCB and the caller did not specify support for it with the EADSCB=OK keyword. It will be set only when other data set name status codes are not applicable. CVAF status code (CVSTAT) of STAT086 (x'56') will be set with these errors. This status code indicates all of the user FCL entries were processed, a resume is not required, one or more errors were found.

RESUME Capability

If CVAF filter service terminates because you failed to provide sufficient buffers, the information necessary for a RESUME function is saved in the filter save area. (Specifying FLTAREA=KEEP on the initial CVAFFILT allocates the filter save area.)

To allow RESUME processing to execute correctly, you must maintain the relationship between the requested volume (identified by CVDEB, CVUCB, or a kept IOAREA), your FCL, and CVAF's FSA. If you observe this requirement, you can initiate and resume multiple CVAF filter service operations asynchronously on one or more DASD volumes. You can ensure this relationship by providing a unique CVPL and FCL for the duration of the READ/RESUME/RELEASE sequence associated with each logical request.

Issuing an ACCESS=RESUME without having previously specified FLTAREA=KEEP causes CVAF filter service to produce return code 4 in register 15 and 66 (decimal) in the CVSTAT field.

If you specify FLTAREA=KEEP, issue a subsequent CVAFFILT call with the ACCESS=RLSE keyword to release filter save area storage.

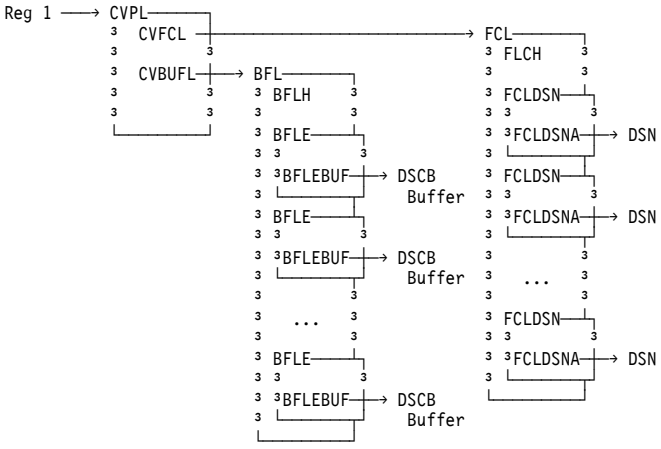


Figure 6. Control blocks required for CVAF filter services

Filter Criteria List (FCL)

The filter criteria list (FCL) consists of a list header and a variable number of list entries. The list entries follow the header, and each entry represents a data set name to be processed by CVAF filter. The header and entries, shown in Table 20 on page 67 and Table 21 on page 68, are mapped by the ICVFCL macro. The format of the FCL header is shown in Table 20 on page 67.

Table 20. Format of a Filter Criteria List Header

Offset	Bytes	Name	Description
0 (X'00')	4	FCLID	EBCDIC 'FCL*bb*' (*bb* here represents a blank.)
4 (X'04')	2	FCLCOUNT	Number of data set name entries provided in the list.
6 (X'06')	2	FCLDSCBR	Number of DSCBs returned.
8 (X'08')	1	FCL1FLAG	Request flag byte.
	1	FCL1LIST	List contains fully-qualified data set names.
	. 1	FCL1ORDR	FCL data set name order requested.
	. . 1	FCL1EQF1	Return only format-1 or format-8 DSCBs.
	. . . 1 . . .	FCL1EQF9	Return only format-1 or format-8 and format-9 DSCBs.
	 xxxx		Reserved.
9 (X'09')	1	FCL2FLAG	Status flag byte.
	1	FCL2SEQ	CVAFFILT executed sequential VTOC access.
	. 1	FCL2SDIR	CVAFFILT executed sequential VTOC access, but did at least one direct DSCB read.
	. . xx xxxx		Reserved.
10 (X'0A')	6	FCLDRSV	Reserved.

FCLID
Must be a 4-character EBCDIC constant of 'FCL*bb*'. (*bb* here represents a blank.)

FCLCOUNT
Specifies the number of data set name entries (FCLDSN) supplied in the list. Do not change this parameter between the initial CVAFFILT call and any subsequent RESUME operations.

- If you specify a partially-qualified data set name, specify FCLCOUNT = 1. See "Partially-Qualified Names for CVAFFILT" on page 99 for the format of partially-qualified data set names.
- If you specify a list of fully-qualified names, CVAFFILT processes only the number of names specified in FCLCOUNT.

FCLDSCBR
Indicates the total number of DSCB entries (including format-1, format-2, and format-3) returned to the caller's buffers by a single CVAFFILT call.

If CVAF encounters an error after successfully processing a data set, you can:

1. Initialize FCLDSCBR to 0 before each READ and RESUME call.
2. Upon return from CVAF filter service, process the number of DSCBs indicated by FCLDSCBR.
3. Then, interpret the CVAF return code and CVSTAT.

FCL1FLAG
Define your request for ACCESS=READ with this flag byte. Any subsequent RESUME requests refer to a copy of these bits in the filter save area (FSA).

FCL1LIST
If you specify a list of fully-qualified data set names, set this bit to 1. If you specify a single partially-qualified data set name, set this bit to 0.

FCL1ORDR
If you specify that CVAF is to return DSCB chains in the data set name sequence implied by the placement of the FCLDSN elements, set this bit to 1.

Note:

1. It can improve performance to allow CVAF to determine the sequence of return for format-1 DSCBs.
2. CVAF returns DSCBs for a given data set in format-1, format-3 order. For an extended address volume, the data set order for EAS eligible data sets is a format-1, one or more format-9, and any format-3 DSCBs.
3. If you specify a single partially-qualified data set name, this field is not used.

FCL1EQF1
To have CVAF return only the format-1 or format-8 DSCBs for the data set names, set this bit to 1.

FCL1EQF9
To have CVAF return only the format-1 or format-8 and format-9 DSCBs for the data set names, set this bit to 1.

FCL2FLAG
CVAF filter indicates the following status conditions in this byte.

FCL2SEQ
This bit is set to 1 if a sequential VTOC access path is most efficient. If CVAF filter selects the direct VTOC access path, it sets this field to 0.

FCL2SDIR
This bit is set to 1 if storage limitations within the sequential VTOC access path require direct DSCB reads. CVAF initializes this bit to 0 on each ACCESS=READ and ACCESS=RESUME request. Testing this bit when CVAF filter returns control can indicate if you need to change the storage limitation.

The format of the FCL entry is shown in Table 21 on page 68.

Table 21. Format of a Filter Criteria List Entry

Offset	Bytes	Name	Description
	8	FCLDSN	Data set name information entry.
0 (X'00')	1	FCLDSNST	Data set name status.
	X'00'		Data set name not yet processed.
	X'01'		DSCBs returned successfully.
	X'02'		Data set name not found.
	X'03'		Error in DSCB chain. RESUME function recommended.
	X'04'		Error in CVAFFILT processing. RESUME not recommended.
	X'05'		Insufficient user buffer list elements. RESUME function recommended.
	X'06'		Request issued to an EAV without EADSCB=OK specified and a format-8 DSCB was found.
	X'07'		Request issued to any volume without EADSCB=OK specified where an EAS eligible data set was found.
1 (X'01')	1	FCLDSNLG	Data set name length.

Table 21. Format of a Filter Criteria List Entry (continued)

Offset	Bytes	Name	Description
2 (X'02')	1	FCL3FLAG	Flag byte.
	1	FCL3UPDT	This data set name processed during this invocation.
	. xxx xxxx		Reserved.
3 (X'03')	1	FCLDSNRV	Reserved.
4 (X'04')	4	FCLDSNA	Data set name address.

FCLDSN
Contains data set name information. This and the following fields are repeated in the FCL as a set as many times as indicated by the value in FCLCOUNT.

FCLDSNST
Indicates DSCB retrieval status.

- CVAF filter initializes this byte to 0 for ACCESS=READ requests.
- After processing the data set name for either ACCESS=READ or ACCESS=RESUME, CVAF filter updates this byte.
- ACCESS=RESUME requests do not process data set names whose FCLDSNST field is nonzero; therefore, results can be unpredictable if you alter this field.
- For partially-qualified data set name requests, CVAF filter does not post the FCLDSNST field until it has returned all DSCB chains for all qualifying data sets. CVAF filter posts the highest numeric value that applied during its processing.
- For fully-qualified data set name requests, CVAF filter returns a FCLDSNST byte for each data set name. If the value is greater than 1, CVAF filter has not returned any DSCBs for the associated data set name.

See Table 21 on page 68 for an explanation of the values in this field.

FCLDSNLG
Indicates the length of the data set name. This value is required.

FCL3FLAG
The status flag byte associated with the data set name pointed to by FCLDSNA.

FCL3UPDT
This bit indicates that CVAF filter processed the associated data set name during the current invocation of CVAFFILT.

- When initializing for either a READ or RESUME request, CVAF filter sets this bit to 0.
- When CVAF filter has completed processing for the associated data set name, it sets this bit to 1.

FCL3DSNRV
Reserved, unused.

FCLDSNA
Specifies the address of a fully-qualified data set name, or, if this is the only data set name and FCL1LIST is 0, a partially-qualified data set name. You must provide both this address and the storage area to which it points.

Example of CVAFFILT Macro Sequences

The example below demonstrates the order that you might issue CVAFFILT macro calls to complete the following tasks:

- Request the DSCBs for a list of data sets.

- Resume CVAFFILT processing interrupted because of insufficient user buffers.
- Release the kept filter save area.

The example assumes the following conditions:

- You are an authorized caller (that is, you are specifying a UCB address and IOAREA=KEEP).
- You have initialized a CVAF buffer list with the following characteristics:
  - Four buffers
  - The buffer list address in your program has the label BUFADDR
  - The same buffer list is used for ACCESS=READ and ACCESS=RESUME processing.
- You have initialized a filter criteria list as follows:
  - FCLCOUNT = 6 (You are requesting DSCB chains for six data set names.)
  - FCL1LIST = '1'B (The data set names are fully qualified.)
  - FCL1ORDR = '1'B (You want the DSCB chains returned in the order implied by data set name elements in the FCL.)
  - The six data set name elements are initialized so that they form a list requesting SYS1.A, SYS2.B, SYS3.C, SYS4.D, SYS5.E, and SYS6.F.
- The first five data sets have DSCB chain lengths or 1, 5, 2, 3, and 1, respectively, on the volume.
- The sixth data set (SYS6.F) is not defined on the volume.

To obtain an initialized CVPL, you could issue the following CVAFFILT macro (list form—does not call CVAF). This example requests the branch entry to CVAF and specifies that the caller is in supervisor state.

```
CVPLIST  CVAFFILT BRANCH=(YES,SUP),MF=L
```

To obtain the first set of DSCB chains, you could issue the following CVAFFILT macro (execute form—calls CVAF). This example specifies that the filter save area is to be kept to allow for ACCESS=RESUME calls. The IOAREA is to be kept for improved efficiency.

```
CVAFFILT ACCESS=READ,BUFLIST=bufaddr,FCL=fcladdr,
         UCB=ucbaddr,FLTAREA=KEEP,IOAREA=KEEP,
         MF=(E,CVPLIST)
```

This CVAFFILT call returns the following DSCBs:

```
Buffer    Contents of Buffer

   1      Format-1 DSCB, SYS1.A
   2      Format-1 DSCB, SYS3.C
   3      Format-3 DSCB, SYS3.C
   4      Undefined (unused)
```

CVAF filter produces return code = 4, CVSTAT = X'40' (RESUME recommended), and FCLDSCBR = 3. (CVAF returns a total of three DSCBs for the two data sets.) CVAF would not return DSCBs for data set SYS2.B because its chain contains more DSCBs than the total number of buffers provided. To retrieve the DSCBs for SYS2.B, you need to specify at least five buffers and execute another ACCESS=READ. (Even though CVAF allows you to specify a different buffer list for each READ or RESUME, or to modify the existing list between READ and RESUME calls, modifying the FCL would cause unpredictable results.) Buffer entry 4 does not have any DSCBs returned, because SYS4.D's DSCB chain size is larger than the number of remaining buffers. The FCL status information would be as follows:

```
DSN      FCLDSNST   FCL3UPDT   Comments
SYS1.A       1          1      DSCBs returned from this call
SYS2.B       5          1      DSCB chain exceeds total buffers
SYS3.C       1          1      DSCBs returned from this call
SYS4.D       0          0      DSCBs can be returned by RESUME
SYS5.E       0          0      DSCBs can be returned by RESUME
SYS6.F       0          0      DSCBs can be returned by RESUME
```

Because this CVAFFILT invocation recommends RESUME, and you specified FLTAREA=KEEP, you could use the following execute form of CVAFFILT to obtain more DSCB chains:

```
CVAFFILT ACCESS=RESUME,MF=(E,CVPLIST)
```

This CVAFFILT call returns DSCBs as follows:

```
Buffer   Contents of Buffer
   1     Format-1 DSCB, SYS4.D
   2     Format-2 DSCB, SYS4.D
   3     Format-3 DSCB, SYS4.D
   4     Format-1 DSCB, SYS5.E
```

CVAF filter produces return code = 0, CVSTAT = 0 (request completed), and updates the FCL status as follows:

```
DSN      FCLDSNST   FCL3UPDT   Comments
SYS1.A       1          0      DSCBs returned from prior call
SYS2.B       5          0      DSCB chain exceeds total buffers
SYS3.C       1          0      DSCBs returned from prior call
SYS4.D       1          1      DSCBs returned from this call
SYS5.E       1          1      DSCBs returned from this call
SYS6.F       2          1      Data set name not found
```

FCLDSCBR would contain 4. (This CVAFFILT call returns a total of four DSCBs.) CVAF filter does not return any DSCBs for SYS6.F, because its format-1 DSCB cannot be found on the volume (FCLDSNST = '2').

Because this status indicates that CVAF filter has returned all requested DSCBs, and you requested FLTAREA=KEEP and IOAREA=KEEP on the previous call, request the RLSE function as follows:

```
CVAFFILT ACCESS=RLSE,FLTAREA=NOKEEP,IOAREA=NOKEEP,
         MF=(E,CVPLIST)
```

Coding CVAF VTOC Access Macros

This section includes VTOC index information that depends on internal system logic. It is intended to help you to use CVAF macro instructions to modify the VTOC

CVAFDIR Macro Overview and Specification

For an indexed or nonindexed VTOC, you can use the CVAFDIR macro to perform the following functions:

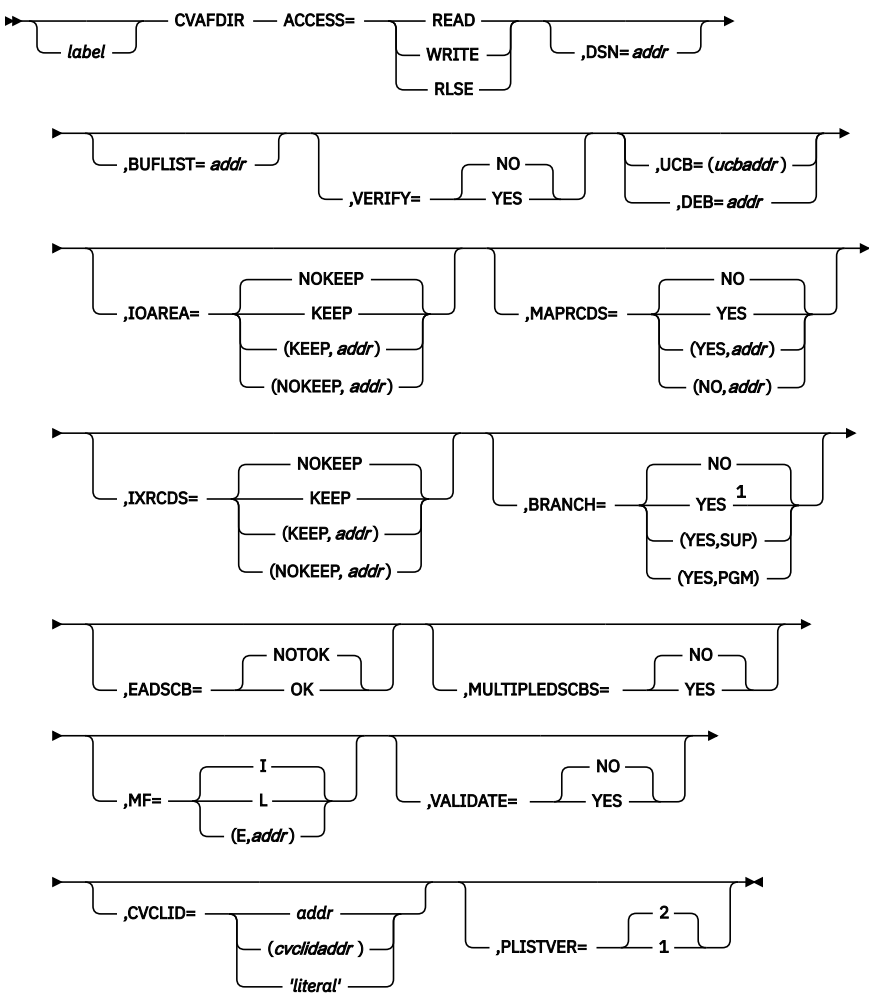
- Read or write one or more DSCBs by specifying the name of the data set they represent.
- Read or write one or more DSCBs by specifying their addresses.

In addition, for an indexed VTOC, you can use the CVAFDIR macro to perform the following functions:

- Read or write VTOC index records. (This allows calling programs to modify the VTOC index.)
- Read and retain in virtual storage the first high-level VIER, and VIERs used during an index search.
- Read and retain in virtual storage the space map VIRs.
- Free VIRs retained in virtual storage.

See "Accessing the DSCB Directly" on page 62 for additional information.

The format of the CVAFDIR macro is:



Notes:

[1] If YES is coded, the default is SUP.

ACCESS: Read or Write a DSCB or VIRs, or Release Buffer Lists

When ACCESS is READ or WRITE, a single DSCB is accessed for an indexed or nonindexed VTOC, or one or more VIRs are accessed for an indexed VTOC.

ACCESS=READ

Specifies that a single DSCB or one or more VIRs are to be read into a buffer whose address is in a buffer list.

If the buffer list is for a DSCB, only one entry is used in the buffer list. The first entry with the skip bit set to zero and a nonzero buffer address is used.

All VIRs whose buffer list entry has the skip bit off are read into a buffer.

DSN and BUFLIST are required if ACCESS=READ for a DSCB buffer list.

ACCESS=WRITE

Specifies that a single DSCB or one or more VIRs are to be written from buffers whose address is in a buffer list.

ACCESS=WRITE is permitted with BRANCH=NO only if the caller is authorized by APF.

DSN and BUFLIST are required if ACCESS=WRITE for a DSCB buffer list.

If any buffer list entry has its modified bit set, only those entries with the modified bit set are written. If no modify bits are on, all VIRs are written.

ACCESS=RLSE
Applies only to VIR buffer lists. It requests the release of one or more buffers in the VIR buffer list chain identified in the BUFLIST keyword, and the release of each buffer list for which all buffers are released.

DSN and BUFLIST are not required if ACCESS=RLSE.

Only buffers in the buffer list with the skip bit set to zero and with a nonzero buffer address are released. The buffer list is not released if any entry has the skip bit set to one.

For an indexed VTOC, if ACCESS=RLSE is coded, buffer lists and buffers pointed to by the BUFLIST keyword are released, along with buffer lists supplied in the CVAF parameter list CVMRCDS and CVIRCDS fields. If the CVMRCDS or the CVIRCDS buffers are supplied in the BUFLIST field, either directly or indirectly through chaining, the keyword MAPRCDS=YES, IXRCDS=KEEP, or MAPRCDS=(NO,0), IXRCDS=(NOKEEP,0) must be coded to prevent CVAF from freeing the buffers more than once. If buffers are released, the CVAF parameter list field pointing to the buffer list is updated.

DSN: Specify the Name of the DSCB

DSN=*addr*
Supplies the address of a 44-byte data set name of the DSCB to be accessed.

DSN is required if ACCESS=READ or WRITE and the request is to read or write a DSCB. If a 140-byte DSCB is specified:

- CVAF validity checks the storage location, but ignores the contents of the location.
- Specify an argument that points to an extent within the VTOC.

BUFLIST: Specify One or More Buffer Lists

BUFLIST=*addr*
Supplies the address of a buffer list used to read or write a DSCB or VIRs.

VALIDATE: Validate Modified fields in a DSCB

VALIDATE=YES
CVAF verifies that the DSCBs being written do not modify essential fields that should not be modified.

This parameter applies only to CVAFDIR ACCESS=WRITE.

VALIDATE=NO

CVAF does not do any checking. This is the default.

Restriction: VALIDATE applies only when writing a DSCB, and is ignored when a VIR is written.

CVCLID: Specify the address of a 4-byte field

Will put the 4-byte field indicated into the CVCLID field in the extended parameter list (requires PLISTVER=2).

This field will appear in any SMF 42 subtype 27 records generated within the CVAFDIR invocation.

CVCLID=addr
Supplies the address of a 4-byte field to be placed into the CVCLID field of the extended parameter list.

CVCLID=(cvclidaddr)
rs-type or (2-12) standard form, or rx-type or (2-12)

execute form Specifies that the address of the 4-byte CVCLID is in the given register. Recommendation: Code the address of the CVCLID parameter as register (2-12).

CVCLID='literal'
Specifies the CVCLID 4-byte field directly into the macro.

Note: Only the literal form is valid for MF=L, the other two CVCLID forms are ignored for MF=L.

PLISTVER: Specify version to be generated

PLISTVER
Specifies the version of the CVAFDIR parameter list to be generated.

PLISTVER=1
A version 1 parameter list is generated.

PLISTVER=2
A version 2 (larger) parameter list is generated that includes extended parameter list fields.

Note: PLISTVER=2 is required with the following keywords: CVCLID

Default for PLISTVER when:

MF=E: ignored, uses a parameter list generated by MF=L

MF=I: 2, if a parameter requiring PLISTVER=2 is specified, otherwise 1

MF=L: 2, if a parameter requiring PLISTVER=2 is specified, otherwise 1

Note: Coding MF=I with PLISTVER=1 and a parameter requiring PLISTVER=2 is invalid. Coding MF=L without a parameter requiring a PLISTVER2, and without explicitly setting PLISTVER=2, will generate a parameter list of length corresponding to PLISTVER=1. Using this parameter list with a parameter requiring PLISTVER=2 will result in a run-time error of Return Code = 4, Reason Code = 90 (parameter list length error).

VERIFY: Verify that a DSCB is a Format-0 DSCB

VERIFY=YES
CVAF verifies that the DSCB is a format-0 DSCB before writing the DSCB. The first four bytes of the key are compared with binary zeros. If the key does not start with 4 bytes of zeros, the DSCB is not written and an error code is returned.

VERIFY=NO
CVAF does not test the key of the DSCB. This is the default.

Restriction: VERIFY applies only when writing a 140-byte DSCB. VERIFY is ignored when a VIR is written.

UCB or DEB: Specify the VTOC to Be Accessed

UCB= ***rs-type or (2-12) standard form*** **UCB=** ***rx-type or (2-12) execute form***
Specifies the address of the UCB for the VTOC to be accessed. The UCB address can be for a captured UCB, or for an actual UCB above or below the 16MB line. Use the address of a UCB, not a UCB copy. An unauthorized caller must not use this parameter. If your program is in 31-bit mode, this address must be in 31-bit address; the high order byte is part of the address. You should not code the UCB parameter with MF=L.

Recommendation: Code the address of the UCB parameter as register (2-12). Coding an RX-Type address gives unpredictable results.

Note: You must supply a UCB address that matches the caller's AMODE. That is, AMODE=24 requires a 24 bit UCB address, while AMODE=31 requires a 31 bit UCB address.

DEB=*addr*
Supplies the address of a DEB opened to the volume table of contents (VTOC) you want to access. CVAF does not allow output requests to the VTOC or VTOC index if you specify the DEB subparameter. If you are not authorized (neither APF nor in a system key), you cannot perform any asynchronous activity (such as EXCP, CLOSE, EOV) against the data set represented by the DEB because CVAF removes the DEB from the DEB table for the duration of the CVAF call. If you are not authorized,

(neither APF authorized nor in a system key), specify a DEB address, not a UCB, to CVAFDIR. See "Identifying the Volume" on page 56 for further details.

If you supply a previously obtained I/O area through the IOAREA keyword, neither UCB nor DEB need be supplied. Otherwise, supply either a UCB or DEB. If you supply a UCB address, it is overlaid in the CVPL by the UCB address in the I/O area. If you supply both the DEB and UCB addresses in the CVPL, the DEB address is used and the UCB address in the CVPL is overlaid by the UCB address in the DEB.

IOAREA: Keep or Free the I/O Work Area

IOAREA=KEEP
Specifies that, upon completion of the CVAF request, the program should keep the CVAF I/O area associated with the CVAF parameter list. You can code IOAREA=KEEP with BRANCH=NO only if the caller is authorized (APF or system key).

If IOAREA=KEEP is coded, the caller must call CVAF with IOAREA=NOKEEP specified at some future time, whether or not any further VTOC access is required. An example of such a caller is the recovery routine of the routine that calls CVAF.

When you code IOAREA=KEEP, it allows subsequent CVAF requests to be more efficient, because the program can bypass certain initialization functions. You do not need to specify either DEB or UCB when a previously obtained CVAF I/O area is supplied; you also cannot change those values.

When IOAREA=KEEP is first issued, CVAF returns the CVAF I/O area in the CVAF parameter list (CVIOAR). Subsequent calls of CVAF can use that same parameter list, and CVAF obtains its I/O area from the CVIOAR.

When processing on the current volume is finished, release all areas that were kept.

IOAREA=(KEEP,*addr*)
Supplies the address of a previously obtained I/O area. If a different CVAF parameter list is used, the previously obtained I/O area can be passed to CVAF by coding its address as the second parameter of the IOAREA keyword.

IOAREA=NOKEEP
Causes the work area to be freed upon completion of the CVAF request.

IOAREA=(NOKEEP,*addr*)
Causes a previously obtained work area to be freed upon completion of the CVAF request.

MAPRCDS: Keep or Free MAPRCDS Buffer List and Buffers

This keyword applies to an indexed VTOC only and specifies the disposition of the MAPRCDS buffer list and buffers.

MAPRCDS=YES
Specifies that the buffer list and buffers are to be retained at the end of processing.

If no buffer list address is in the CVAF parameter list, CVAF reads the MAP VIRs into buffers it obtains. The buffer list that contains the address and RBAs of the VIRs can be accessed after processing from the CVAF parameter list field, CVMRCDS. The buffer list and VIR buffers are in your protect key: subpool 0 if you are not authorized; subpool 229 if you are.

When processing on the current volume is finished, release all areas that were kept.

MAPRCDS=(YES,*addr*)
If MAPRCDS=YES is coded and the buffer list address (CVMRCDS in CVAF parameter list) is supplied, VIRs are not read.

The CVMRCDS buffer list used in CVAFDIR macro can be passed to another CVAF macro call through the MAPRCDS keyword.

If MAPRCDS=YES is coded for a nonindexed VTOC, the function is performed, but an error code is returned.

MAPRCDS=NO
If MAPRCDS=NO is coded, all the buffers without the skip bit on in the buffer list whose address is in the CVMRCDS field of the CVPL are freed. If all the buffers are freed, the buffer list is also freed.

MAPRCDS=(NO,*addr*)
Frees buffer lists and buffers previously obtained by CVAF.

You must free buffer lists and buffers obtained by CVAF. This can be done in one of three ways:

- By coding MAPRCDS=NO on the CVAFDIR macro that obtained the buffers
- By coding MAPRCDS=NO on a subsequent CVAF macro
- By coding CVAFDIR ACCESS=RLSE and providing the address of the buffer list in the BUFLIST keyword.

Requirement: You must enqueue the VTOC and reserve the unit to maintain the integrity of MAP records read.

IXRCDS: Retain VIERS in Virtual Storage

This keyword applies to indexed VTOCs only.

IXRCDS=KEEP
Specifies that VIERs read into storage are to be kept in virtual storage. The VIERs are retained even if processing cannot complete successfully. The CVAF parameter list in field CVIRCDS contains the address of a buffer list with the VIR buffer addresses and RBAs of the VIERs read.

The index search function dynamically updates the buffer list and, when necessary, obtains additional buffer lists and chains them together.

If IXRCDS=KEEP is specified and no buffer list is supplied to CVAF in the CVPL, CVAF obtains a buffer list and buffers and reads the first high-level VIER. The address of the buffer list is placed in the CVIRCDS field of the CVPL.

The buffer list and VIR buffers are in your protect key. The subpool is 0 if you are not authorized; it is subpool 229 if you are.

If IXRCDS=KEEP is coded for a nonindexed VTOC, a request to read or write a DSCB is performed, but an error code is returned.

When processing on the current volume is finished, release all areas that were kept.

IXRCDS=(KEEP,*addr*)
The index records buffer list address from one CVAF request is being passed to this CVAF parameter list by specifying its address as the second parameter in the IXRCDS keyword.

IXRCDS=NOKEEP
If IXRCDS=NOKEEP is coded, the VIERs that are accessed (if any) are not retained. Furthermore, the buffer list supplied in the CVIRCDS field in the CVAF parameter list is released, as are all buffers found in the buffer list. If the skip bit is set in any entry in the buffer list, the buffer and buffer list are not freed. This is the default.

IXRCDS=(NOKEEP,*addr*)
Specifies that previously accessed VIERs are not to be retained.

You must free buffer lists and buffers obtained by CVAF. This can be done in one of three ways:

- By coding IXRCDS=NOKEEP on the CVAFDIR macro that obtained the buffers
- By coding IXRCDS=NOKEEP on a subsequent CVAF macro
- By coding CVAFDIR ACCESS=RLSE and providing the address of the buffer list in the BUFLIST keyword.

Requirement: You must enqueue the VTOC and reserve the unit to maintain the integrity of the VIERs read.

BRANCH: Specify the Entry to the Macro

BRANCH=(YES,SUP)
Requests the branch entry. Your program be in supervisor state. Protect key checking is bypassed.

If BRANCH=YES is coded, an 18-word save area must be supplied. No lock can be held on entry to CVAF. SRB mode is not allowed.

BRANCH=YES
Equivalent to BRANCH=(YES,SUP), because SUP is the default when YES is coded. Protect key checking is bypassed.

BRANCH=(YES,PGM)
Requests the branch entry. Your program be authorized by APF and be in problem state. Protect key checking is bypassed.

BRANCH=NO
Requests the SVC entry. If any output operations are requested, your program must be authorized by APF. Protect key checking is performed. This is the default.

EADSCB: Specify the support level for extended attribute DSCBs

EADSCB=OK
This specification indicates that the calling program supports extended attribute DSCBs. An extended address volume may have these DSCBs allocated to it. The returned DSCBs (format-3, format-8) may contain extent descriptors described by 28-bit cylinder addresses or DSCBs (format-9) that contain additional attribute information.

For search calls where the data set name is passed (CVAFDIR ACCESS=READ,BFLEARG=0), a CVAFDIR request will fail if the EADSCB=OK indicator is not set and the DSCB associated with this data set name is a format-8 DSCB.

For seek calls where the record address is passed (CVAFDIR ACCESS=READ,BFLEARG=*cchhr*), a CVAFDIR request issued to an EAV volume will be failed if the EADSCB=OK indicator is not set and the DSCB associated with this address is a format-8 or format-9 DSCB.

For seek calls where the record address is passed (CVAFDIR ACCESS=READ,BFLEARG=*cchhr*), and MULTIPLEDSCBS=NO is specified or defaulted to NO, a CVAFDIR request will faileif the EADSCB=OK indicator is not set and the DSCB associated with this address is a format-3 DSCB that contain track addresses above 65,520 cylinders.

The failing error code for these cases will be reflected as follows:

- CVAF status code (CVSTAT) set to STAT082.
- Return code 4.

EADSCB=OK will set the CV4EADOK indicator in the CVPL.

For all other calls, the EADSCB=OK keyword is ignored.

EADSCB=NOTOK
Indicates a calling program does not support extended attribute DSCBs. The specification of this will resolve to the CV4EADOK indicator in the CVPL to be set off. This is the default.

MULTIPLEDSCBS: Specify whether multiple DSCBs should be processed

MULTIPLEDSCBS=NO
This specification indicates that the calling program requests that only one DSCB should be processed. This is the default for MF=L and MF=I forms of the CVAFDIR macro. When the MULTIPLEDSCBS keyword is not specified on the MF=E form, the existing setting of CV4MULTD is left unchanged. When MULTIPLEDSCBS=NO is specified or defaulted, only the first available buffer list entry is processed.

MULTIPLEDSCBS=YES
This specification indicates that the calling program requests to read/write multiple DSCBs to/from a buffer list that contains more than one buffer list entry. This parameter causes an indicator in the CVPL, CV4MULTD, to be set on. Multiple DSCB processing for reads and writes is requested by specifying the MULTIPLEDSCBS=YES keyword and providing a buffer list that contains more than one buffer list entry (BFLHNOE>1).

MF: Specify the Form of the Macro

This keyword specifies whether the list, execute, or normal form of the macro is requested.

MF=I
If I is coded or if neither L nor E is coded, the CVAF parameter list is generated and CVAF is called. This is the normal form of the macro.

MF=L
Indicates the list form of the macro. A parameter list is generated, but CVAF is not called.

MF=(E,*addr*)
Indicates the execute form of the macro. The CVAF parameter list whose address is in *addr* can be modified by this form of the macro.

Return Codes from CVAFDIR

On return from CVAF, register 1 contains the address of the CVPL (CVAF parameter list), and register 15 contains one of the following return codes:

Return Code	Meaning
0 (X'00')	The request was successful. However, if the CVAFDIR request is to read or write a DSCB and a VTOC index structure error is encountered, the CVSTAT field indicates the structure error that was encountered. (CVSTAT code descriptions are in *z/OS DFSMSdfp Diagnosis*.)
4 (X'04')	An error occurred. The CVSTAT field in the CVPL contains an indication of the cause of the error. (CVSTAT code descriptions are in *z/OS DFSMSdfp Diagnosis*.)
8 (X'08')	Invalid VTOC index structure while processing a request to read or write a VTOC index record. The CVSTAT field in the CVPL contains an indication of the cause of the error. (CVSTAT code descriptions are in *z/OS DFSMSdfp Diagnosis*.)
12 (X'0C')	The CVAF parameter list is not in your protect key or is not valid (the ID is not valid, or the length field is incorrect, or the CVFCTN (function code) field is not valid or is not supported in this release). The CVPL has not been modified.
16 (X'10')	An I/O error was encountered.

Example of Using the CVAFDIR Macro with a VTOC

This example uses the CVAFDIR macro to read a DSCB of a given data set name and determines whether the DSCB is for a partitioned data set. The address of the 44-byte data set name is supplied to the program in register 5 (labeled RDSN in the example). The address of a DEB open to the VTOC is supplied to the program in register 4 (labeled RDEB in the example).

The buffer list is in the program and is generated by the ICVAFBFL macro. The DSCB buffer is in the program and is generated by the IECSDSL1 macro.

Example of CVAFDIR Macro with VTOC Part 1 of 2

```
DIRXMP1  CSECT
         STM   14,12,12(RSAVE)
         BALR  12,0
         USING *,12
         ST    RSAVE,SAVEAREA+4
         LA    RWORK,SAVEAREA
```

```
         ST    RWORK,8(,RSAVE)
         LR    RSAVE,RWORK
*************************************************************
*
*        REGISTERS
*
*************************************************************
REG1     EQU   1                 REGISTER 1
RWORK    EQU   3                 WORK REGISTER
RDEB     EQU   4                 DEB ADDRESS
RDSN     EQU   5                 ADDRESS OF DATA SET NAME
RSAVE    EQU   13                SAVE AREA ADDRESS
REG15    EQU   15                RETURN CODE REGISTER 15
*************************************************************
*
*        RETURN CODES
*
*************************************************************
PDSRTN   EQU   0                 DATA SET A PDS RETURN CODE
NOTFND   EQU   4                 DATA SET NOT FOUND RETURN CODE
NOTPDS   EQU   8                 DATA SET NOT A PDS RETURN CODE
UNEXPECD EQU   12                UNEXPECTED ERROR RETURN CODE
*************************************************************
*
*        READ DSCB INTO DS1FMTID.
*        DATA SET NAME ADDRESS SUPPLIED IN RDSN.
*        ADDRESS OF DEB OPEN TO VTOC SUPPLIED IN RDEB.
*        DETERMINE IF DATA SET IS A PARTITIONED DATA SET.
*        THIS PROGRAM IS NEITHER REENTRANT NOR REUSABLE.
*
*************************************************************
         XC   BUFLIST(BFLHLN+BFLELN),BUFLIST ZERO BUFFER LIST
         OI   BFLHFL,BFLHDSCB      DSCBS TO BE READ WITH BUFFER LIST
         MVI  BFLHNOE,1            ONE BUFFER LIST ENTRY
         LA   RWORK,DS1FMTID       ADDRESS OF DSCB BUFFER
         ST   RWORK,BFLEBUF        PLACE IN BUFFER LIST
         OI   BFLEFL,BFLECHR       CCHHR OF DSCB RETURNED BY CVAF
         MVI  BFLELTH,DSCBLTH      DATA PORTION OF DSCB READ - DSN     *
                                   SUPPLIED IN CVPL
         MVC  DS1DSNAM,0(RDSN)     MOVE IN DATA SET NAME TO WORKAREA
         CVAFDIR ACCESS=READ,DSN=DS1DSNAM,BUFLIST=BUFLIST,DEB=(RDEB)
         USING CVPL,REG1           ADDRESSABILITY TO CVPL
         LTR  REG15,REG15          ANY ERROR
         BZ   NOERROR              BRANCH IF NOT
```

Example of CVAFDIR Macro with VTOC Part 2 of 2

```
*************************************************************
*
*        DETERMINE WHAT ERROR IS
*
*************************************************************
         C    REG15,ERROR4         IS RETURN CODE 4
         BNE  OTHERERR             BRANCH IF NOT 4
         CLI  CVSTAT,STAT001       IS IT DATA SET NAME NOT FOUND?
         BNE  OTHERERR             BRANCH IF NOT
         DROP REG1                 ADDRESSABILITY TO CVPL NOT NEEDED
*************************************************************
*
*        DATA SET NAME NOT FOUND
*
*************************************************************
         L    RSAVE,4(,RSAVE)
         RETURN (14,12),RC=NOTFND  SET UP DATA SET NOT FOUND ERROR
NOERROR  EQU  *                    DSCB READ
         MVC  F1CCHHR,BFLEARG      MOVE CCHHR OF FORMAT 1/4 DSCB TO    *
                                   WORKAREA
         CLI  DS1FMTID,C'4'        IS DSCB A FORMAT 4 DSCB
         BE   NOTF1                BRANCH IF YES. NOT A FORMAT 1
         TM   DS1DSORG,DS1DSGPO    IS FORMAT 1 DSCB FOR PARTITIONED    *
                                   DATA SET
         BO   PDS                  BRANCH IF PDS
NOTF1    EQU  *                    DSCB IS NOT A PDS
         L    RSAVE,4(,RSAVE)
         RETURN (14,12),RC=NOTPDS  SET UP NOT PDS RETURN CODE
PDS      EQU  *                    DATA SET IS PARTITIONED
         L    RSAVE,4(,RSAVE)
         RETURN (14,12),RC=PDSRTN  SET UP PDS RETURN CODE
OTHERERR EQU  *                    UNEXPECTED ERROR
         L    RSAVE,4(,RSAVE)
```

```
         RETURN (14,12),RC=UNEXPECD

ERROR4   DC   F'4'                ERROR RETURN CODE 4
BUFLIST  ICVAFBFL DSECT=NO        BUFFER LIST
         IECSDSL1 (1)             FORMAT 1 DSCB DATA SET NAME AND     *
                                  BUFFER
DSCBLTH  EQU  *-IECSDSL1-L'DS1DSNAM LENGTH OF DATA PORTION OF DSCB
F1CCHHR  DS   XL5                 CCHHR OF DSCB
SAVEAREA DS   18F                 SAVE AREA
CVPL     ICVAFPL ,                CVPL MAPPING MACRO

         END
```

Example of Using the CVAFDIR Macro with an Indexed VTOC

This example uses the CVAFDIR macro to read one or more DSCBs from a VTOC. The UCB is supplied to the program in register 4 (labeled RUCB). The TTR of each DSCB read is to be returned to the caller as well as the return code received back from CVAFDIR. This program must be APF authorized.

The address of a parameter list is supplied to the program in register 5 (labeled RLIST). The parameter list contains one or more 4-word entries. The format of each 4-word entry is mapped by the LISTMAP DSECT. The first word contains the address of the data set name of the DSCB to be read. The second word contains the address of the 96-byte buffer into which the DSCB is to be read. The third word contains the address of the 3-byte TTR of the DSCB read. The fourth word contains the return code received back from CVAFDIR for the DSCB read.

The CVPL is generated by a list form of the CVAFDIR macro at label CVPL. The BUFLIST, IXRCDS, IOAREA, and BRANCH keywords are coded on the list form of the macro. IXRCDS=KEEP and IOAREA=KEEP are coded to avoid overhead if two or more DSCBs are to be read. BRANCH=(YES,PGM) is coded in the list form of the CVAFDIR macro to cause the CVPL to have the CV1PGM bit set to one; this indicates to CVAF that the caller is authorized by APF and not in supervisor state. The execute forms of the CVAFDIR macro then specify BRANCH=YES, and not BRANCH=(YES,PGM), because the CV1PGM bit is set in the list form of the macro.

The CVAFDIR macro with ACCESS=RLSE is coded before the program exits to release the CVAF I/O area and the index records buffer list. BUFLIST=0 is coded because no user-supplied buffer list is to be released; BUFLIST was coded on the list form of the CVAFDIR macro and, therefore, is in the CVBUFL field of the CVPL. This field must be set to zero for the release function.

```
DIRXMP2  CSECT
         STM   14,12,12(13)
         BALR  12,0
         USING *,12
         ST    13,SAVEAREA+4
         LA    RWORK,SAVEAREA
         ST    RWORK,8(,13)
         LR    13,RWORK
*
*************************************************************
*
* REGISTERS
*
*************************************************************
*
RWORK    EQU   3                  WORK REGISTER
RUCB     EQU   4                  UCB ADDRESS SUPPLIED BY CALLER
RLIST    EQU   5                  ADDRESS OF PARAMETER LIST (SUPPLIED)
RDSN     EQU   6                  ADDRESS OF DATA SET NAME
RTTR     EQU   7                  ADDRESS OF TTR
REG15    EQU   15                 RETURN CODE REGISTER 15
*
*************************************************************
*
* UCB ADDRESS SUPPLIED IN RUCB.
* READ DSCB OF DATA SET NAME SUPPLIED.
* RETURN TTR OF DSCB.
* RETURN RETURN CODE FOR CVAFDIR REQUEST FOR DSCB.
* ADDRESS OF PARAMETER LIST IN RLIST.
* WORD 1 OF PARAMETER LIST = ADDRESS OF DATA SET NAME
* WORD 2 OF PARAMETER LIST = ADDRESS OF DSCB TO BE RETURNED
* WORD 3 OF PARAMETER LIST = ADDRESS OF TTR TO BE RETURNED
* WORD 4 OF PARAMETER LIST = RETURN CODE RETURNED FROM CVAFDIR FOR DSCB
```

```
* WORDS 1-4 CAN BE DUPLICATED FOR MULTIPLE REQUESTS
* THE HIGH ORDER BIT OF WORD 3 SET TO X'80' FOR THE LAST ENTRY ONLY.
*
************************************************************
*
         USING LISTMAP,RLIST      ADDRESSABILITY TO PARMLIST
TOPLOOP  EQU   *                  LOOP FOR EACH DSCB
         XC    BUFLIST(BFLHLN+BFLELN),BUFLIST ZERO BUFFER LIST
         OI    BFLHFL,BFLHDSCB    DSCBS TO BE READ WITH BUFFER LIST
         MVI   BFLHNOE,1          ONE BUFFER LIST ENTRY
         L     RWORK,LISTDSCB     ADDRESS OF DSCB BUFFER
         ST    RWORK,BFLEBUF      PLACE IN BUFFER LIST
         OI    BFLEFL,BFLETTR     TTR OF DSCB RETURNED BY CVAF
         MVI   BFLELTH,DSCBLTH    DATA PORTION OF DSCB READ - DSN      *
                                  SUPPLIED IN CVPL
         L     RDSN,LISTDSN       ADDRESS OF DATA SET NAME
         CVAFDIR DSN=(RDSN),UCB=(RUCB),MF=(E,CVPL),BRANCH=YES
         LA    RTTR,LISTARG       ADDRESS OF TTR TO BE RETURNED
         USING TTRMAP,RTTR        MAP OF TTR
         LTR   REG15,REG15        ANY ERROR
         BZ    NOERROR            BRANCH IF NOT
         XC    TTR,TTR            ZERO TTR INDICATING NO DSCB
         ST    REG15,LISTRC       STORE RC FROM CVAFDIR INTO LISTRC
         B     RELOOP             GET NEXT ENTRY
NOERROR  EQU   *                  DSCB READ
         MVC   TTR(3),BFLEATTR    RETURN TTR OF DSCB
         ST    REG15,LISTRC       STORE RC FROM CVAFDIR INTO LISTRC
RELOOP   EQU   *                  GET NEXT ENTRY
         TM    LASTLIST,LASTBIT   IS IT LAST ENTRY IN LIST?
         LA    RLIST,NEXTLIST     GET NEXT ENTRY
         BZ    TOPLOOP            PROCESS NEXT LIST
         CVAFDIR ACCESS=RLSE,     RELEASE CVAF OBTAINED AREAS          *
               IOAREA=NOKEEP,     RELEASE IOAREA                       *
               IXRCDS=NOKEEP,     RELEASE VIER BUFFER LIST             *
               BUFLIST=0,         NO USER BUFFER LIST SUPPLIED TO RLSE *
               BRANCH=YES,        BRANCH ENTER CVAF                    *
               MF=(E,CVPL)
         L     13,SAVEAREA+4
         RETURN (14,12)

BUFLIST  ICVAFBFL DSECT=NO        BUFFER LIST

SAVEAREA DS    18F                REGISTER SAVE AREA
LISTMAP  DSECT
LISTDSN  DS    F                  ADDRESS OF DATA SET NAME
LISTDSCB DS    F                  ADDRESS OF BUFFER FOR DSCB TO BE
*                                 RETURNED
LISTARG  DS    0F                 ADDRESS OF FLAG AND TTR
*                                 RETURNED
LASTLIST DS    X                  FIRST BYTE
LASTBIT  EQU   X'80'              LAST ENTRY IN LIST
LISTTTR  DS    XL3                REMAINDER OF TTR ADDRESS
LISTRC   DS    F                  RETURN CODE FROM CVAFDIR FOR THIS DSN
NEXTLIST EQU   *                  NEXT LIST
DSCB     DSECT
         IECSDSL1 (1)
DSCBLTH  EQU   *-DSCB-L'DS1DSNAM  LENGTH OF DATA PORTION OF DSCB
TTRMAP   DSECT
TTRFLG   DS    XL1                FLAG VALUE
TTR      DS    XL3                TTR VALUE
DIRXMP2  CSECT
CVPL     CVAFDIR ACCESS=READ,BUFLIST=BUFLIST,MF=L,                     *
               IOAREA=KEEP,       KEEP IOAREA TO AVOID OVERHEAD        *
               IXRCDS=KEEP,       KEEP VIERS FOR 2ND / SUBSEQUENT CALLS*
               BRANCH=(YES,PGM)   CALLED IN PROGRAM STATE BUT APF      *
                                  AUTHORIZED SO UCB IS SUPPLIED
         ORG   CVPL               OVERLAY CVPL WITH EXPANSION OF MAP
CVPLMAP  ICVAFPL DSECT=NO

         END
```

Example of Using the CVAFDIR macro to read multiple DSCBs

This example uses the CVAFDIR macro to read multiple DSCBs for a VSAM data set that has 123 extents using the MULTIPLEDSCBS=YES parameter. Refer to the documentation within the sample source for program logic and expected output.

The following is the sample JCL used to execute the sample source module below.

```
//*
//*********************************
//* JCL TO EXECUTE CVDIR027 MODULE *
//*********************************
//*
//STEP001 EXEC PGM=CVDIR027
//STEPLIB   DD DISP=SHR,DSN=YOUR.TEST.LOAD
//SYSPRINT  DD SYSOUT=*
//CVAFDD    DD DISP=SHR,UNIT=3390,VOL=SER=1P9503  /* VSAM01 VOLSER  */
//OUTDD     DD SYSOUT=*                           /* OUTPUT DATASET */
//*
```

The following is the CVAFDIR sample source to read multiple DSCBs.

```
CVDIR027 CSECT
CVDIR027 AMODE 31
CVDIR027 RMODE 24
*
*********************************************************************
*                                                                   *
*    CVDIR027 - MODULE THAT ISSUES THE CVAFDIR MACRO WHICH RETURNS    *
*               THE DSCBS FOR A GIVEN DATASET USING THE KEYWORDS      *
*               MULTIPLEDSCBS=YES AND EADSCB=OK.                      *
*                                                                     *
*               THIS MODULE USES A PASSED DSN (SEARCH) AND ISSUES THE *
*               CVAFDIR MACRO TO PERFORM A READ OF ALL DSCBS FOR THE  *
*               DATASET.  THE DATASET PROCESSED IS VSAM AND HAS 123   *
*               EXTENTS.  THE DSN IS CVAFDIR1.VSAM01.DATA.            *
*                                                                     *
*               THE CVAFDIR MACRO CALL WILL USE THE FOLLOWING:        *
*               EADSCB=OK CODED                                       *
*               MULTIPLEDSCBS=YES CODED                               *
*                                                                     *
*                                                                     *
*               THIS MODULE HAS BEEN WRITTEN TO RUN AT THE Z/OS 1.10  *
*               LEVEL AND WILL CREATE A SLIGHTLY DIFFERENT OUTPUT     *
*               REPORT DEPENDENT UPON DEVICE TYPE (EAV OR NON EAV).   *
*               THE NUMBER OF BUFFER LIST ENTRIES NEEDED WILL BE 11   *
*               FOR A NON EAV DEVICE AND THE NUMBER OF BUFFER LIST    *
*               ENTRIES NEEDED WILL BE 12 FOR AN EAV DEVICE TO        *
*               ACCOUNT FOR THE FORMAT 9 DSCB.                        *
*                                                                     *
*                                                                     *
*               THIS MODULE WILL CREATE AN OUTPUT REPORT DIRECTED TO  *
*               THE OUTDD DD THAT SHOULD LOOK LIKE THE FOLLOWING:     *
*                                                                     *
*---------------------------------------------------------------------*
*                                                                     *
*                        NON EAV VOLUME                               *
*                        --------------                               *
*                                                                     *
*             CVDIR027 START OF OUTPUT MESSAGES                       *
*                                                                     *
*              PROCESSING DSN: CVAFDIR1.VSAM01.DATA                   *
*              CVAFDIR  CALL: EADSCB=OK AND MULTIPLEDSCBS=YES CODED   *
*              CV4EADOK BIT SET / EADSCB=OK                           *
*              CV4MULTD BIT SET / MULTIPLEDSCBS=YES                   *
*              RC00 VERIFIED - THE REQUEST WAS SUCCESSFUL             *
*               X"00"    DEC"000"    00 - CVSTAT CODE VERIFIED        *
*              BUFFER LIST ENTRIES PROVIDED: 12                       *
*              BUFFER LIST ENTRIES NEEDED  : 11                       *
*                                                                     *
*             CVDIR027 END OF OUTPUT MESSAGES                         *
*                                                                     *
*                                                                     *
*                                                                     *
*                           EAV VOLUME                                *
*                           ----------                                *
*                                                                     *
*             CVDIR027 START OF OUTPUT MESSAGES                       *
*                                                                     *
*              PROCESSING DSN: CVAFDIR1.VSAM01.DATA                   *
*              CVAFDIR  CALL: EADSCB=OK AND MULTIPLEDSCBS=YES CODED   *
*              CV4EADOK BIT SET / EADSCB=OK                           *
*              CV4MULTD BIT SET / MULTIPLEDSCBS=YES                   *
*              RC00 VERIFIED - THE REQUEST WAS SUCCESSFUL             *
*               X"00"    DEC"000"    00 - CVSTAT CODE VERIFIED        *
*              BUFFER LIST ENTRIES PROVIDED: 12                       *
*              BUFFER LIST ENTRIES NEEDED  : 12                       *
*                                                                     *
```

```
*             CVDIR027 END OF OUTPUT MESSAGES                         *
*                                                                     *
*                                                                     *
*                                                                     *
***********************************************************************
*                                                                     *
*    CVDIR027 - LOGIC NOTES                                           *
*                                                                     *
*    THIS MODULE WILL PERFORM THE FOLLOWING:                          *
*                                                                     *
*    INITIALIZATION                                                   *
*    - OBTAIN THE NECESSARY INFORMATION FROM THE DASD VOLUME          *
*    - OPEN AN OUTPUT FILE AND WRITE THE NECESSARY OUTPUT MESSAGES    *
*                                                                     *
*    MAINLINE                                                         *
*    - INVOKE SETU1RTN TO SETUP BUFFER LIST FOR 12 ENTRIES            *
*    - INVOKE VSAM1RTN TO PROCESS VSAM01 DSN                          *
*      - INVOKE READRTN - EADSCB=OK AND MULTIPLEDSCBS=YES CODED       *
*        - CHECK CV4FL BIT SETTINGS AFTER CVAFDIR CALL                *
*      - CHECK RC AND CVSTAT CODES RETURNED                           *
*      - CHECK BUFFERS PROVIDED AND BUFFERS NEEDED FROM BUFFER LIST   *
*      - WRITE OUTPUT RECS                                            *
*      - ISSUE CVAFDIR TO RELEASE WORK AREAS                          *
*      - WRITE OUTPUT RECS                                            *
*                                                                     *
*                                                                     *
*    FINALIZATION                                                     *
*    - CLOSE THE OUTPUT FILE                                          *
*    - EXIT                                                           *
*                                                                     *
*                                                                     *
*    CVDIR027 - JOB INFORMATION                                       *
*                                                                     *
*    NORMAL END OF JOB:                                               *
*    - RC=00 AND OUTDD OUTPUT AS DETAILED ABOVE                       *
*                                                                     *
*                                                                     *
*    ABNORMAL END OF JOB:                                             *
*    - ABEND 100 - ERROR OPENING VTOC ON THE DASD VOLUME THAT IS      *
*                  ASSOCIATED WITH THE CVAFDD DD STATEMENT            *
*    - ABEND 101 - ERROR OPENING THE OUTDD DATASET                    *
*    - ABEND 102 - ERROR CLOSING THE OUTDD DATASET                    *
*                                                                     *
*                                                                     *
*                                                                     *
*                                                                     *
***********************************************************************
*
***********************************************************************
*                                                                     *
* HOUSEKEEPING                                                        *
* - SAVE CALLER'S REGISTERS AND ESTABLISH A NEW REGISTER SAVE AREA     *
*                                                                     *
***********************************************************************
*
         STM   R14,R12,12(R13)    STANDARD LINKAGE CONVENTION
         BALR  R11,0              DCL R11 AS IMPLIED BASE REG
         USING BASE,R11,R12       R12 IS ALSO BASE REG
BASE     L     R12,BASEADDR       SET UP ADDRESSING FOR R12
         B     CV000000           BRANCH AROUND DECLARES
BASEADDR DC    A(BASE+4096)       ADDRESSING FOR R12
CV000000 DS    0H                 CONTINUE...
         ST    R13,SAVE+4         SAVE PTR TO CALLER'S SAVE AREA
         LA    R14,SAVE           GET ADDRESS OF THE NEW SAVE AREA
         ST    R14,8(,R13)        CHAIN CALLER'S AREA TO OURS
         LR    R13,R14            ESTABLISH THE NEW SAVE AREA
*
***********************************************************************
*                                                                     *
* INITIALIZATION                                                      *
*                                                                     *
***********************************************************************
*
INITIAL  DS    0H                 INITIALIZATION SECTION
         BAL   R14,IDVOLRTN       INVOKE RTN TO IDENTIFY THE VOLUME
         OPEN  (OUTFILE,(OUTPUT)) OPEN THE OUTDD OUTPUT FILE
         TM    OUTFILE+48,X'10'   TEST IF FILE IS OPEN (OUTFILE)
         BO    INIT0010           IF OPEN OK - BRANCH AROUND ABEND
         ABEND 101                ELSE ISSUE USER ABEND 101
INIT0010 DS    0H                 FILE IS OPEN WRITE START MESSAGE
         PUT   OUTFILE,STRTMSG    WRITE A RECORD TO THE OUTPUT FILE
         PUT   OUTFILE,BLNKLINE   WRITE A RECORD TO THE OUTPUT FILE
```

```
*
*********************************************************************
*                                                                   *
* MAINLINE                                                          *
*                                                                   *
*********************************************************************
*
MAINLINE DS    0H                 MAINLINE SECTION
*
MAIN0010 DS    0H                 PROCESS CVAFDIR1.VSAM01.DATA DATASET
*
         BAL   R14,SETU1RTN       SETUP FOR CVAFDIR - 12 ENTRIES
         BAL   R14,VSAM1RTN       PROCESS VSAM01 DATASET ROUTINE
         PUT   OUTFILE,BLNKLINE   WRITE A RECORD TO THE OUTPUT FILE
*
*
*********************************************************************
*                                                                   *
* FINALIZATION                                                      *
*                                                                   *
*********************************************************************
*
FINAL    DS    0H                 FINALIZATION SECTION
         PUT   OUTFILE,ENDMSG     WRITE A RECORD TO THE OUTPUT FILE
         CLOSE (OUTFILE)          CLOSE OUTPUT FILE
         C     R15,RCODE00        IF FILE CLOSE IS OK
         BE    FINL0010            BRANCH AROUND ABEND
         ABEND 102                 ELSE ISSUE USER ABEND 102
FINL0010 DS    0H                 EXIT MODULE
         L     R13,4(R13)         RESTORE REGISTER
         LM    R14,R12,12(R13)    RESTORE CALLERS REGISTERS
         LA    R15,0              SET RC TO 0
         BR    R14                RETURN TO CALLER
*
*********************************************************************
*                          IDVOLRTN                                 *
*     - OBTAIN THE NECESSARY INFORMATION FROM THE DASD VOLUME       *
*********************************************************************
*
IDVOLRTN DS    0H                 IDENTIFY VOLUME ROUTINE
         ST    R14,IDVLSAVE       STORE C(R14) INTO SAVE AREA
         RDJFCB (VTOCDCB,(INPUT)) READ JFCB / OPEN VTOC
         MVI   JFCB1,X'04'        PUT IN ID FOR FORMAT 4
         MVC   JFCB1+1(43),JFCB1  SETUP FOR VTOC OPEN
         OPEN  (VTOCDCB,(INPUT)),TYPE=J  OPEN VTOC (OPEN TYPE=J)
         TM    VTOCDCB+48,X'10'   IF OPEN OF VTOC IS OK
         BO    IDVL0010           BRANCH AROUND ABEND
         ABEND 100                ELSE ISSUE USER ABEND 100
IDVL0010 DS    0H
         SLR   R4,R4              INIT REG4 FOR DEB PTR
         SLR   R5,R5              INIT REG5 FOR UCB PTR
         ICM   R4,B'0111',VTOCDCB+45  GET DEB ADDRESS
         ST    R4,DEBADD          STORE C(R4) INTO DEBADD
         ICM   R5,B'0111',33(R4)  GET UCB ADDRESS
         ST    R5,UCBADD          STORE UCB ADDRESS
IDVLEXIT DS    0H                 EXIT FROM IDVOLRTN
         L     R14,IDVLSAVE       LOAD C(IDVLSAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                          SETU1RTN                                 *
*            - SETUP FOR CVAFDIR - BFLHNOE = 12                     *
*            - WILL SETUP 12 ENTRIES FOR CVAFDIR CALL               *
*            - CVAFDIR READ CALL (SEARCH) PASSED DSN                *
*              - 96 BYTE BUFFER - 1ST BUFFER                        *
*              - 140 BYTE BUFFERS - REMAINING BUFFERS               *
*              - DSNAME=DSN WILL BE USED FOR SEARCH                 *
*              - CCHHR = ZERO                                       *
*********************************************************************
*
SETU1RTN DS    0H                 SETUP FOR CVAFDIR CALL
         ST    R14,SET1SAVE       STORE C(R14) INTO SAVE AREA
         LA    R4,BUFLHDR         GET ADDR OF BUF LIST HEADER
         L     R5,NBRENT          LOAD R5 WITH NBR OF ENTRIES (12)
         USING BFLHDR,R4          GET ADDRESSABILITY TO HEADER
         MVI   BFLHNOE,TWELVE     INDICATE 12 ENTRIES
         MVI   BFLHKEY,BFLHDSCB   INDICATE READ DSCB
         LA    R6,DSCBBUF         LOAD R6 WITH ADDR OF 1ST DSCB BUFFER
         LA    R7,BUFLIST1        LOAD R7 WITH ADDR OF BUFLIST1
         USING BFLE,R7            GET ADDRESSABILITY TO ENTRY
*
* INITIALIZE 1ST ENTRY
```

```
*
         OI    BFLEFL,BFLECHR     INDICATE CCHHR TO BE READ
         MVC   BFLEARG(5),CCHHR0  SET ZEROES FOR ARGUMENT
         MVI   BFLELTH,DSCBL96    GET LENGTH OF BUFFER
         ST    R6,BFLEBUF         PUT DSCB BUF ADDR IN ENTRY
         LA    R7,ENTLENG(,R7)    INCREMENT ADDR TO NEXT ENTRY
         LA    R6,DSCBL96(,R6)    INCREMENT ADDR TO NEXT DSCB BUFFER
         S     R5,ONE             C(R5) = C(R5) - 1
*
* INITIALIZE REMAINING ENTRIES
*
SETU0010 DS    0H                 INIT REMAINING 140 BYTE BUFFERS
         OI    BFLEFL,BFLECHR     INDICATE CCHHR TO BE READ
         MVI   BFLELTH,DSCBL140   GET LENGTH OF BUFFER
         ST    R6,BFLEBUF         PUT DSCB BUF ADDR IN ENTRY
         LA    R7,ENTLENG(,R7)    INCREMENT ADDR TO NEXT ENTRY
         LA    R6,DSCBL140(,R6)   INCREMENT ADDR TO NEXT DSCB BUFFER
         BCT   R5,SETU0010        BRANCH TO SETU0010 IF MORE ENTRIES
SET1EXIT DS    0H                 EXIT FROM SETU1RTN
         L     R14,SET1SAVE       LOAD C(SAVE AREA) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                        VSAM1RTN                                   *
*            - PROCESS VSAM01 DATASET ROUTINE                       *
*              - CVAFDIR READ: EADSCB=OK AND MULTIPLEDSCBS=YES CODED  *
*              - CHECK RC / CVSTAT CODES                            *
*              - ISSUE CVAFDIR RELEASE                              *
*********************************************************************
*
VSAM1RTN DS    0H                 PROCESS VSAM01 DATASET ROUTINE
         ST    R14,VSAMSAVE       STORE C(R14) INTO SAVE AREA
         MVC   DSNAME(44),VSAM01  DSNAME=CVAFDIR1.VSAM01.DATA
         PUT   OUTFILE,VSAM1MSG   WRITE A RECORD TO THE OUTPUT FILE
         PUT   OUTFILE,CALLMR11   WRITE A RECORD TO THE OUTPUT FILE
         BAL   R14,READRTN        READRTN - EADSCB=OK/MULTIPLEDSCBS=YES
         L     R15,RETCODE        LOAD R15 WITH SAVED RETURN CODE
         C     R15,RCODE00        IF RC FROM READ IS EQUAL TO ZERO
         BE    VSAM0010           BRANCH TO VSAM0010
         PUT   OUTFILE,UNEXPMS2   ELSE WRITE RC ERROR MESSAGE
         B     VSAM0020           BRANCH TO VSAM0020
VSAM0010 DS    0H
         PUT   OUTFILE,RC00MSG    WRITE RC00 MESSAGE
VSAM0020 DS    0H
         LA    R9,CVAFDIR         GET ADDRESS OF CVAF PARM LIST
         USING CVPL,R9            GET ADDRESSABILITY TO CVPL
         CLI   CVSTAT,STAT000     IF CVSTAT FROM READ IS EQUAL TO ZERO
         BE    VSAM0030           BRANCH TO VSAM0030
         PUT   OUTFILE,UNEXPMS3   ELSE WRITE CVSTAT ERROR MESSAGE
         B     VSAM0040           BRANCH TO VSAM0040
VSAM0030 DS    0H
         PUT   OUTFILE,CV00MSG    WRITE CV00 MESSAGE
VSAM0040 DS    0H
         CLI   BFLHNOE,X'0C'      IF NUMBER OF BUFFERS PROVIDED = 12
         BE    VSAM0050           BRANCH TO VSAM0050
         PUT   OUTFILE,UNEXPMS4   ELSE WRITE BUFFER ERROR MESSAGE
         B     VSAM0060           BRANCH TO VSAM0060
VSAM0050 DS    0H
         PUT   OUTFILE,BUFSMSG1   WRITE BUFFER MESSAGE
VSAM0060 DS    0H
         CLI   BFLHNOEN,X'0B'     IF NUMBER OF BUFFERS NEEDED = 11
         BE    VSAM0070           BRANCH TO VSAM0070
         CLI   BFLHNOEN,X'0C'     IF NUMBER OF BUFFERS NEEDED = 12
         BE    VSAM0080           BRANCH TO VSAM0080
         PUT   OUTFILE,UNEXPMS5   ELSE WRITE BUFFER ERROR MESSAGE
         B     VSAM0090           BRANCH TO VSAM0090
VSAM0070 DS    0H
         PUT   OUTFILE,BUFNMSG1   WRITE BUFFER MESSAGE
         B     VSAM0090           BRANCH TO VSAM0090
VSAM0080 DS    0H
         PUT   OUTFILE,BUFNMSG2   WRITE BUFFER MESSAGE
VSAM0090 DS    0H
*
         CVAFDIR ACCESS=RLSE,IXRCDS=NOKEEP,BUFLIST=0,                  X
               MF=(E,CVAFDIR)
*
         C     R15,RCODE00        IF RC FROM RLSE IS EQUAL TO ZERO
         BE    VSAMEXIT           BRANCH TO VSAMEXIT
         PUT   OUTFILE,UNEXPMS6   ELSE WRITE RLSE ERROR MESSAGE
*
VSAMEXIT DS    0H                 EXIT FROM ROUTINE
         DROP  R9                 DROP R9
```

```
         L     R14,VSAMSAVE       LOAD C(SAVE AREA) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                         READRTN                                   *
*            - CVAFDIR READ ROUTINE                                 *
*              MULTIPLEDSCBS=YES IS CODED                           *
*              EADSCB=OK IS CODED                                   *
*            - CHECK CV4FL BIT SETTINGS AFTER READ                  *
*            - REPORT ON CV4FL BIT SETTINGS FOR CV4EADOK AND CVMULTD *
*********************************************************************
*
READRTN  DS    0H                 CVAFDIR READ ROUTINE
         ST    R14,READSAVE       STORE C(R14) INTO SAVE AREA
         L     R2,DEBADD          LOAD R2 WITH DEB ADDRESS
*
         CVAFDIR ACCESS=READ,DEB=(R2),BUFLIST=BUFLHDR,MAPRCDS=YES,     X
               DSN=DSNAME,MULTIPLEDSCBS=YES,EADSCB=OK,                 X
               MF=(E,CVAFDIR)
*
         ST    R15,RETCODE        STORE RC INTO RETCODE
*
         LA    R9,CVAFDIR         GET ADDRESS OF CVAF PARM LIST
         USING CVPL,R9            GET ADDRESSABILITY TO CVPL
         CLI   CVFL4,CV4EADOK     IF CVFL4 = X'10' CV4EADOK ONLY
         BE    READ0010           BRANCH TO READ0010
         CLI   CVFL4,CV4MULTD     IF CVFL4 = X'08' CV4MULTD ONLY
         BE    READ0020           BRANCH TO READ0020
         CLI   CVFL4,BOTH         IF CVFL4 = X'18' CV4MULTD/CV4EADOK
         BE    READ0030           BRANCH TO READ0030
         PUT   OUTFILE,UNEXPMS1   ELSE WRITE ERROR RECORD PL
         B     READEXIT           BRANCH TO EXIT ROUTINE
READ0010 DS    0H                 WRITE MESSAGES X'10'
         PUT   OUTFILE,OKMSG      WRITE OK MSG RECORD
         PUT   OUTFILE,NOMULTMS   WRITE NO MULTI MSG RECORD
         B     READEXIT           BRANCH TO EXIT ROUTINE
READ0020 DS    0H                 WRITE MESSAGES X'08'
         PUT   OUTFILE,NOTOKMSG   WRITE NOTOK MSG RECORD
         PUT   OUTFILE,MULTMSG    WRITE MULTI MSG RECORD
         B     READEXIT           BRANCH TO EXIT ROUTINE
READ0030 DS    0H                 WRITE MESSAGES X'18'
         PUT   OUTFILE,OKMSG      WRITE OK MSG RECORD
         PUT   OUTFILE,MULTMSG    WRITE MULTI MSG RECORD
READEXIT DS    0H                 EXIT FROM READRTN
         DROP  R9                 DROP R9
         L     R14,READSAVE       LOAD C(SAVE AREA) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                       WORKING  STORAGE                            *
*********************************************************************
*
         DS    0D
         DC    CL36'CVDIR027-WORKING STORAGE BEGINS HERE'
*
*********************************************************************
*                          EQUATES                                  *
*********************************************************************
*
R0       EQU    0
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
BOTH     EQU   X'18'
TWELVE   EQU   X'0C'
*
*********************************************************************
*                         SAVE AREAS                                *
*********************************************************************
*
```

```
SAVE     DC    18F'0'             MAIN PROGRAM SAVE AREA
IDVLSAVE DC    F'0'               IDENTIFY VOLUME ROUTINE SAVE AREA
READSAVE DC    F'0'               CVAFDIR READ ROUTINE SAVE AREA
SET1SAVE DC    F'0'               SETUP ROUTINE SAVE AREA
VSAMSAVE DC    F'0'               VSAM01 ROUTINE SAVE AREA
*
*********************************************************************
*                         CONSTANTS                                 *
*********************************************************************
*
CCHHR0   DS    XL5'00'            CCHHR ARGUMENT - ZERO
NBRENT   DC    F'12'              CONSTANT-12 NUMBER OF BUFFER ENTRIES
RCODE00  DC    F'0'               RETURN CODE 0
ONE      DC    F'1'               CONSTANT - ONE
VSAM01   DC    CL44'CVAFDIR1.VSAM01.DATA'
*
*********************************************************************
*                    PROGRAM BUFFERS                                *
*********************************************************************
*
BUFLIST  DS    0F                 BUFFER LIST DECLARATIONS
BUFLHDR  DC    2F'0'              BUFFER LIST HEADER
BUFLIST1 DC    3F'0'              BUFFER LIST ENTRY 1
ENTLENG  EQU   *-BUFLIST1         ENTRY LENGTH - 12
BUFLIST2 DC    3F'0'              BUFFER LIST ENTRY 2
BUFLIST3 DC    3F'0'              BUFFER LIST ENTRY 3
BUFLIST4 DC    3F'0'              BUFFER LIST ENTRY 4
BUFLIST5 DC    3F'0'              BUFFER LIST ENTRY 5
BUFLIST6 DC    3F'0'              BUFFER LIST ENTRY 6
BUFLIST7 DC    3F'0'              BUFFER LIST ENTRY 7
BUFLIST8 DC    3F'0'              BUFFER LIST ENTRY 8
BUFLIST9 DC    3F'0'              BUFFER LIST ENTRY 9
BUFLISTA DC    3F'0'              BUFFER LIST ENTRY 10
BUFLISTB DC    3F'0'              BUFFER LIST ENTRY 11
BUFLISTC DC    3F'0'              BUFFER LIST ENTRY 12
*
DSCBBUF  DS    XL96               DSCB BUFFER DECLARATION 1 - 96 BYTE
DSCBL96  EQU   *-DSCBBUF          LENGTH - 96
DSCBBUF2 DS    XL140              DSCB BUFFER DECLARATION 2 - 140 BYTE
DSCBL140 EQU   *-DSCBBUF2         LENGTH - 140
DSCBBUF3 DS    XL140              DSCB BUFFER DECLARATION 3 - 140 BYTE
DSCBBUF4 DS    XL140              DSCB BUFFER DECLARATION 4 - 140 BYTE
DSCBBUF5 DS    XL140              DSCB BUFFER DECLARATION 5 - 140 BYTE
DSCBBUF6 DS    XL140              DSCB BUFFER DECLARATION 6 - 140 BYTE
DSCBBUF7 DS    XL140              DSCB BUFFER DECLARATION 7 - 140 BYTE
DSCBBUF8 DS    XL140              DSCB BUFFER DECLARATION 8 - 140 BYTE
DSCBBUF9 DS    XL140              DSCB BUFFER DECLARATION 9 - 140 BYTE
DSCBBUFA DS    XL140              DSCB BUFFER DECLARATION 10- 140 BYTE
DSCBBUFB DS    XL140              DSCB BUFFER DECLARATION 11- 140 BYTE
DSCBBUFC DS    XL140              DSCB BUFFER DECLARATION 12- 140 BYTE
*
*********************************************************************
*                    PROGRAM MESSAGES                               *
*********************************************************************
*
BLNKLINE DC    CL80' '
STRTMSG  DC    CL80'CVDIR027 START OF OUTPUT MESSAGES                '
ENDMSG   DC    CL80'CVDIR027 END OF OUTPUT MESSAGES                  '
VSAM1MSG DC    CL80' PROCESSING DSN: CVAFDIR1.VSAM01.DATA            '
CALLMR11 DC CL80' CVAFDIR  CALL: EADSCB=OK AND MULTIPLEDSCBS=YES CODED'
OKMSG    DC    CL80' CV4EADOK BIT SET / EADSCB=OK                    '
NOTOKMSG DC    CL80' CV4EADOK BIT NOT SET / EADSCB=NOTOK             '
MULTMSG  DC    CL80' CV4MULTD BIT SET / MULTIPLEDSCBS=YES            '
NOMULTMS DC    CL80' CV4MULTD BIT NOT SET / MULTIPLEDSCBS=NO         '
RC00MSG  DC    CL80' RC00 VERIFIED - THE REQUEST WAS SUCCESSFUL      '
CV00MSG  DC    CL80'  X"00"    DEC"000"    00 - CVSTAT CODE VERIFIED  '
BUFSMSG1 DC    CL80' BUFFER LIST ENTRIES PROVIDED: 12                '
BUFNMSG1 DC    CL80' BUFFER LIST ENTRIES NEEDED  : 11                '
BUFNMSG2 DC    CL80' BUFFER LIST ENTRIES NEEDED  : 12                '
UNEXPMS1 DC    CL80' ERROR: UNEXPECTED BIT SETTING FOR CVFL4         '
UNEXPMS2 DC    CL80' ERROR: UNEXPECTED RETURN CODE FROM CVAFDIR READ  '
UNEXPMS3 DC    CL80' ERROR: UNEXPECTED CVSTAT CODE FROM CVAFDIR READ  '
UNEXPMS4 DC    CL80' ERROR: UNEXPECTED NUMBER OF BUFFERS PROVIDED     '
UNEXPMS5 DC    CL80' ERROR: UNEXPECTED NUMBER OF BUFFERS NEEDED       '
UNEXPMS6 DC    CL80' ERROR: UNEXPECTED RETURN CODE FROM CVAFDIR RLSE  '
*
*********************************************************************
*                          WORK AREAS                               *
*********************************************************************
*
DEBADD   DC    F'0'               DEB ADDRESS SAVE AREA
UCBADD   DC    F'0'               UCB ADDRESS SAVE AREA
```

```
RETCODE  DC    F'999'             RETURN CODE SAVE AREA
DSNAME   DS    CL44               DSNAME
*
*************************************************************************
*                     DCB - OUTPUT FILE (OUTFILE)                       *
*************************************************************************
*
OUTFILE  DCB   DDNAME=OUTDD,                                            X
               DSORG=PS,                                                X
               RECFM=FB,                                                X
               LRECL=80,                                                X
               MACRF=PM
*
*************************************************************************
*                      VTOC DCB AREA                                    *
*************************************************************************
*
VTOCDCB  DCB   DDNAME=CVAFDD,MACRF=E,EXLST=XLST1,DSORG=PS,DCBE=VTOCDCBE
XLST1    DC    X'87'
         DC    AL3(JFCB1)
JFCB1    DS    0CL176
TESTNAME DS    CL44
         DS    CL8
         DS    BL1
         DS    CL123
*
VTOCDCBE DCBE  EADSCB=OK
*
*************************************************************************
*                        CVAF DECLARATIONS                              *
*************************************************************************
*
CVAFDIR  CVAFDIR MF=L
*
*************************************************************************
*                        MAPPING MACROS                                 *
*************************************************************************
*
         ICVAFPL
         ICVAFBFL
         DSECT
         IECSDSL1 (1,3,8,9)
*
*
         END   CVDIR027           END OF CVDIR027
/*
```

CVAFDSM Macro Overview and Specification

The CVAFDSM macro is used to obtain volume information for an indexed or nonindexed VTOC.

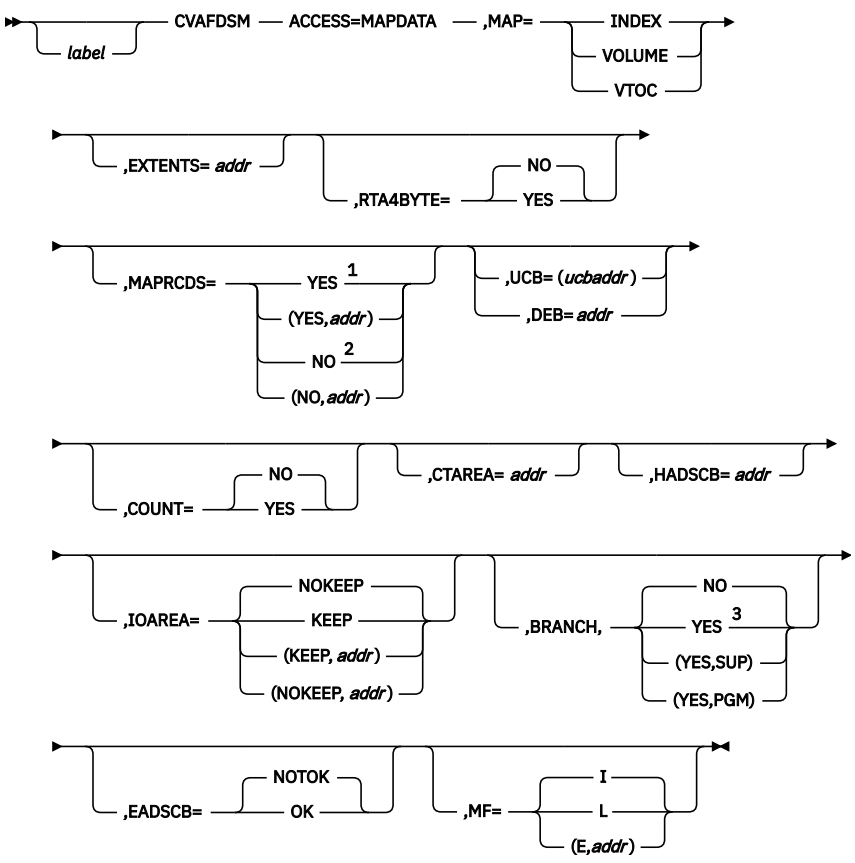
The CVAFDSM macro can be used for an indexed VTOC to obtain:

- One or more extents that describe unallocated space on the volume
- A count of free DSCBs on the VTOC
- A count of free VTOC index records in the VTOC index.
- A value that represents the highest allocated DSCB as determined by the VTOC INDEX.

The CVAFDSM macro can be used for an nonindexed VTOC to obtain:

- One or more extents that describe unallocated space on the volume

The format of the CVAFDSM macro is:



Notes:

[1] Default if MF=I.
[2] Default if MF=L or MF=(E, addr).
[3] If YES is coded, the default is SUP.

ACCESS: Request Information from Index Space Maps or the VTOC

ACCESS=MAPDATA

Obtains data from index space maps or free space DSCBs.

The following data is available from the index space maps:

- The number of format-0 DSCBs (the data is obtained from the VTOC map of DSCBs)
- The number of unallocated VIRs in the index (the data is obtained from the VTOC index map)
- The number (and location) of extents of unallocated pack space (the data is obtained from the VTOC pack space map).

The following data is available from nonindexed VTOCs:

- The number (and location) of extents of unallocated pack space.

MAP: Identify the Map to Be Accessed

MAP=INDEX

Specifies that the VTOC index map (VIXM) is to be accessed and a count of unallocated VIRs returned. COUNT=YES must also be coded.

MAP=VOLUME

For indexed VTOCs, specifies that the VTOC pack space map (VPSM) is to be accessed and information on unallocated extents of pack space returned.

For nonindexed VTOCs, specifies that the information from the free space DSCBs on unallocated extents is to be returned.

For indexed and nonindexed VTOCs, EXTENTS=*addr* and COUNT=NO must also be coded.

MAP=VTOC
Specifies that the VTOC map of DSCBs (VMDS) is to be accessed and a count of format-0 DSCBs returned. COUNT=YES must also be coded.

EXTENTS: Storage Area Where Extents Are Returned

EXTENTS=*addr*
Specifies the address where extent information is to be returned. You specify this parameter only when you also specify MAP=VOLUME to request that unallocated space information from the volume is to be returned.

When RTA4BYTE=YES is specified the following occurs:

- Information about free extents uses mapping macro ICVEDT02 (see "Using Macro ICVEDT02 to Map the Extents Area" on page 61 for ICVEDT02 format).
- EXTENTS= specifies the address of a control block (ICVEDT02) used to pass 4 byte relative track addresses of the unallocated space. You provide this storage, and you must use this mapping macro to initialize it.

  Prior to calling CVAF, initialize the ICVEDT02 control block as follows:

  – Place "ICVEDT02" in the first 8 bytes (DT2X7EYE).
  – Place the total area length, in bytes, in DT2X7LEN. This value is 36 + (8 * the value in DT2X7ENT).
  – Set the control block level number to "1" (DT2X7LEV). (Note that this value can be different in a possible future release, if IBM makes changes that effect the control block.)
  – Set the remaining fields in the control block static area DT2X7FLG through DT2X7RE2 to zero prior to calling CVAF for the first time. Leave these fields unchanged from the way the previous call returned them, when calling for additional extents.
  – Set DT2X7ENT to the total number of extent descriptor entries that will fit in the storage you provide.
  – Place the relative track address (RTA) at which CVAF should start the search into the first four bytes of the first extent area DT2RTAST(1).

    CVAF updates the first extent entry with information about the next free extent found that has a higher starting RTA than that provided. Each subsequent extent entry is filled in with information about additional free space extents (in ascending RTA order).

    For all calls, if all the unallocated extents from the volume are returned before the provided storage area is filled, the remaining entries are set to zero. CVAF will now set return code 4 in register 15, and will set the CVSTAT field to X'20' to indicate end of data.

    If End Of Data is reached before the provided storage area is filled with unallocated extents, CVAF will set return code 4 in register 15. If return code 0 is set in register 15, you can call CVAF again to get the remaining unallocated space information if there is any. Do NOT modify ANY header information in ICVEDT02, as CVAF can have saved internal use restart information there. Instead, copy the last ending RTA+1 (DT2RTAED) returned from the previous CVAF search into DT2RTAST.

When RTA4BYTE=NO is specified or defaulted the following occurs:

- Information about free extents has the format of *XXYYZ* (see "RTA4BYTE: Specify the Type of Extent Area Used" on page 91 for *XXYYZ* format).
- EXTENTS= is the address of a 1-byte count field containing the number of 5-byte entries that follow. You provide this storage area. The length of the area, in bytes, is 1 + (*count* * 5), where *count* is the value of the first byte of the area. The first two bytes ("XX") of the first 5-byte extent area entry, is the relative track address (RTA) at which CVAF will start the search. CVAF updates the first entry with information about the next free extent found that has a higher starting RTA than that supplied.

Each subsequent entry is filled in with information about additional free extents (in ascending track address order).

- CVAF can be called multiple times, as needed, to retrieve more extents than the area can hold in a single call. The first extent returned is the first free extent after the relative track address ("XX") recorded in the first extent (*XXYYZ*) in the area.
- To retrieve the first free extent on the volume, set "XX" in the first entry to zero. When calling additional times, set "XX" in the first entry to the LAST relative track address returned by the previous call.

Recommendation: If you use larger volumes, specify RTA4BYTE=YES when you request extent information. If an extent is beyond the 64x1024 tracks boundary when the program specifies RTA4BYTE=NO or allows the default, the CVAF request fails with a CVSTAT of STAT075.

RTA4BYTE: Specify the Type of Extent Area Used

RTA4BYTE=YES

Specifies that the extents area contains pairs of addresses in the format RTA RTA+1, where RTA is the four byte relative address of the first track of the extent, and RTA+1 is the four byte relative address of the last track of the extent plus 1.

You must use the macro ICVEDT02 to map the extent area if you specify RTA4BYTE=YES. See Table 19 on page 62 for a description.

RTA4BYTE=YES is a required parameter for an nonindexed VTOC, and optional for an indexed VTOC.

RTA4BYTE=NO

Specifies that the extents area is in the format *XXYYZ* where:

XX	The relative track address of the first track of the extent
YY	The number of whole cylinders in the extent
Z	The number of additional tracks in the extent.

If you do not specify the RTA4BYTE parameter, the default is RTA4BYTE=NO.

MAPRCDS: Keep or Free MAPRCDS Buffer List and Buffers

MAPRCDS=YES

Specifies that the buffer list and buffers are to be retained at the end of the function.

If MAPRCDS=YES is specified and no buffer list is supplied through the CVAF parameter list, CVAF reads the VIRs into buffers obtained by CVAF. The buffer list that contains the address and RBAs of the VIRs can be accessed after the CVAF call from the CVAF parameter list field, CVMRCDS. The buffer list and VIR buffers are in the caller's protect key: subpool 0 if the caller is not authorized; subpool 229 if the caller is authorized.

MAPRCDS=YES is the default if MF=I is specified or defaulted.

When processing on the current volume is finished, release all areas that were kept.

MAPRCDS=(YES,*addr*)

If MAPRCDS=YES is coded, but the buffer list address (CVMRCDS in CVAF parameter list) is supplied, the VIRs are not read.

The CVMRCDS buffer list from one CVAF call can be passed to another CVAF macro call through the MAPRCDS keyword.

MAPRCDS=NO

Specifies that the MAP records buffers and buffer list are freed upon completion of the CVAFDSM function.

NO is the default if MF=L is specified.

MAPRCDS=(NO,*addr*)
Frees buffer lists and buffers previously obtained by CVAF.

Buffer lists and buffers obtained by CVAF must be freed by the caller. This can be done in one of the following ways:

- By coding MAPRCDS=NO on the call that obtained the buffers
- By coding MAPRCDS=NO on a subsequent CVAF call
- By coding CVAFDIR ACCESS=RLSE and providing the buffer list in the BUFLIST keyword.

If MF=(E,*addr*) is coded and MAPRCDS is not coded, the parameter list value of MAPRCDS is not changed.

Requirement: You must enqueue the VTOC and reserve the unit to maintain the integrity of the MAP records read.

UCB or DEB: Specify the VTOC to Be Accessed

UCB= *rs-type or (2-12) standard form* UCB= *rx-type or (2-12) execute form*
Specifies the address of the UCB for the VTOC to be accessed. The UCB address can be for a captured UCB, or for an actual UCB above or below the 16MB line. Use the address of a UCB, not a UCB copy. An unauthorized caller can not use this parameter. If your program is in 31-bit mode, this address must be in 31-bit address; the high order byte is part of the address. You should not code the UCB parameter with MF=L.

DEB=*addr*
Supplies the address of a DEB opened to the VTOC you want to access. CVAF does not allow output requests to the VTOC or VTOC index if you specify the DEB subparameter. Without authorization, you cannot perform any asynchronous activity (such as EXCP, CLOSE, EOV) against the data set represented by the DEB because CVAF removes the DEB from the DEB table for the duration of the CVAF call. If you are not authorized (neither APF authorized nor in a system key), specify a DEB address, not a UCB, to CVAFDSM. See "Identifying the Volume" on page 56 for further details.

If you supply a previously obtained I/O area through the IOAREA keyword, neither UCB nor DEB need be supplied. Otherwise, supply either a UCB or DEB. If you supply a UCB address, it is overlaid in the CVPL by the UCB address in the I/O area. If you supply both the UCB and the DEB addresses in the CVPL, the DEB address is used and the UCB address in the CVPL is overlaid by the UCB address in the DEB.

COUNT: Obtain a Count of Unallocated DSCBs or VIRs

COUNT=YES
Indicates that a count of unallocated DSCBs or VIRs in the designated space map is requested. MAP=VTOC or MAP=INDEX must be specified if COUNT=YES is coded.

COUNT=NO
Indicates that a count of unallocated DSCBs or VIRs is not desired but, rather, information on free space on the pack is desired. MAP=VOLUME must be coded if COUNT=NO is coded or the default.

CTAREA: Supply a Field to Contain the Number of Format-0 DSCBs

CTAREA=*addr*
Supplies the address of a 4-byte field to contain the number of format-0 DSCBs when COUNT=YES, MAP=VTOC is specified; or the number of unallocated VIRs in the VTOC index when COUNT=YES, MAP=INDEX is specified.

HADSCB: Supply a Field to Contain the CCHHR of the Highest Allocated DSCB

HADSCB=*addr*
Supplies the address of a 5–byte field to contain the CCHHR of the highest allocated DSCB in the VTOC when COUNT=YES and MAP=VTOC are specified.

IOAREA: Keep or Free the I/O Work Area

IOAREA=KEEP
Specifies that the CVAF I/O area associated with the CVAF parameter list is to be kept upon completion of the CVAF request. IOAREA=KEEP can be coded with BRANCH=NO only if the caller is authorized (APF or system key).

If IOAREA=KEEP is coded, the caller must call CVAF with IOAREA=NOKEEP specified at some future time, whether or not any further VTOC access is required. An example of such a caller is the recovery routine of the caller of CVAF.

Coding IOAREA=KEEP allows subsequent CVAF requests to be more efficient, as certain initialization functions can be bypassed. Neither DEB nor UCB need be specified when a previously obtained CVAF I/O area is supplied; neither can they be changed.

When IOAREA=KEEP is first issued, CVAF returns the CVAF I/O area in the CVAF parameter list (CVIOAR). Subsequent calls of CVAF can use that same parameter list, and CVAF obtains its I/O area from the CVIOAR.

When processing on the current volume is finished, release all areas that were kept.

IOAREA=(KEEP,*addr*)
Supplies the address of a previously obtained I/O area. If a different CVAF parameter list is used, the previously obtained CVAF I/O area can be passed to CVAF by coding its address as the second parameter of the IOAREA keyword.

IOAREA=NOKEEP
Causes the work area to be freed upon completion of the CVAF request. This is the default.

IOAREA=(NOKEEP,*addr*)
Causes a previously obtained work area to be freed upon completion of the CVAF request.

BRANCH: Specify the Entry to the Macro

BRANCH=(YES,SUP)
Requests the branch entry. The caller must be in supervisor state. Protect key checking is bypassed.

If BRANCH=YES is coded, an 18-word save area must be supplied. No lock can be held on entry to CVAF. SRB mode is not allowed.

BRANCH=YES
Equivalent to BRANCH=(YES,SUP), because SUP is the default when YES is coded. Protect key checking is bypassed.

BRANCH=(YES,PGM)
Requests the branch entry. The caller must be APF authorized and in problem state. Protect key checking is bypassed.

BRANCH=NO
Requests the SVC entry. The caller must be APF authorized if any output operations are requested. Protect key checking is performed. This is the default.

EADSCB: Specify the support level for extended attribute DSCBs

EADSCB=OK
This specification indicates that the calling program supports RTAs that could contain tracks an extended address volume.

For calls that request unallocated space (ACCESS=MAPDATA, MAP=VOLUME, RTA4BYTE=YES and EXTENTS=address), a CVAFDSM request issued to an EAV volume will be failed if the EADSCB=OK indicator is not set.

The failing error code for these cases will be reflected as follows:

- CVAF status code (CVSTAT) set to STAT082.

- Return code 4.

EADSCB=OK will set the CV4EADOK indicator in the CVPL. All other calls to CVAFDSM are allowed and EADSCB=OK will be ignored.

EADSCB=NOTOK
Indicates a calling program does not support extended attribute DSCBs. EADSCB=NOTOK will set the CV4EADOK indicator in the CVPL to off. This is the default.

MF: Specify the Form of the Macro

This keyword specifies whether the list, execute, or normal form of the macro is requested.

MF=I
If I is coded or if neither L nor E is coded, the CVAF parameter list is generated, as is code, to call CVAF. This is the default.

MF=L
Indicates the list form of the macro. A parameter list is generated, but code to call CVAF is not generated.

MF=(E,*addr*)
Indicates the execute form of the macro. The remote CVAF parameter list supplied as *addr* is used in, and can be modified by, the execute form of the macro.

Return Codes from CVAFDSM

On return from CVAF, register 1 contains the address of the CVPL (CVAF parameter list), and register 15 contains one of the following return codes:

Return Code	Meaning
0 (X'00')	The request was successful.
4 (X'04')	End of data (CVSTAT is set to decimal 32), or an error was encountered. The CVSTAT field in the CVPL contains an indication of the cause of the error. (CVSTAT code descriptions are in *z/OS DFSMSdfp Diagnosis*.)
8 (X'08')	Invalid VTOC index structure. CVSTAT contains an indication of the cause of the error. (CVSTAT code descriptions are in *z/OS DFSMSdfp Diagnosis*.)
12 (X'0C')	One of the following • The CVAF parameter list is not in your protect key. • The CVAF parameter list protect key is invalid. • The CVAF parameter list ID is invalid. • The CVAF parameter list length is incorrect. • The CVAF parameter list has not been modified. • The function code (CVFCTN) field is not valid. • The function code (CVFCTN) field is not supported by this release.
16 (X'10')	An I/O error was encountered.

CVAFFILT Macro Overview and Specification

You can use the CVAFFILT macro to invoke the CVAF filter service. You can also use it to map or initialize the CVAF parameter list (CVPL). CVAF filter retrieves data set DSCB chains from an indexed or nonindexed VTOC and places them in buffers you provide. You can request the DSCBs for a single partially-qualified data set name or for a list of fully-qualified data set names.

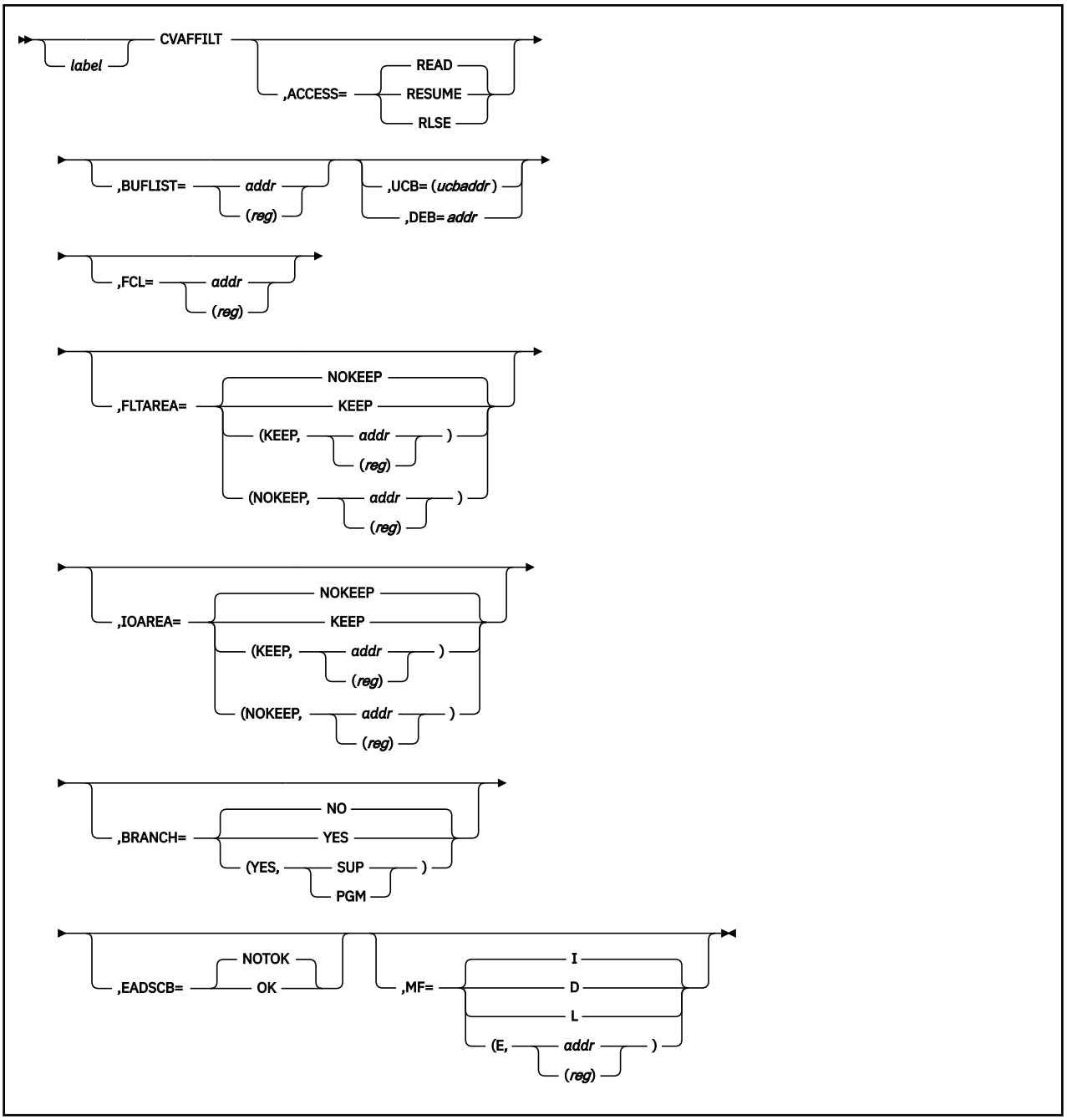
Identify a specific DASD device and provide both a filter criteria list (FCL) defining the request, and a CVAF buffer list (with buffers) for DSCB return. The format of the two elements of the FCL is shown in Table 20

on page 67 and Table 21 on page 68. The format of the buffer list is shown in "Using Buffer Lists" on page 60. CVAFFILT returns a complete set of DSCBs in the order that they are chained in the VTOC (format-1, format-2, then format-3).

Keywords coded on the list form of the macro need not be coded on the execute form. Keywords coded on one CVAFFILT call remain in effect for subsequent calls unless overridden, if you use the same CVAFFILT parameter list.

See "Reading Sets of DSCBs with CVAF Filter" on page 65 for additional information.

The format of the CVAFFILT macro is:



Restriction: For the first operand following CVAFFILT, do not code the leading comma.

Control Block Address Resolution: Keyword=*addr* or *(reg)*

You, as the caller, either define or reference the control blocks needed by CVAF filter. Caller-defined control blocks are: BUFLIST, CVPL, and FCL. Caller-referenced control blocks are: DEB, FLTAREA, IOAREA,

and UCB. The CVAFFILT macro generates different instructions for `keyword=`*addr* and `keyword=`(*reg*) depending upon whether you are specifying a defined or referenced control block.

- When you specify any control block's address as (*reg*), the CVAFFILT macro assumes that the register specified contains that address.
- When you specify a "defined" control block's address as *addr*, the CVAFFILT macro assumes that the specified location is that of the control block itself. The macro generates a load address instruction (LA) to obtain the control block's address.
- When you specify a "referenced" control block's address as *addr*, the CVAFFILT macro assumes that the specified location is that of a word containing the address of the control block. The macro generates a load instruction (L) to obtain the control block's address.

ACCESS: Retrieve a DSCB, or Release FLTAREA and/or IOAREA

ACCESS=READ
Retrieves all DSCBs associated with the data set names specified in the filter criteria list (FCL), placing them in your buffers. You can select (filter) the retrieved DSCBs by providing either a list of one or more fully-qualified names, or a single partially-qualified name, using single or double asterisk notation. (See the example of partially-qualified names in "Partially-Qualified Names for CVAFFILT" on page 99.)

If the number of buffers is not large enough to hold all the requested DSCBs, CVAFFILT indicates this in the CVSTAT status byte of the CVAF parameter list (CVPL). You can resume the READ function by issuing a call with ACCESS=RESUME. See "Codes Put in the CVSTAT Field" on page 128.

When selecting DSCBs by partially-qualified name, CVAFFILT uses only the first data set name in the FCL list. Set the FCLCOUNT count field in the FCL to 1 or CVAFFILT returns error code 63 in the CVSTAT status byte of the CVPL. The DSCBs returned by CVAFFILT might not be in sequence by data set name; however, the DSCBs for each data set are always in order (format-1, format-2, format-3).

When selecting DSCBs by fully-qualified names, you can request that CVAF filter return the DSCBs for the selected data set names in the data set name order implied by the FCL. See the FCL1ORDR flag in Table 20 on page 67.

Always test the status byte of each data set name in the FCL list to ensure successful completion. (Some error conditions result in failure to return a data set's DSCBs.) See the FCLDSNST byte in Table 21 on page 68.

ACCESS=RESUME
Resumes a previously initiated READ or RESUME function that was terminated because you did not provide enough buffers to contain all the requested DSCBs. For the RESUME function to execute correctly, the keyword FLTAREA=KEEP must be coded in each of the previous READ and RESUME function calls.

ACCESS=RLSE
Releases the previously kept filter save area (FLTAREA) and/or CVAF I/O work area (IOAREA).

BUFLIST: Specify a Buffer List

BUFLIST=*addr* or (*reg*)
Supplies the address of a buffer list used to read DSCBs. When you specify ACCESS=RLSE, the BUFLIST keyword is required for the standard form of the macro. See the format of the buffer list header and buffer list entry in Table 17 on page 60 and Table 18 on page 61, respectively.

UCB or DEB: Specify the VTOC to Be Accessed

UCB= *rs-type or (2-12) standard form* UCB= *rx-type or (2-12) execute form*
Specifies the address of the UCB for the VTOC to be accessed. The UCB address might be for a captured UCB, or for an actual UCB above or below the 16 MB line. Use the address of a UCB, not a UCB copy. An unauthorized caller must not use this parameter. If your program is in 31-bit mode, this

address must be in 31-bit address; the high order byte is part of the address. You should not code the UCB parameter with MF=L.

DEB=*addr or (reg)*
Supplies the address of a DEB opened to the VTOC you want to access. If you are not authorized, specify a DEB address, not a UCB, to CVAFFILT; also, without authorization, you cannot perform any asynchronous activity against the data set represented by the DEB (such as EXCP, CLOSE, EOV), because CVAF removes the DEB from the DEB table for the duration of the CVAF call. See "Identifying the Volume" on page 56 for further details.

If you supply a previously obtained I/O area through the IOAREA keyword, neither UCB nor DEB is needed. Otherwise, supply either a UCB or DEB. If you supply a UCB address, it is overlaid in the CVPL by the UCB address in the I/O area. If you supply both the UCB and the DEB addresses in the CVPL, the DEB address is used and the UCB address in the CVPL is overlaid by the UCB address in the DEB.

FCL: Specify a Filter Criteria List

FCL=*addr or (reg)*
Supplies the address of a filter criteria list. It is required when ACCESS=READ is specified on the standard form of the macro. The format of the two elements of the filter criteria list is shown in Table 20 on page 67 and Table 21 on page 68.

FLTAREA: Keep or Free the Filter Save Area

FLTAREA=KEEP
Specifies keeping the filter save area. Code this operand if the RESUME function might be called later (to resume processing prematurely terminated because the number of caller-supplied buffers is not enough to contain all the returned DSCBs).

CVAFFILT returns the address of the kept filter save area in the CVAFFILT parameter list (CVFSA field). If you specify the same parameter list in subsequent RESUME calls, CVAFFILT reuses the same filter save area.

Tip: If you code this operand, you must subsequently issue CVAFFILT with ACCESS=RLSE to release the filter save area.

FLTAREA=(KEEP,*addr or (reg)*)
Supplies the address of a previously obtained filter save area. See the description of FLTAREA=KEEP operand for additional concerns.

FLTAREA=NOKEEP
Frees the filter save area upon completion of the CVAF request.

FLTAREA=(NOKEEP,*addr or (reg)*)
Frees a previously obtained filter save area upon completion of the CVAF request.

IOAREA: Keep or Free the I/O Work Area

IOAREA=KEEP
Specifies keeping the CVAF I/O work area. For authorized callers, CVAFFILT returns the address of the kept I/O work area in the CVAFFILT parameter list (CVIOAR). If you specify the same parameter list in subsequent calls, CVAFFILT reuses the same I/O work area.

Tip: If you code this operand, you must subsequently issue CVAFFILT with ACCESS=RLSE to release the I/O work area.

IOAREA=(KEEP,*addr or (reg)*)
Supplies the address of a previously obtained filter save area. See the description of IOAREA=KEEP operand for additional concerns.

IOAREA=NOKEEP
Frees the filter save area upon completion of the CVAF request.

IOAREA=(NOKEEP,*addr or (reg)*)
Frees a previously obtained CVAF I/O work area upon completion of the CVAF request.

BRANCH: Specify the Entry to the Macro

BRANCH=NO
Requests the SVC (default) entry. Protect key checking is performed.

BRANCH=YES
Equivalent to BRANCH=(YES,SUP), because SUP is the default when you code YES. You must be in supervisor state. Protect key checking is bypassed.

BRANCH=(YES,SUP)
Requests the branch entry. You must be in supervisor state. Protect key checking is bypassed. If you specify BRANCH=YES, supply an 18-word save area. You cannot hold a lock at entry to CVAF. You cannot be in SRB mode.

BRANCH=(YES,PGM)
Requests the branch entry. You must be APF authorized and be in problem state. Protect key checking is bypassed.

EADSCB=value: Specify the support level for extended attribute DSCBs.

EADSCB=OK
This specification indicates that the calling program supports extended attribute DSCBs. An extended address volume may have these DSCBs allocated to it. The returned DSCBs (format-3, format-8) may contain extent descriptors described by 28-bit cylinder addresses or DSCBs (format-9) that contain additional attribute information.

For fully qualified data set names in the Filter Criteria List, a CVAFFILT request fails if the EADSCB=OK, indicator is not set and the DSCB associated with the fully qualified data set name is a format-8 DSCB.

For partially qualified data set names in the Filter Criteria List, a CVAFFILT request fails if the EADSCB=OK, indicator is not set and a DSCB associated with a data set that matches the Filter Criteria List is a format-8 DSCB.

The failing error code for these cases will be reflected as follows:

- Data set name status in the FCL (FCLDSNST) is set to a status value of (X'06'). This status code indicates that a data set name is described by a format-8 DSCB and the caller did not specify support for an EAV with the EADSCB=OK keyword.
- Set the no resume CVAF status code (CVSTAT) of STAT072
- Return code 4.

EADSCB=OK will set the CV4EADOK indicator in the CVPL.

EADSCB=NOTOK
Indicates a calling program does not support extended attribute DSCBs. EADSCB=NOTOK will set the CV4EADOK indicator in the CVPL to off. This is the default.

MF: Specify the Form of the Macro

Specifies whether the DSECT, list, execute, or normal form of the macro is requested. You can be in either 24-bit or 31-bit addressing mode. If you are not authorized, you must pass the address of a DEB built by OPEN. If you are authorized, you can pass either the DEB address or the UCB address. You must ensure that the volume is allocated and will remain mounted (for example, by dynamic allocation).

MF=I
Specifies the standard form of the macro. The CVAF parameter list is generated and CVAF is called. The default is MF=I.

MF=D
Specifies the DSECT form of the macro. The macro generates a request for the ICVAFPL macro to map the unique CVAF filter CVPL (4-bytes longer than standard CVPL).

MF=L
Specifies the list form of the macro. The CVAF parameter list is generated, but CVAF is not called.

MF=(E,*addr or (reg)*)
Specifies the execute form of the macro. The CVAF parameter list whose address is in *addr* or *reg* is used. You can modify the parameter list with this form of the macro.

Return Codes from CVAFFILT

CVAF filter service does not issue any messages. Upon return from CVAF, register 1 contains the address of the CVAF parameter list and register 15 contains one of the following return codes:

Return Code	Meaning
0 (X'00')	The request was successful.
4 (X'04')	Logical error; status information in CVSTAT.
8 (X'08')	Invalid VTOC structure.
12 (X'0C')	CVAFFILT parameter list in wrong key, or not valid.
16 (X'10')	I/O error.

CVSTAT in the CVAF parameter list contains the status code. See “Codes Put in the CVSTAT Field” on page 128 for a list of the status codes.

Partially-Qualified Names for CVAFFILT

CVAFFILT supports partially-qualified data set names using single or double asterisk notation and the percent sign as shown in the following text:

- You can use a single asterisk to represent a single qualifier. For example, SYS1.*.LOAD designates any data set with three qualifiers, the first being SYS1, the second being any qualifier, and the third being LOAD.
- You can also use a single asterisk to represent zero or more unspecified characters. For example, LOAD.*LIB designates any data set having only two qualifiers, with LOAD being the first, and the second qualifier ending with the character string LIB (for example, LINKLIB). The asterisk can appear anywhere within the qualifier. You can use two single asterisks in the following way: LOAD.A*B*.LIB. CVAFFILT does not support the use of two or more single asterisks with any other character within a single qualifier (for example, LOAD.B**.LIB is not valid).
- A double asterisk represents a place holder for zero or more qualifiers. For example, SYS1.** designates any data set having SYS1 as its first or only qualifier.
- You can specify a percent sign (%) as part of a partially-qualified name. The percent sign designates any data set whose name matches the partially-qualified name, except for the single character in the position indicated by the percent sign. For example, SYS%.*.LOAD% designates any data set with three qualifiers, the first being any 4-character qualifier beginning with SYS, the second being any qualifier, and the third being any five character qualifier beginning with LOAD.

Example of Using the CVAFFILT Macro

This example uses the CVAFFILT macro to read all format-1 and format-3 DSCBs for specific data sets within a given VTOC as well as for all data sets within a given VTOC. It will also calculate the number of DSCBs and print the totals and appropriate messages to an output file. Refer to the documentation within the sample source in “CVAFFILT Macro Overview and Specification” on page 94 for setup requirements, program logic, and expected output.

The CVAF parameter list, buffer list, and filter criteria list are defined in the sample source. The ICVAFPL macro generates the CVAF parameter list, the ICVAFBFL macro generates the buffer list, and the ICVFCL macro generates the filter criteria list.

Sample JCL for the CVAFFILT macro

The following is the sample JCL used to Assemble, Link, and Execute the example source. Changes will have to be made to this JCL as appropriate for each customer environment.

```
//CVAFFEXP JOB ,MSGCLASS=X,TIME=(,10),
//             NOTIFY=&SYSUID
//*
//STEP01  EXEC PROC=ASMACLG
//SYSIN     DD *
    (INCLUDE EXAMPLE SOURCE HERE)
/*
//*
//L.SYSLMOD DD DSN=YOUR.AUTH.LINKLIB(CVAFFEXP),DISP=SHR
//L.SYSIN   DD *
  SETCODE AC(1)
  ENTRY   CVAFFEXP
/*
//*
//G.SYSABEND DD  SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//*
//G.CVAFDD  DD DISP=SHR,UNIT=3390,VOL=SER=339L62
//G.OUTDD   DD DSN=CVAFFLT1.OUTPUT,
//             DISP=(NEW,CATLG),
//             UNIT=3390,VOL=SER=339L61,
//             SPACE=(TRK,(2,2)),
//             DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//*
//
```

Code example of the CVAFFILT Macro

```
CVAFFEXP TITLE 'CVAF CVAFFILT EXAMPLE'
CVAFFEXP CSECT
CVAFFEXP AMODE 24
CVAFFEXP RMODE 24
*
*********************************************************************
*                                                                   *
*    CVAFFEXP - CVAFFILT EXAMPLE                                    *
*                                                                   *
*                                                                   *
*    CVAFFILT EXAMPLE TO BE USED IN DFSMS ADVANCED SERVICES MANUAL  *
*                                                                   *
*    CVAFFILT MACRO RUN IN AMODE24/RMODE24                          *
*                                                                   *
*    CVAFFILT TEST DSN'S / PROCESSING USED:                         *
*                                                                   *
*    1) 2 SEQUENTIAL DATASETS WITH 5 EXTENTS. (1 FMT1/1 FMT3 EACH)  *
*       DSCBS RETURNED WILL BE COMBINED TO TOTAL 2 FMT1 / 2 FMT3'S  *
*       DSN: CVAFFLT1.DATA01                                        *
*       DSN: CVAFFLT1.DATA02                                        *
*       CREATE ON THE VOLUME ASSOCIATED WITH THE CVAFDD DD          *
*                                                                   *
*    2) PDSE DATASET WITH 122 EXTENTS. (1 FMT1/10 FMT3'S)           *
*       DSN: CVAFFLT1.PDSE01                                        *
*       CREATE ON THE VOLUME ASSOCIATED WITH THE CVAFDD DD          *
*                                                                   *
*    3) 1 SEQUENTIAL DATASET WITH 5 EXTENTS (1 FMT1/1 FMT3)         *
*       DSN: CVAFFLT1.DATA01                                        *
*       CREATE ON THE VOLUME ASSOCIATED WITH THE CVAFDD DD          *
*                                                                   *
*    4) RETURN DSCB COUNT FOR ENTIRE VOLUME USING CVAFFILT RESUME   *
*       PROCESING.                                                  *
*       DSN'S ON THE VOLUME: (IN SEQUENCE ORDER)                    *
*       SYS1.VTOCIX.V39L62   (VTOC INDEX - 1 FMT1)                  *
*       CVAFFLT1.DATA01      (5 EXTENTS - 1 FMT1/1 FMT3)            *
*       CVAFFLT1.DATA02      (5 EXTENTS - 1 FMT1/1 FMT3)            *
*       CVAFFLT1.PDSE01      (122 EXTENTS - 1 FMT1/10 FMT3'S)       *
*       DSN'S CREATED ON THE VOLUME ASSOCIATED WITH THE CVAFDD DD   *
*                                                                   *
*                                                                   *
*       OUTPUT IN OUTDD DATASET SHOULD BE THE FOLLOWING:            *
*-------------------------------------------------------------------*
*                                                                   *
*                                                                   *
*   CVAFFEXP START OF OUTPUT MESSAGES                               *
```

```
*                                                                     *
*   RC00 VERIFIED - THE REQUEST WAS SUCCESSFUL                        *
*   CVAFFILT RETURNED THE FOLLOWING DSCBS FOR DSN: CVAFFLT1.DATA01    *
*                                    AND FOR DSN: CVAFFLT1.DATA02     *
*    NUMBER OF FORMAT 1 DSCBS - 0000002                               *
*    NUMBER OF FORMAT 3 DSCBS - 0000002                               *
*                                                                     *
*   RC00 VERIFIED - THE REQUEST WAS SUCCESSFUL                        *
*   CVAFFILT RETURNED THE FOLLOWING DSCBS FOR DSN: CVAFFLT1.PDSE01    *
*    NUMBER OF FORMAT 1 DSCBS - 0000001                               *
*    NUMBER OF FORMAT 3 DSCBS - 0000010                               *
*                                                                     *
*   RC00 VERIFIED - THE REQUEST WAS SUCCESSFUL                        *
*   CVAFFILT RETURNED THE FOLLOWING DSCBS FOR DSN: CVAFFLT1.DATA01    *
*    NUMBER OF FORMAT 1 DSCBS - 0000001                               *
*    NUMBER OF FORMAT 3 DSCBS - 0000001                               *
*                                                                     *
*   RC04 VERIFIED - CVSTAT 064  RESUME IS NECESSARY                   *
*   CVAFFILT (INITIAL) RETURNED THE FOLLOWING DSCBS FOR THE VOLUME:   *
*    NUMBER OF FORMAT 1 DSCBS - 0000003                               *
*    NUMBER OF FORMAT 3 DSCBS - 0000002                               *
*   CVAFFILT (RESUME)  RETURNED THE FOLLOWING DSCBS FOR THE VOLUME:   *
*    NUMBER OF FORMAT 1 DSCBS - 0000001                               *
*    NUMBER OF FORMAT 3 DSCBS - 0000010                               *
*   CVAFFILT RESUME OPERATION COMPLETE - ALL DSCBS RETURNED           *
*                                                                     *
*   CVAFFEXP END OF OUTPUT MESSAGES                                   *
*                                                                     *
*                                                                     *
*---------------------------------------------------------------------*
*                                                                     *
* NOTE: THE NUMBER OF DSCBS RETURNED WILL VARY IF THE DATASET DOES    *
*       NOT EXIST ON THE VOLUME OF IF THE DATASET EXTENT IS NOT AS    *
*       LISTED ABOVE OR IF THERE IS NO VTOC INDEX ON THE VOLUME.      *
*                                                                     *
* NOTE: WHEN CREATING A PDSE ON A SMS MANAGED VOLUME A VVDS           *
*       WILL ALSO BE CREATED AND THE FMT1 COUNT WILL BE               *
*       INCREASED BY ONE.                                             *
*                                                                     *
*                                                                     *
***********************************************************************
*
***********************************************************************
*                                                                     *
*    CVAFFEXP - LOGIC NOTES                                           *
*                                                                     *
*    THIS EXAMPLE WILL PERFORM THE FOLLOWING:                         *
*                                                                     *
*    INITIALIZATION                                                   *
*    - OBTAIN THE NECESSARY INFORMATION FROM THE DASD VOLUME          *
*    - OPEN AN OUTPUT FILE AND WRITE THE NECESSARY OUTPUT MESSAGES    *
*    - INITIALIZE THE NECESSARY BUFFER LIST FOR CVAFFILT              *
*                                                                     *
*    MAINLINE                                                         *
*    - INITIALIZE A FCL TO READ FOR TWO SPECIFIC SEQ DATASETS         *
*    - ISSUE CVAFFILT READ TO READ THE DSCBS FOR THE TWO DATASETS     *
*    - CHECK THE RETURN CODE AND CVSTAT CODE FROM CVAFFILT            *
*    - COUNT THE NUMBER OF FMT1 AND FMT3 DSCBS FOR THE REQUEST        *
*    - FORMAT THE DSCB COUNTS AND WRITE TO THE OUTPUT DATASET         *
*    - ISSUE CVAFFILT RLSE TO RELEASE THE WORK AREAS USED             *
*                                                                     *
*    - INITIALIZE A FCL TO READ FOR ONE SPECIFIC PDSE DATASET         *
*    - ISSUE CVAFFILT READ TO READ THE DSCBS FOR THE PDSE DATASET     *
*    - CHECK THE RETURN CODE AND CVSTAT CODE FROM CVAFFILT            *
*    - COUNT THE NUMBER OF FMT1 AND FMT3 DSCBS FOR THE REQUEST        *
*    - FORMAT THE DSCB COUNTS AND WRITE TO THE OUTPUT DATASET         *
*    - ISSUE CVAFFILT RLSE TO RELEASE THE WORK AREAS USED             *
*                                                                     *
*    - INITIALIZE A FCL TO READ FOR ONE SPECIFIC SEQ DATASET          *
*    - ISSUE CVAFFILT READ TO READ THE DSCBS FOR THE ONE DATASET      *
*    - CHECK THE RETURN CODE AND CVSTAT CODE FROM CVAFFILT            *
*    - COUNT THE NUMBER OF FMT1 AND FMT3 DSCBS FOR THE REQUEST        *
*    - FORMAT THE DSCB COUNTS AND WRITE TO THE OUTPUT DATASET         *
*    - ISSUE CVAFFILT RLSE TO RELEASE THE WORK AREAS USED             *
*                                                                     *
*    - INITIALIZE A FCL TO READ THE DSCBS ON THE ENTIRE VOLUME        *
*    - ISSUE CVAFFILT READ TO READ THE DSCBS FOR THE VOLUME           *
*    - CHECK THE RETURN CODE AND CVSTAT CODE FROM CVAFFILT            *
*    - PROCESS ALL DSCBS USING CVAFFILT RESUME AS NECESSARY           *
*    - COUNT THE NUMBER OF FMT1 AND FMT3 DSCBS FOR THE REQUEST        *
*    - FORMAT THE DSCB COUNTS AND WRITE TO THE OUTPUT DATASET         *
*    - ISSUE CVAFFILT RLSE TO RELEASE THE WORK AREAS USED             *
```

```
*                                                                     *
*    FINALIZATION                                                     *
*    - CLOSE THE OUTPUT FILE                                          *
*    - EXIT                                                           *
*                                                                     *
*                                                                     *
*    CVAFFEXP - JOB INFORMATION                                       *
*                                                                     *
*    NORMAL END OF JOB:                                               *
*    - RC=00 AND OUTDD OUTPUT AS DETAILED ABOVE                       *
*                                                                     *
*                                                                     *
*    ABNORMAL END OF JOB:                                             *
*    - ABEND 100 - ERROR OPENING VTOC ON THE DASD VOLUME THAT IS      *
*                  ASSOCIATED WITH THE CVAFDD DD STATEMENT            *
*    - ABEND 101 - ERROR OPENING THE OUTDD DATASET                    *
*    - ABEND 102 - ERROR CLOSING THE OUTDD DATASET                    *
*                                                                     *
*                                                                     *
*    DASD VOLUMES USED IN THIS EXAMPLE:                               *
*    - 339L61 - 3390 WHERE OUTDD DATASET IS DEFINED                   *
*    - 339L62 - 3390 WHERE TEST DATASETS DETAILED ABOVE ARE DEFINED   *
*                                                                     *
*                                                                     *
***********************************************************************
*
***********************************************************************
*                                                                     *
* HOUSEKEEPING                                                        *
* - SAVE CALLER'S REGISTERS AND ESTABLISH A NEW REGISTER SAVE AREA    *
*                                                                     *
***********************************************************************
*
         STM   R14,R12,12(R13)    STANDARD LINKAGE CONVENTION
         BALR  R11,0              DCL R11 AS IMPLIED BASE REG
         USING BASE,R11,R12       R12 IS ALSO BASE REG
BASE     L     R12,BASEADDR       SET UP ADDRESSING FOR R12
         B     CVAFFL00           BRANCH AROUND DECLARES
BASEADDR DC    A(BASE+4096)       ADDRESSING FOR R12
CVAFFL00 DS    0H                 CONTINUE...
         ST    R13,SAVE+4         SAVE PTR TO CALLER'S SAVE AREA
         LA    R14,SAVE           GET ADDRESS OF THE NEW SAVE AREA
         ST    R14,8(,R13)        CHAIN CALLER'S AREA TO OURS
         LR    R13,R14            ESTABLISH THE NEW SAVE AREA
*
***********************************************************************
*                                                                     *
* INITIALIZATION                                                      *
*                                                                     *
***********************************************************************
*
INITIAL  DS    0H                 INITIALIZATION SECTION
         BAL   R14,IDVOLRTN       INVOKE RTN TO IDENTIFY THE VOLUME(S)
         BAL   R14,OPENRTN        INVOKE OPEN OUTPUT DATASET RTN
         MVC   PDETLINE(133),STRTMSG  MOVE START MSG TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         MVC   PDETLINE(133),BLNKLNE  MOVE BLANK LINE TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         BAL   R14,BUFLRTN        INVOKE RTN TO INIT BUFFER LIST (H/E)
*
***********************************************************************
*                                                                     *
* MAINLINE                                                            *
*                                                                     *
***********************************************************************
*
MAINLINE DS    0H                 MAINLINE SECTION
         MVI   DSNNBR,X'02'       SET DSNNBR FLAG TO TWO
         BAL   R14,FCL1RTN        INVOKE RTN TO INIT FCL (DS01/DS02)
         BAL   R14,CVAFRD2        INVOKE CVAFFILT READ RTN 2
         BAL   R14,TSTRCRTN       INVOKE TEST RC RTN
         L     R15,RETCODE        LOAD R15 WITH RETURN CODE
         CH    R15,RCODE00        IS RC = 0
         BNE   MAIN0010            NO - DO NOT FORMAT/PRINT LINES
         MVC   DS(44),DS01        ELSE MOVE DSN FOR DS01 TO MSG LINE
         MVC   PDETLINE(133),DSCBMSGA  MOVE MSG TO LINE
         PUT   OUTFILE,PDETLINE      WRITE A RECORD TO THE OUTPUT FILE
         MVC   DS2(44),DS02          MOVE DSN FOR DS02 TO MSG LINE
         MVC   PDETLINE(133),DSCBMSGB MOVE MSG TO LINE
         PUT   OUTFILE,PDETLINE      WRITE A RECORD TO THE OUTPUT FILE
         BAL   R14,FMTOPRTN          INVOKE FORMAT OUTPUT MSG RTN
         B     MAIN0020              BRANCH TO MAIN0020
```

```
MAIN0010 DS    0H                 CVAFFILT - CVSTAT LOGICAL ERROR
         MVC   DS3(44),DS01       MOVE DSN FOR DS01 TO MSG LINE
         MVC   PDETLINE(133),DSCBMSGC MOVE MSG TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         MVC   DS3(44),DS02       MOVE DSN FOR DS02 TO MSG LINE
         MVC   PDETLINE(133),DSCBMSGC MOVE MSG TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
MAIN0020 DS    0H
         MVC   PDETLINE(133),BLNKLNE  MOVE BLANK LINE TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         BAL   R14,CVAFRL         INVOKE CVAFFILT RELEASE RTN
         MVI   DSNNBR,X'01'       SET DSNNBR FLAG TO ONE
         BAL   R14,FCL2RTN        INVOKE RTN TO INIT FCL (DS03)
         BAL   R14,CVAFRD1        INVOKE CVAFFILT READ RTN 1
         BAL   R14,TSTRCRTN       INVOKE TEST RC RTN
         L     R15,RETCODE        LOAD R15 WITH RETURN CODE
         CH    R15,RCODE00        IS RC = 0
         BNE   MAIN0030            NO - DO NOT FORMAT/PRINT LINES
         MVC   DS(44),DS03        ELSE MOVE DSN FOR DS03 TO MSG LINE
         MVC   PDETLINE(133),DSCBMSGA  MOVE MSG TO LINE
         PUT   OUTFILE,PDETLINE      WRITE A RECORD TO THE OUTPUT FILE
         BAL   R14,FMTOPRTN          INVOKE FORMAT OUTPUT MSG RTN
         B     MAIN0040              BRANCH TO MAIN0040
MAIN0030 DS    0H                 CVAFFILT - CVSTAT LOGICAL ERROR
         MVC   DS3(44),DS03       MOVE DSN FOR DS03 TO MSG LINE
         MVC   PDETLINE(133),DSCBMSGC MOVE MSG TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
MAIN0040 DS    0H
         MVC   PDETLINE(133),BLNKLNE  MOVE BLANK LINE TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         BAL   R14,CVAFRL         INVOKE CVAFFILT RELEASE RTN
         BAL   R14,FCL3RTN        INVOKE RTN TO INIT FCL (DS01)
         BAL   R14,CVAFRD1        INVOKE CVAFFILT READ RTN
         BAL   R14,TSTRCRTN       INVOKE TEST RC RTN
         L     R15,RETCODE        LOAD R15 WITH RETURN CODE
         CH    R15,RCODE00        IS RC = 0
         BNE   MAIN0050            NO - DO NOT FORMAT/PRINT LINES
         MVC   DS(44),DS01        ELSE MOVE DSN FOR DS01 TO MSG LINE
         MVC   PDETLINE(133),DSCBMSGA  MOVE MSG TO LINE
         PUT   OUTFILE,PDETLINE      WRITE A RECORD TO THE OUTPUT FILE
         BAL   R14,FMTOPRTN          INVOKE FORMAT OUTPUT MSG RTN
         B     MAIN0060              BRANCH TO MAIN0060
MAIN0050 DS    0H                 CVAFFILT - CVSTAT LOGICAL ERROR
         MVC   DS3(44),DS01       MOVE DSN FOR DS01 TO MSG LINE
         MVC   PDETLINE(133),DSCBMSGC MOVE MSG TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
MAIN0060 DS    0H
         MVC   PDETLINE(133),BLNKLNE  MOVE BLANK LINE TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         BAL   R14,CVAFRL         INVOKE CVAFFILT RELEASE RTN
         BAL   R14,FCL4RTN        INVOKE RTN TO INIT FCL (ENTIRE VOL)
         BAL   R14,CVAFRDA        INVOKE CVAFFILT READ RTN (FOR RESUME)
         LA    R9,CVPLDEFA        ESTABLISH ADDRESSABILITY
         USING CVPLMAP,R9         TO THE CVPL (FOR CVSTAT)
         BAL   R14,TSTRCRTN       INVOKE TEST RC RTN
*
*********************************************************************
*                                                                   *
* FINALIZATION                                                      *
*                                                                   *
*********************************************************************
*
FINAL    DS    0H                 FINALIZATION SECTION
         MVC   PDETLINE(133),BLNKLNE  MOVE BLANK LINE TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         MVC   PDETLINE(133),ENDMSG  MOVE END MSG TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         BAL   R14,CLOSERTN       INVOKE CLOSE OUTPUT DATASET RTN
         L     R13,4(R13)         RESTORE REGISTER
         LM    R14,R12,12(R13)    RESTORE CALLERS REGISTERS
         LA    R15,0              SET RC TO 0
         BR    R14                RETURN TO CALLER
*
*********************************************************************
*                        OPENRTN                                    *
*          - ROUTINE TO OPEN OUTPUT FILE USED BY THIS MODULE        *
*********************************************************************
*
OPENRTN  DS    0H                 OPEN FILES ROUTINE
         ST    R14,OPENSAVE       STORE C(R14) INTO SAVE AREA
         OPEN  (OUTFILE,(OUTPUT)) OPEN THE OUTDD OUTPUT FILE FOR MSGS
         TM    OUTFILE+(DCBOFLGS-IHADCB),DCBOFOPN  IS FILE OPEN?
```

```
         BO    OPENEXIT           FILE OPEN OK - EXIT OPEN RTN
         LA    R1,EABN101         OUTPUT FILE NOT OPEN-USER ABEND 101
         BAL   R14,ABENDRTN       INVOKE ABEND ROUTINE
OPENEXIT DS    0H                 EXIT FROM OPEN ROUTINE
         L     R14,OPENSAVE       LOAD C(OPENSAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                          CLOSERTN                                 *
*          - ROUTINE TO CLOSE OUTPUT FILE USED BY THIS MODULE       *
*********************************************************************
*
CLOSERTN DS    0H                 CLOSE FILES ROUTINE
         ST    R14,CLOSSAVE       STORE C(R14) INTO SAVE AREA
         CLOSE (OUTFILE)          CLOSE OUTPUT FILE
         LTR   R15,R15            CHECK IF CLOSED OK
         BZ    CLOSEXIT           IF OK BRANCH TO CLOSEXIT
         LA    R1,EABN102         ELSE SETUP FOR USER ABEND 102
         BAL   R14,ABENDRTN       INVOKE ABEND ROUTINE
CLOSEXIT DS    0H                 EXIT FROM CLOSE ROUTINE
         L     R14,CLOSSAVE       LOAD C(CLOSSAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                          ABENDRTN                                 *
*                    - FORCE AN ABEND ROUTINE                       *
*********************************************************************
*
ABENDRTN DS    0H                 ABEND ROUTINE
         ST    R14,ABENSAVE       STORE C(R14) INTO SAVE AREA
         ABEND (R1),DUMP          ISSUE USER ABEND WITH DUMP
ABENEXIT DS    0H                 EXIT FROM ABEND ROUTINE
         L     R14,ABENSAVE       LOAD C(ABENSAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                         IDVOLRTN                                  *
*     - OBTAIN THE NECESSARY INFORMATION FROM THE DASD VOLUME       *
*********************************************************************
*
IDVOLRTN DS    0H                 IDENTIFY VOLUME ROUTINE
         ST    R14,IDVLSAVE       STORE C(R14) INTO SAVE AREA
         RDJFCB (VTOCDCB,(INPUT)) READ JFCB / OPEN VTOC
         MVI   JFCB1,X'04'        PUT IN ID FOR FORMAT 4
         MVC   JFCB1+1(43),JFCB1  SETUP FOR VTOC OPEN
         OPEN  (VTOCDCB,(INPUT)),TYPE=J  OPEN VTOC (OPEN TYPE=J)
         TM    VTOCDCB+(DCBOFLGS-IHADCB),DCBOFOPN
         BO    IDVOL010           BRANCH TO IDVOL010 - GOOD OPEN
         LA    R1,EABN100         ELSE SETUP FOR USER ABEND 100
         BAL   R14,ABENDRTN       INVOKE ABEND ROUTINE
IDVOL010 DS    0H                 GOOD OPEN - OBTAIN VOLUME INFORMATION
         SLR   RDEB,RDEB          INIT REG1 FOR DEB PTR
         SLR   RUCB,RUCB          INIT REG2 FOR UCB PTR
         ICM   RDEB,B'0111',VTOCDCB+(DCBDEBA-IHADCB) GET DEB ADDRESS
         ST    RDEB,DEBADD        SAVE DEB ADDRESS INTO R1
         ICM   RUCB,B'0111',(DEBBASND-DEBBASIC)+(DEBUCBA-DEBDASD)(RDEB)
         ST    RUCB,UCBADD        SAVE UCB ADDRESS INTO R2
IDVLEXIT DS    0H                 EXIT FROM IDVOLRTN
         L     R14,IDVLSAVE       LOAD C(IDVLSAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                         TSTRCRTN                                  *
*             - TEST RETURN CODE FROM CVAFFILT                      *
*             - FORMAT AND PRINT MESSAGES AS NEEDED                 *
*********************************************************************
*
TSTRCRTN DS    0H                 CHECK RETURN CODE ROUTINE
         ST    R14,TSTRSAVE       STORE C(R14) INTO SAVE AREA
         L     R15,RETCODE        LOAD R15 WITH RC SAVED FROM LAST CALL
         CH    R15,RCODE00        IF RETURN CODE = 00
         BE    TSTRC00               BRANCH TO PROCESS RC00
         CH    R15,RCODE04        IF RETURN CODE = 04
         BE    TSTRC04               BRANCH TO PROCESS RC04
TSTRCER  DS    0H                 ELSE PRINT GENERAL ERROR MESSAGE
         MVC   PDETLINE(133),RCERMSG MOVE ERROR MSG TO PRINT LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         B     TSTREXIT           BRANCH TO EXIT RTN
TSTRC00  DS    0H                 PROCESS RETURN CODE = 00
         MVC   PDETLINE(133),RC00MSG MOVE RC00 MSG TO PRINT LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         BAL   R14,FMTCTRTN       BRANCH TO FMT COUNT RTN
```

```
         BAL   R14,FMTOPRTN       INVOKE FORMAT OUTPUT MSG ROUTINE
         B     TSTREXIT           BRANCH TO EXIT RTN
TSTRC04  DS    0H                 PROCESS RETURN CODE = 04
         CLI   CVSTAT,STAT064     DO WE NEED RESUME?
         BNE   TSTR0010           NO - PRINT LOGICAL ERROR MSG
         BAL   R14,CVAFRS         INVOKE CVAFRS RESUME ROUTINE
         B     TSTREXIT           BRANCH TO EXIT RTN
TSTR0010 DS    0H                 PRINT RC04-OTHER LOGICAL ERROR/CVSTAT
         MVC   PDETLINE(133),RC04MSG MOVE RC04 MSG TO PRINT LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
TSTREXIT DS    0H                 EXIT FROM TSTRCRTN
         L     R14,TSTRSAVE       LOAD C(TSTRSAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                          FMTCTRTN                                 *
*      - COUNT THE NUMBER OF FMT1 AND FMT3 DSCBS FOR THE REQUEST    *
*********************************************************************
*
FMTCTRTN DS    0H                 COUNT DSCBS RETURNED ROUTINE
         ST    R14,FMTCSAVE       STORE C(R14) INTO SAVE AREA
         SLR   R6,R6              ZERO OUT R6
         SLR   R7,R7              ZERO OUT R7
         CLI   DSNNBR,X'01'       IF DSNNBR FLAG = 1
         BE    FMTC0010              BRANCH TO LOAD ADDR OF FCL
         LA    R4,FCLDEF2         ELSE LOAD R4 WITH ADDR OF FCL2
         B     FMTC0020                BRANCH TO SET USING
FMTC0010 DS    0H
         LA    R4,FCLDEF          LOAD R4 WITH ADDR OF FCL
FMTC0020 DS    0H                 CONTINUE - SET USING
         USING FCLMAP,R4          ESTABLISH ADDRESSABILITY TO FCL
         SLR   R5,R5              ZERO OUT R5
         ICM   R5,B'0011',FCLDSCBR  DETERMINE IF ANY DSCBS RETURNED
         BZ    FMTC0060             NO - GO AND PRINT APPROPRIATE MSG
         LA    R4,DSCBDEF         LOAD R4 WITH ADDR OF DSCB MAP
         USING DSCBMAP,R4         ESTABLISH ADDRESSABILITY TO DSCB
FMTC0030 DS    0H                 COUNT DSCBS BY TYPE RETURNED
         CLI DS1FMTID,X'F1'       IF FORMAT1
         BE    FMTC0040            BRANCH TO FMTC0040
         CLI DS1FMTID,X'F3'       ELSE IF NOT FORMAT3
         BNE   FMTC0050                 BRANCH TO FMTC0050
         LA    R7,1(R7)           ADD 1 TO FORMAT3 COUNT
         B     FMTC0050           BRANCH TO FMTC0050
FMTC0040 DS    0H                 FMTC0040 - FORMAT1 INCREMENT
         LA    R6,1(R6)           ADD 1 TO FORMAT1 COUNT
FMTC0050 DS    0H                 FMTC0050 - PROCESS THROUGH DSCBS
         LA    R4,DSCBSIZ(R4)     ADD DSCBSIZ TO R4
         BCT   R5,FMTC0030        SUBTRACT 1 FROM DSCB COUNT AND CONT
         ST    R6,RETF1           STORE #FMT1'S INTO RETF1
         ST    R7,RETF3           STORE #FMT3'S INTO RETF3
         DROP  R4                 DROP R4 USING
         B     FMTCEXIT           BRANCH AROUND FMTC0060
FMTC0060 DS    0H                 PRINT MSG - NO DSCB'S RETURNED
         MVC   PDETLINE(133),NODSCBM   MOVE MSG TO PRINT LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         SLR   R6,R6              ZERO OUT R6
         ST    R6,RETF1           STORE ZERO INTO RETF1
         SLR   R7,R7              ZERO OUT R7
         ST    R7,RETF3           STORE ZERO INTO RETF3
FMTCEXIT DS    0H                 EXIT FROM FMTCTRTN
         L     R14,FMTCSAVE       LOAD C(FMTCSAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                          FMTOPRTN                                 *
*     - FORMAT THE DSCB COUNTS AND WRITE TO THE OUTPUT DATASET      *
*********************************************************************
*
FMTOPRTN DS    0H                 FORMAT OUTPUT ROUTINE
         ST    R14,FMTOSAVE       STORE C(R14) INTO SAVE AREA
         MVC   MSG(29),DSCBMSG1   MOVE MSG TO FORMAT LINE
         L     R6,RETF1           LOAD R6 WITH NBR OF FMT1'S RETURNED
         CVD   R6,WF1             CONVERT TO DEC FOR PRINTING
         UNPK  WFMTREC+29(7),WF1  UNPACK TO FORMAT LINE
         OI    WFMTREC+35,X'F0'   SET APPROPRIATE BITS
         MVC   PDETLINE(133),WFMTREC   MOVE RECORD TO OUTPUT LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         MVC   MSG(29),DSCBMSG2   MOVE MSG TO FORMAT LINE
         L     R7,RETF3           LOAD R7 WITH NBR OF FMT3'S RETURNED
         CVD   R7,WF3             CONVERT TO DEC FOR PRINTING
         UNPK  WFMTREC+29(7),WF3  UNPACK TO FORMAT LINE
         OI    WFMTREC+35,X'F0'   SET APPROPRIATE BITS
```

```
         MVC   PDETLINE(133),WFMTREC   MOVE RECORD TO OUTPUT LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
FMTOEXIT DS    0H               EXIT FROM FMTOPRTN
         L     R14,FMTOSAVE       LOAD C(FMTOSAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                       BUFLRTN                                     *
*             - INITIALIZE BUFFER LIST HEADER (BFLH)                *
*             - INITIALIZE BUFFER LIST ELEMENTS (BFLE)              *
*********************************************************************
*
BUFLRTN  DS    0H                 BUFFER LIST INITIALIZATION ROUTINE
         ST    R14,BUFLSAVE       STORE C(R14) INTO SAVE AREA
         XC    BFLHDEF(BFLSIZE),BFLHDEF   CLEAR BUFR LIST AREA
         LA    R1,BFLHDEF         R1 - BUFFER LIST HEADER
         USING BFLMAP,R1          ESTABLISH ADDRESSABILITY
         MVI   BFLHNOE,BUFNBR     SET NUMBER OF BUFFER ELEMENTS
         OI    BFLHFL,BFLHDSCB    IDENTIFY AS DSCB BUFR ELEMENT LIST
         LA    R2,BFLHDEF+BFLHLN  R2 - FIRST BUFFER LIST ELEMENT
         USING BFLE,R2            ESTABLISH ADDRESSABILITY
         LA    R3,DSCBDEF         R3 - FIRST DSCB BUFFER
         LA    R4,BUFNBR          R4 = NUMBER OF ELEMENTS AND BUFRS
BFL0010  OI    BFLEFL,BFLECHR     REQUEST CCHHR ON RETURN
         MVI   BFLELTH,DSCBSIZ    SET BUFR LNGTH TO FULL DSCB SIZE
         ST    R3,BFLEBUF         SET ADDR(DSCB BUFFER)
         LA    R2,BFLELN(R2)      R2 - NEXT BUFFER LIST ELEMENT
         LA    R3,DSCBSIZ(R3)     R3 - NEXT DSCB BUFFER
         BCT   R4,BFL0010         LOOP THROUGH ALL ELEMENTS
         DROP  R1,R2              DROP TEMP USINGS
BUFLEXIT DS    0H                 EXIT FROM BUFLRTN
         L     R14,BUFLSAVE       LOAD C(BUFLSAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                         FCL1RTN                                   *
*      - INITIALIZE FILTER CRITERIA LIST (FCL) HEADER AND ELEMENT   *
*      - INITIALIZE A FCL TO READ FOR TWO SPECIFIC SEQ DATASETS     *
*********************************************************************
*
FCL1RTN  DS    0H                 FCL INITIALIZATION ROUTINE
         ST    R14,FCL1SAVE       STORE C(R14) INTO SAVE AREA
         XC    FCLDEF2(FCLSIZE2),FCLDEF2  CLEAR FCL AREA
         LA    R1,FCLDEF2         R1 - FCL HEADER
         USING FCLMAP,R1          ESTABLISH ADDRESSABILITY
         LA    R2,FCLHDEND        R2 - FIRST FCL ELEMENT
         USING FCLDSN,R2          ESTABLISH ADDRESSABILITY
         MVC   FCLID,CFCLID       SET THE EYECATCHER 'FCL '
         MVC   FCLCOUNT,=H'2'     SET NUMBER OF FCL ELEMENTS
         MVI   FCL1FLAG,X'80'     SET FLAG FOR FULLY QUAL DSN
         MVI   FCLDSNLG,X'0F'     SET LENGTH FOR DSN1 (15)
         LA    R3,DS01            DSN=CVAFFLT1.DATA01
*                                 SEQ DS - 5 EXTENTS
*                                 1 FORMAT1 AND 1 FORMAT3
         ST    R3,FCLDSNA         SET ADDR OF DSN
         LA    R2,FCLDSNEL(R2)    LOAD R2 WITH ADDR OF 2ND FCL ELEMENT
         MVI   FCLDSNLG,X'0F'     SET LENGTH FOR DSN2 (15)
         LA    R3,DS02            DSN=CVAFFLT1.DATA02
*                                 SEQ DS - 5 EXTENTS
*                                 1 FORMAT1 AND 1 FORMAT3
         ST    R3,FCLDSNA         SET ADDR OF DSN
         DROP  R1,R2              DROP TEMP USING
FC1EXIT  DS    0H                 EXIT FROM FCLRTN
         L     R14,FCL1SAVE       LOAD C(FCL1SAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                        FCL2RTN                                    *
*     - INITIALIZE FILTER CRITERIA LIST (FCL) HEADER AND ELEMENT    *
*     - INITIALIZE A FCL TO READ FOR ONE SPECIFIC PDSE DATASET      *
*********************************************************************
*
FCL2RTN  DS    0H                 FCL INITIALIZATION ROUTINE
         ST    R14,FCL2SAVE       STORE C(R14) INTO SAVE AREA
         XC    FCLDEF(FCLSIZE),FCLDEF  CLEAR FCL AREA
         LA    R1,FCLDEF          R1 - FCL HEADER
         USING FCLMAP,R1          ESTABLISH ADDRESSABILITY
         LA    R2,FCLHDEND        R2 - FIRST (ONLY) FCL ELEMENT
         USING FCLDSN,R2          ESTABLISH ADDRESSABILITY
         MVC   FCLID,CFCLID       SET THE EYECATCHER 'FCL '
         MVC   FCLCOUNT,=H'1'     SET NUMBER OF FCL ELEMENTS
         MVI   FCL1FLAG,X'80'     SET FLAG FOR FULLY QUAL DSN
```

```
         MVI   FCLDSNLG,X'0F'     SET LENGTH FOR DSN (15)
         LA    R3,DS03            DSN=CVAFFLT1.PDSE01
*                                 PDSE DS - 122 EXTENTS
*                                 1 FORMAT1 AND 10 FORMAT3'S
         ST    R3,FCLDSNA         SET ADDR OF DSN
         DROP  R1,R2              DROP TEMP USING
FC2EXIT  DS    0H                 EXIT FROM FCLRTN
         L     R14,FCL2SAVE       LOAD C(FCL2SAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                           FCL3RTN                                 *
*     - INITIALIZE FILTER CRITERIA LIST (FCL) HEADER AND ELEMENT    *
*     - INITIALIZE A FCL TO READ FOR ONE SPECIFIC SEQ DATASET       *
*********************************************************************
*
FCL3RTN  DS    0H                 FCL INITIALIZATION ROUTINE
         ST    R14,FCL3SAVE       STORE C(R14) INTO SAVE AREA
         XC    FCLDEF(FCLSIZE),FCLDEF  CLEAR FCL AREA
         LA    R1,FCLDEF          R1 - FCL HEADER
         USING FCLMAP,R1          ESTABLISH ADDRESSABILITY
         LA    R2,FCLHDEND        R2 - FIRST (ONLY) FCL ELEMENT
         USING FCLDSN,R2          ESTABLISH ADDRESSABILITY
         MVC   FCLID,CFCLID       SET THE EYECATCHER 'FCL '
         MVC   FCLCOUNT,=H'1'     SET NUMBER OF FCL ELEMENTS
         MVI   FCL1FLAG,X'80'     SET FLAG FOR FULLY QUAL DSN
         MVI   FCLDSNLG,X'0F'     SET LENGTH FOR DSN (15)
         LA    R3,DS01            DSN=CVAFFLT1.DATA01
*                                 SEQ DS - 5 EXTENTS
*                                 1 FORMAT1 AND 1 FORMAT3
         ST    R3,FCLDSNA         SET ADDR OF DSN
         DROP  R1,R2              DROP TEMP USING
FC3EXIT  DS    0H                 EXIT FROM FCLRTN
         L     R14,FCL3SAVE       LOAD C(FCL3SAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                           FCL4RTN                                 *
*     - INITIALIZE FILTER CRITERIA LIST (FCL) HEADER AND ELEMENT    *
*     - INITIALIZE A FCL TO READ THE DSCBS ON THE ENTIRE VOLUME     *
*********************************************************************
*
FCL4RTN  DS    0H                 FCL INITIALIZATION ROUTINE
         ST    R14,FCL4SAVE       STORE C(R14) INTO SAVE AREA
         XC    FCLDEF(FCLSIZE),FCLDEF  CLEAR FCL AREA
         LA    R1,FCLDEF          R1 - FCL HEADER
         USING FCLMAP,R1          ESTABLISH ADDRESSABILITY
         LA    R2,FCLHDEND        R2 - FIRST (ONLY) FCL ELEMENT
         USING FCLDSN,R2          ESTABLISH ADDRESSABILITY
         MVC   FCLID,CFCLID       SET THE EYECATCHER 'FCL '
         MVC   FCLCOUNT,=H'1'     SET NUMBER OF FCL ELEMENTS
         MVI   FCL1FLAG,X'00'     SET FLAG FOR GENERIC DSN
         MVI   FCLDSNLG,X'02'     SET LENGTH FOR DSN - ** (02)
         LA    R3,=C'**'          ALL DATASETS ON THE VOLUME
         ST    R3,FCLDSNA         SET ADDR OF DSN
         DROP  R1,R2              DROP TEMP USING
FC4EXIT  DS    0H                 EXIT FROM FCLRTN
         L     R14,FCL4SAVE       LOAD C(FCL4SAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                          CVAFRD1                                  *
*     - INVOKE THE CFAFFILT MACRO AND READ THE DSCBS (1 DSN)        *
*********************************************************************
*
CVAFRD1  DS    0H                 CVAFFILT - READ ROUTINE (2 DSN)
         ST    R14,CVR1SAVE       STORE C(R14) INTO SAVE AREA
         SLR   R2,R2              ZERO OUT R2
         LA    R2,CVPLDEF         LOAD R2 WITH ADDR OF CVPL
         L     R3,UCBADD          LOAD R3 WITH UCB ADDRESS
         CVAFFILT ACCESS=READ,UCB=(R3),FCL=FCLDEF,BUFLIST=BFLHDEF,     X
               MF=(E,(R2))
         ST    R15,RETCODE        STORE RC FOR LATER INTERROGATION
CVR1EXIT DS    0H                 EXIT FROM CVAFRD1
         L     R14,CVR1SAVE       LOAD C(CVR1SAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                          CVAFRD2                                  *
*     - INVOKE THE CFAFFILT MACRO AND READ THE DSCBS (2 DSN'S)      *
*********************************************************************
*
```

```
CVAFRD2  DS    0H                 CVAFFILT - READ ROUTINE (2 DSN'S)
         ST    R14,CVR2SAVE       STORE C(R14) INTO SAVE AREA
         SLR   R2,R2              ZERO OUT R2
         LA    R2,CVPLDEF         LOAD R2 WITH ADDR OF CVPL
         L     R3,UCBADD          LOAD R3 WITH UCB ADDRESS
         LA    R4,FCLDEF2         LOAD R4 WITH ADDR OF FCLDEF2
         LA    R5,BFLHDEF         LOAD R5 WITH ADDR OF FCLDEF2
         CVAFFILT ACCESS=READ,UCB=(R3),FCL=(R4),BUFLIST=(R5),          X
               MF=(E,(R2))
         ST    R15,RETCODE        STORE RC FOR LATER INTERROGATION
CVR2EXIT DS    0H                 EXIT FROM CVAFRD2
         L     R14,CVR2SAVE       LOAD C(CVR2SAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                        CVAFRDA                                    *
*     - INVOKE THE CFAFFILT MACRO AND READ ALL THE DSCBS            *
*********************************************************************
*
CVAFRDA  DS    0H                 CVAFFILT - READ ALL DSCBS ROUTINE
         ST    R14,CVRASAVE       STORE C(R14) INTO SAVE AREA
         CVAFFILT ACCESS=READ,UCB=UCBADD,FCL=FCLDEF,BUFLIST=BFLHDEF,   X
               FLTAREA=KEEP,IOAREA=KEEP,                               X
               MF=(E,CVPLDEFA)
         ST    R15,RETCODE        STORE RC FOR LATER INTERROGATION
CVRAEXIT DS    0H                 EXIT FROM CVAFRDA
         L     R14,CVRASAVE       LOAD C(CVRASAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                        CVAFRS                                     *
*           - INVOKE THE CFAFFILT MACRO USING RESUME                *
*********************************************************************
*
CVAFRS   DS    0H                 CVAFFILT - RESUME ROUTINE
         ST    R14,CVRSSAVE       STORE C(R14) INTO SAVE AREA
CVRS0000 DS    0H                 NOW CHECK RC AND STAT CODES
         CH    R15,RCODE00        IS THE RC FROM RESUME 0?
         BE    CVRS0050            YES BRANCH TO CVRS0050
         CH    R15,RCODE04        IS THE RC FROM RESUME 4?
         BE    CVRS0020            YES BRANCH TO CVRS0020
CVRS0010 DS    0H                 ELSE PRINT MSG (RC IS 8,12,OR 16)
         MVC   PDETLINE(133),RCERMSG  MOVE MSG TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         B     CVRSEXIT           BRANCH TO EXIT ROUTINE
CVRS0020 DS    0H                 PROCESS FOR RC04 - CHECK FOR STAT064
         CLI   CVSTAT,STAT064     IS THE CVSTAT CODE 064(RESUME NEEDED)
         BNE   CVRS0060            NO - BRANCH TO CVRS0060
         CLI   RESFLG,X'01'       IS RESUME FLAG ON
         BE    CVRS0030            YES - BRANCH TO CVRS0030
         MVC   PDETLINE(133),ST64MSG1 MOVE MSG TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         MVC   PDETLINE(133),ST64MSG2 MOVE MSG TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         BAL   R14,FMTCTRTN       INVOKE FORMAT COUNT ROUTINE
         BAL   R14,FMTOPRTN       INVOKE FORMAT OUTPUT MSG ROUTINE
         MVI   RESFLG,X'01'       RESUME NEEDED - SET FLAG ON
         B     CVRS0040           BRANCH TO CVRS0040
CVRS0030 DS    0H                 RESUME PROCESSING
         MVC   PDETLINE(133),ST64MSG3 MOVE MSG TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         BAL   R14,FMTCTRTN       INVOKE FORMAT COUNT ROUTINE
         BAL   R14,FMTOPRTN       INVOKE FORMAT OUTPUT MSG ROUTINE
CVRS0040 DS    0H                 RESUME PROCESSING
         CVAFFILT ACCESS=RESUME,MF=(E,CVPLDEFA)
         B     CVRS0000           BRANCH TO CHECK RETURN CODE AGAIN
CVRS0050 DS    0H                 RC IS 0 NO LONGER NEED RESUME
         MVC   PDETLINE(133),ST64MSG3 MOVE MSG TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         BAL   R14,FMTCTRTN       INVOKE FORMAT COUNT ROUTINE
         BAL   R14,FMTOPRTN       INVOKE FORMAT OUTPUT MSG ROUTINE
         MVC   PDETLINE(133),ST64MSG4 MOVE MSG TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
*
*                                 RELEASE WORK AREAS
*
         CVAFFILT ACCESS=RLSE,FLTAREA=NOKEEP,IOAREA=NOKEEP,            X
               MF=(E,CVPLDEFA)
         B     CVRSEXIT           BRANCH TO EXIT ROUTINE
CVRS0060 DS    0H                 RC IS 4 BUT CVSTAT IS NOT 064
         MVC   PDETLINE(133),RC04MSG  MOVE MSG TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
```

```
CVRSEXIT DS    0H                 EXIT FROM CVAFRS
         L     R14,CVRSSAVE       LOAD C(CVRSSAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                          CVAFRL                                   *
*     - INVOKE THE CFAFFILT MACRO AND RELEASE WORK AREAS            *
*********************************************************************
*
CVAFRL   DS   0H                  CVAFFILT - RLSE ROUTINE
         ST   R14,CVRLSAVE        STORE C(R14) INTO SAVE AREA
         CVAFFILT ACCESS=RLSE,FCL=0,BUFLIST=0,FLTAREA=NOKEEP,          X
               MF=(E,CVPLDEF)
         CH    R15,RCODE00        IF RC = 0 THEN
         BE    CVRLEXIT              BRANCH TO EXIT
         MVC   PDETLINE(133),RLSEMSG ELSE MOVE MSG TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
CVRLEXIT DS    0H                 EXIT FROM CVAFRL
         L     R14,CVRLSAVE       LOAD C(CVRLSAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                      WORKING  STORAGE                             *
*********************************************************************
*
         DS    0D
         DC    CL36'CVAFFEXP-WORKING STORAGE BEGINS HERE'
*
*********************************************************************
*                          EQUATES                                  *
*********************************************************************
*
EABN100  EQU    100           USER ABEND CODE 100 - VTOC OPEN ERROR
EABN101  EQU    101           USER ABEND CODE 101 - OUTDD OPEN ERROR
EABN102  EQU    102           USER ABEND CODE 102 - OUTDD CLOSE ERROR
BUFNBR   EQU    11            11 BUFFERS TO BE USED
R0       EQU    0
R1       EQU    1
RDEB     EQU    1             REG1 FOR DEB ADDRESS
R2       EQU    2
RUCB     EQU    2             REG2 FOR UCB ADDRESS
R3       EQU    3
R4       EQU    4
R5       EQU    5
R6       EQU    6
R7       EQU    7
R8       EQU    8
R9       EQU    9
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
*
*********************************************************************
*                        SAVE AREAS                                 *
*********************************************************************
*
SAVE     DC    18F'0'             MAIN PROGRAM SAVE AREA
OPENSAVE DC    F'0'               OPEN FILES ROUTINE SAVE AREA
CLOSSAVE DC    F'0'               CLOSE FILES ROUTINE SAVE AREA
ABENSAVE DC    F'0'               ABEND ROUTINE SAVE AREA
IDVLSAVE DC    F'0'               IDNETIFY VOLUME ROUTINE SAVE AREA
BUFLSAVE DC    F'0'               INITIALIZE BUFFER ROUTINE SAVE AREA
FCL1SAVE DC    F'0'               INITIALIZE FCL1 ROUTINE SAVE AREA
FCL2SAVE DC    F'0'               INITIALIZE FCL2 ROUTINE SAVE AREA
FCL3SAVE DC    F'0'               INITIALIZE FCL3 ROUTINE SAVE AREA
FCL4SAVE DC    F'0'               INITIALIZE FCL4 ROUTINE SAVE AREA
TSTRSAVE DC    F'0'               TEST RETURN CODE ROUTINE SAVE AREA
CVR1SAVE DC    F'0'               CVAFFILT READ 1 DSN ROUTINE SAVE AREA
CVR2SAVE DC    F'0'               CVAFFILT READ 2 DSN ROUTINE SAVE AREA
CVRASAVE DC    F'0'               CVAFFILT READ ALL ROUTINE SAVE AREA
CVRLSAVE DC    F'0'               CVAFFILT RLSE ROUTINE SAVE AREA
CVRSSAVE DC    F'0'               CVAFFILT RESUME ROUTINE SAVE AREA
FMTCSAVE DC    F'0'               FORMAT DSCB COUNT ROUTINE SAVE AREA
FMTOSAVE DC    F'0'               FORMAT OUTPUT ROUTINE SAVE AREA
*
*********************************************************************
*                        CONSTANTS                                  *
*********************************************************************
*
RETCODE  DC    F'999'
```

```
RCODE00  DC    H'0'               RETURN CODE 0 - HALFWORD
RCODE04  DC    H'4'               RETURN CODE 4 - HALFWORD
DSNNBR   DC    X'FF'              INDICATE NBR OF DSNS TO PROCESS
RESFLG   DC    X'00'              RESUME FLAG - OFF
BLNKLNE  DC    CL133' '
STRTMSG  DC    CL133' CVAFFEXP START OF OUTPUT MESSAGES               '
ENDMSG   DC    CL133' CVAFFEXP END OF OUTPUT MESSAGES                 '
ST64MSG1 DC    CL133' RC04 VERIFIED - CVSTAT 064  RESUME IS NECESSARY '
ST64MSG2 DS    0CL133
         DC    CL49' CVAFFILT (INITIAL) RETURNED THE FOLLOWING DSCBS '
         DC    CL84'FOR THE VOLUME:                                   '
ST64MSG3 DS    0CL133
         DC    CL49' CVAFFILT (RESUME)  RETURNED THE FOLLOWING DSCBS '
         DC    CL84'FOR THE VOLUME:                                   '
ST64MSG4 DS    0CL133
         DC    CL48' CVAFFILT RESUME OPERATION COMPLETE - ALL DSCBS '
         DC    CL85'RETURNED'
DSCBMSGA DS    0CL133
         DC    CL48' CVAFFILT RETURNED THE FOLLOWING DSCBS FOR DSN:'
DS       DC    CL44' '
         DC    CL41' '
DSCBMSGB DS    0CL133
         DC    CL48'                                    AND FOR DSN:'
DS2      DC    CL44' '
         DC    CL41' '
DSCBMSGC DS    0CL133
         DC    CL48' CVAFFILT LOGICAL ERROR STATUS RETURNED  - DSN:'
DS3      DC    CL44' '
         DC    CL41' '
NODSCBM  DC    CL133' NO DSCBS RETURNED FROM CVAFFILT                 '
RC00MSG  DC    CL133' RC00 VERIFIED - THE REQUEST WAS SUCCESSFUL      '
RC04MSG  DC    CL133' RC04 VERIFIED - LOGICAL ERROR STATUS IN CVSTAT  '
RCERMSG  DC    CL133' RC08, RC12, OR RC16 RETURNED FROM CVAFFILT      '
RLSEMSG  DC    CL133' NON ZERO RETURN CODE BACK FROM RLSE             '
DSCBMSG1 DC    CL29'  NUMBER OF FORMAT 1 DSCBS - '
DSCBMSG2 DC    CL29'  NUMBER OF FORMAT 3 DSCBS - '
DS01     DC    CL44'CVAFFLT1.DATA01'
DS02     DC    CL44'CVAFFLT1.DATA02'
DS03     DC    CL44'CVAFFLT1.PDSE01'
CFCLID   DC    CL4'FCL '
*************************************************************************
*                         WORK AREAS                                    *
*************************************************************************
*
WF1      DS    D                  DOUBLE WORD - FORMAT 1 WORK AREA
WF3      DS    D                  DOUBLE WORD - FORMAT 3 WORK AREA
DEBADD   DC    F'0'               DEB ADDRESS SAVE AREA
UCBADD   DC    F'0'               UCB ADDRESS SAVE AREA
RETF1    DC    F'0'               COUNT OF FMT 1 DSCBS RET BY CVAFFILT
RETF3    DC    F'0'               COUNT OF FMT 3 DSCBS RET BY CVAFFILT
WFMTREC  DS    0CL133             WORK FORMAT RECORD FOR OUTPUT
MSG      DC    CL29' '            GENERAL MESSAGE
FMTCNT   DC    CL08' '            FORMAT COUNT
TOEOL    DC    CL96' '            SPACES TO END OF LINE
*
*************************************************************************
*                         PRINT LINES                                   *
*************************************************************************
*
         DS    0D                 TAKE CARE OF SLACK BYTES
PDETLINE DS    0CL133             DETAIL LINE
         DC    CL133' '
EPDETLEN EQU   *-PDETLINE         LENGTH OF DETAIL LINE
*
*************************************************************************
*                    DCB - OUTPUT FILE (OUTFILE)                        *
*************************************************************************
*
OUTFILE  DCB   DDNAME=OUTDD,                                           X
               DSORG=PS,                                               X
               RECFM=FBA,                                              X
               LRECL=133,                                              X
               MACRF=PM
*
*************************************************************************
*                     VTOC DCB AREA                                     *
*************************************************************************
*
VTOCDCB  DCB   DDNAME=CVAFDD,MACRF=E,EXLST=XLST1,DSORG=PS
XLST1    DC    X'87'
         DC    AL3(JFCB1)
JFCB1    DS    0CL176
```

```
TESTNAME DS    CL44
         DS    CL8
         DS    BL1
         DS    CL123
*
*************************************************************************
*                     MAPPPING MACROS                                   *
*************************************************************************
*
CVPLMAP  ICVAFPL CVPLFSA=YES      CVAF PARMLIST
FCLMAP   ICVFCL                   FILTER CRITERIA LIST
BFLMAP   ICVAFBFL                 BUFFER LIST
         PUSH  PRINT
         PRINT NOGEN
DSCBMAP  DSECT                    DSCB DSECT
         IECSDSL1 (1)             USE FMT 1 DSCB MAPPING TO GET BUFFER
DSCBSIZ  EQU   *-IECSDSL1         LENGTH OF FULL DSCB
         DCBD DSORG=XE,DEVD=DA    MAP OF DCB
         IEZDEB                   MAP OF DEB
         POP   PRINT
*
*
CVAFFEXP CSECT ,                  CONT OF CSECT AFTER MAPPING MACROS
*
*
*************************************************************************
*                       CVAF PARAMETER LISTS                            *
*************************************************************************
*
CVPLDEF  CVAFFILT MF=L,BRANCH=NO,FLTAREA=KEEP
CVPLDEFA CVAFFILT BRANCH=(YES,SUP),MF=L
*
*************************************************************************
*        SPACE ALLOCATION FOR CVPL, FCL, BFL, AND DSCB BUFFERS          *
*************************************************************************
*
FCLDEF   DS    (FCLHDLEN+FCLDSNEL)X FCL HEADER AND ONE FCL ELEMENT
FCLSIZE  EQU   *-FCLDEF
*                                  DEFINE A CVAF BUFFER LIST WITH
FCLDEF2  DS    (FCLHDLEN+2*FCLDSNEL)X FCL HEADER AND TWO FCL ELEMENTS
FCLSIZE2 EQU   *-FCLDEF2
*                                  DEFINE A CVAF BUFFER LIST WITH
*                                  N BUFFER LIST ELEMENTS
BFLHDEF  DS    (BFLHLN)X          BUFFER LIST HEADER
BFLEDEF  DS    (BUFNBR*BFLELN)X   N BUFFER LIST ELEMENTS
BFLSIZE  EQU   *-BFLHDEF
*                                  DEFINE N FULL DSCB BUFFERS
DSCBDEF  DS    (BUFNBR*DSCBSIZ)X
*
         END   CVAFFEXP           END OF CVAFFEXP
```

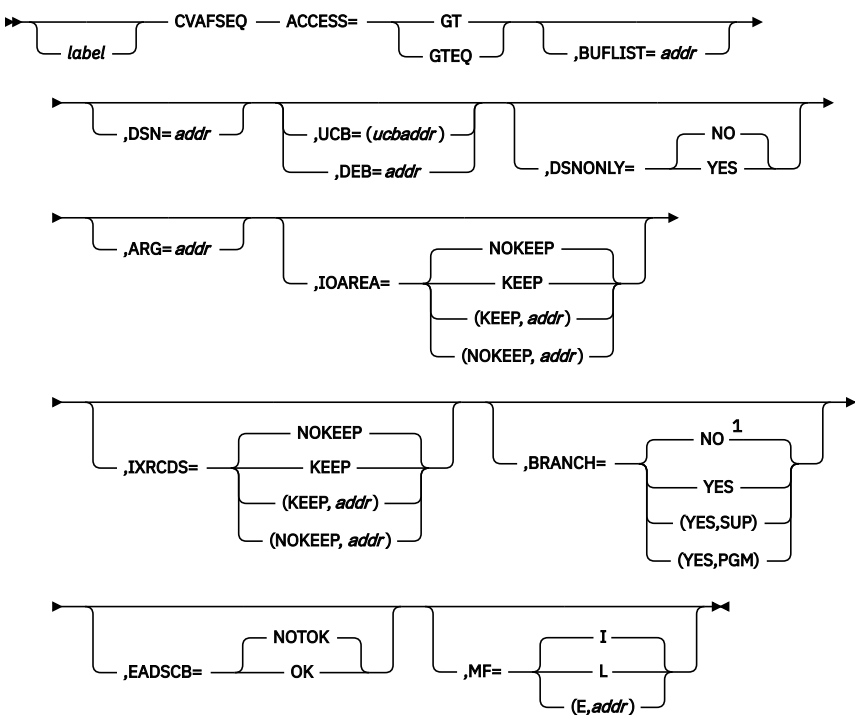
CVAFSEQ Macro Overview and Specification

The CVAFSEQ macro can be used to:

- Read an indexed VTOC sequentially in data-set-name (DSN) order
- Read an indexed VTOC or a nonindexed VTOC in physical-sequential order.

See “Accessing DSNs or DSCBs in Sequential Order” on page 64 for additional information.

The format of the CVAFSEQ macro is:



Notes:

[1] If YES is coded, the default is SUP.

ACCESS: Specify Relationship between Supplied and Returned DSN

ACCESS=GT
Specifies that the DSN or argument value is to be used to return a DSCB whose DSN or argument is greater than that supplied.

ACCESS=GTEQ
Specifies that the DSN or argument value is to be used to return a DSCB whose DSN or argument is greater than or equal to that supplied.

Recommendation: A CVAF call specifying ACCESS=GTEQ should be followed by an ACCESS=GT request, or the same DSCB or name is returned.

BUFLIST: Specify One or More Buffer Lists

BUFLIST=*addr*
Supplies the address of a buffer list used to read or write DSCBs and VIRs.

DSN: Specify Access by DSN Order or by Physical-Sequential Order

DSN=*addr*
Supplies the address of a 44-byte area containing either zeros or a data set name. Specifying the DSN keyword causes access of an indexed VTOC by DSN order. BUFLIST is required if DSNONLY=NO is coded or the default.

DSN omitted
If you omit the DSN keyword, access of an indexed or nonindexed VTOC is by physical-sequential order. BUFLIST is required.

If the order is physical-sequential, initialize the argument field in the first buffer list entry to zero or to the argument of the DSCB. If the argument is zero (BFLEARG=00), the read begins at the start of the

VTOC. You must be authorized (APF or system key) to read multiple DSCBs with a single invocation of the CVAFSEQ macro. See "Initiating Physical-Sequential Access" on page 65 for more information.

UCB or DEB: Specify the VTOC to Be Accessed

UCB= ***rs-type or (2-12) standard form*** **UCB=** ***rx-type or (2-12) execute form***

Specifies the address of the UCB for the VTOC to be accessed. The UCB address can be for a captured UCB, or for an actual UCB above or below the 16MB line. Use the address of a UCB, not a UCB copy. An unauthorized caller must not use this parameter. If your program is in 31-bit mode, this address must be in 31-bit address; the high order byte is part of the address. You should not code the UCB parameter with MF=L.

Recommendations::

- Code the address of the UCB parameter only as register (2-12). Coding an RX-type address gives you unpredictable results.
- Do not use the UCB address passed back in the CVPL from a previous CVAFSEQ request, particularly in AMODE=24, as it may be invalid (because it is a captured and then uncaptured address). The recommendation is to use IOAREA=KEEP.

DEB=*addr*

Supplies the address of a DEB opened to the VTOC you want to access. CVAF does not allow output requests to the VTOC or VTOC index if you specify the DEB subparameter. If you are not authorized, you cannot perform any asynchronous activity (such as EXCP, CLOSE, EOV), against the data set represented by the DEB because CVAF removes the DEB from the DEB table for the duration of the CVAF call. If you are not authorized (neither APF authorized nor in a system key), specify a DEB address, not a UCB, to CVAFSEQ. See "Identifying the Volume" on page 56 for further details.

If you supply a previously obtained I/O area through the IOAREA keyword, neither UCB nor DEB need be supplied. Otherwise, supply either a UCB or DEB. If you supply a UCB address, it is overlaid in the CVPL by the UCB address in the I/O area. If you supply both the UCB and the DEB addresses in the CVPL, the DEB address is used and the UCB address in the CVPL is overlaid by the UCB address in the DEB.

DSNONLY: Specify That Only the Data Set Name Is Read

This keyword is applicable only to accessing an indexed VTOC in DSN order.

DSNONLY=NO

Requests that the data set name be obtained from the VTOC index and the DSCB be read into a buffer supplied through the BUFLIST keyword. BUFLIST is required.

DSNONLY=YES

Requests that only the data set name be obtained from the VTOC index. If the ARG keyword is coded, the argument of the DSCB is returned.

ARG: Specify Where the Argument of the DSCB Is to Be Returned

This keyword is applicable only to accessing an indexed VTOC in DSN order with DSNONLY=YES coded.

ARG=*addr*

Supplies the address of the 5-byte area where the CCHHR of each data set name in the VTOC index is returned when DSNONLY=YES is coded.

IOAREA: Keep or Free the I/O Work Area

IOAREA=KEEP

Specifies that the CVAF I/O area associated with the CVAF parameter list is to be kept upon completion of the CVAF request. IOAREA=KEEP can be coded with BRANCH=NO only if the caller is authorized (APF, or system key).

If IOAREA=KEEP is coded, the caller must call CVAF with IOAREA=NOKEEP specified at some future time, whether or not any further VTOC access is required: for example, the recovery routine of the caller of CVAF.

Coding IOAREA=KEEP allows subsequent CVAF requests to be more efficient, because certain initialization functions can be bypassed. Neither DEB nor UCB need be specified when a previously obtained CVAF I/O area is supplied; nor can they be changed.

When IOAREA=KEEP is first issued, CVAF returns the CVAF I/O area in the CVAF parameter list (CVIOAR). Subsequent calls of CVAF can use that same parameter list, and CVAF obtains its I/O area from the CVIOAR.

When processing on the current volume is finished, release all areas that were kept.

Note: Do not switch back and forth between AMODE=24 and AMODE=31 during a succession of CVAFSEQ requests while IOAREA=KEEP is active. This can cause problems such as ABENDS in CVAF.

IOAREA=(KEEP,*addr*)
Supplies the address of a previously obtained I/O area. If a different CVAF parameter list is used, the previously obtained CVAF I/O area can be passed to CVAF by coding its address as the second parameter of the IOAREA keyword.

IOAREA=NOKEEP
Causes the work area to be freed upon completion of the CVAF request.

IOAREA=(NOKEEP,*addr*)
Causes a previously obtained work area to be freed upon completion of the CVAF request.

IXRCDS: Retain VIERs in Virtual Storage

This keyword applies to an indexed VTOC only.

IXRCDS=KEEP
Specifies that the VIERs read into storage during the CVAF function are to be kept in virtual storage. The VIERs are retained even if the index function is unsuccessful. The VIERs are accessed from the CVAF parameter list (CVIRCDS). CVIRCDS is the address of a buffer list containing the VIR buffer addresses and RBAs of the VIERs read.

Index search function dynamically updates the buffer list and, when necessary, obtains additional buffer lists and chains them together.

If IXRCDS=KEEP is specified and no buffer list is supplied to CVAF in the CVPL, CVAF obtains a buffer list and buffers and reads the first high-level VIER. The address of the buffer list is placed in the CVIRCDS field of the CVPL.

The buffer list and VIR buffers are in the caller's protect key. The subpool is 0 if the caller is not authorized; subpool 229 if the caller is authorized.

If IXRCDS=KEEP for an nonindexed VTOC, a request to read a DSCB can be performed, but an error code is returned.

When processing on the current volume is finished, release all areas that were kept.

IXRCDS=(KEEP,*addr*)
The CVIRCDS from one CVAF call can be passed to another CVAF parameter list by specifying the address as the second parameter in the IXRCDS keyword.

IXRCDS=NOKEEP
If IXRCDS=NOKEEP is coded, the VIERs that are accessed (if any) are not retained. Furthermore, the buffer list supplied in the CVIRCDS field in the CVAF parameter list is released, as are all buffers found in the buffer list. If the skip bit is set in any entry in the buffer list, the buffer and buffer list are not freed.

IXRCDS=(NOKEEP,*addr*)
specifies that previously accessed VIERs are not to be retained.

You must free buffer lists and buffers obtained by CVAF. This can be done in one of three ways:

- By coding IXRCDS=NOKEEP on the CVAFSEQ macro that obtained the buffers
- By coding IXRCDS=NOKEEP on a subsequent CVAF macro
- By coding CVAFDIR ACCESS=RLSE and providing the address of the buffer list in the BUFLIST keyword.

Requirement: You must enqueue the VTOC and reserve the unit to maintain the integrity of the VIERs read.

BRANCH: Specify the Entry to the Macro

BRANCH=(YES,SUP)
Requests the branch entry. The caller must be in supervisor state. Protect key checking is bypassed.

If BRANCH=YES is coded, an 18-word save area must be supplied. No lock can be held on entry to CVAF. SRB mode is not allowed.

BRANCH=YES
Equivalent to BRANCH=(YES,SUP), because SUP is the default when YES is coded. Protect key checking is bypassed.

BRANCH=(YES,PGM)
Requests the branch entry. The caller must be APF authorized and in problem state. Protect key checking is bypassed.

BRANCH=NO
requests the SVC entry. The caller must be APF authorized if any output operations are requested. Protect key checking is performed. This is the default.

EADSCB=value: Specify the support level for extended attribute DSCBs.

EADSCB=OK
This specification indicates that the calling program supports extended attribute DSCBs. An extended address volume may have these DSCBs allocated to it. The returned DSCBs (format-3, format-8) may contain extent descriptors described by 28-bit cylinder addresses or DSCBs (format-9) that contain additional attribute information.

For calls that initiate physical sequential access (DSN=0 or omitted), a CVAFSEQ request issued to an EAV volume will be failed if this new, EADSCB=OK, indicator is not set.

For calls that initiate index order (DSN=*address*) where the BUFLIST=*address* keyword is specified, a CVAFSEQ request issued to an EAV volume will be failed if the EADSCB=OK indicator is not set and the DSCB associated with this address is a format-8 DSCB.

The failing error code for these cases will be reflected as follows:

- CVAF status code (CVSTAT) set to STAT082.
- Return code 4

EADSCB=OK will set the CV4EADOK indicator in the CVPL. All other calls to CVAFSEQ are allowed and EADSCB=OK will be ignored. That is CVAFSEQ calls with DSNONLY=YES and ARG=*address* specified.

EADSCB=NOTOK
Indicates a calling program does not support extended attribute DSCBs. EADSCB=NOTOK will turn off the CV4EADOK indicator in the CVPL. This is the default.

MF: Specify the Form of the Macro

This keyword specifies whether the list, execute, or normal form of the macro is requested.

MF=I
If I is coded, or neither L nor E is coded, the CVAF parameter list is generated, as is code, to call CVAF. This is the normal form of the macro.

MF=L
Indicates the list form of the macro. A parameter list is generated, but code to call CVAF is not generated.

MF=(E,*addr*)
Indicates the execute form of the macro. The remote CVAF parameter list supplied as *addr* is used in and can be modified by the execute form of the macro.

Return Codes from CVAFSEQ

On return from CVAF, register 1 contains the address of the CVPL (CVAF parameter list), and register 15 contains one of the following return codes:

Return Code	Meaning
0 (X'00')	The request was successful.
4 (X'04')	End of data (CVSTAT is set to decimal 32), or an error was encountered. The CVSTAT field in the CVPL contains an indication of the cause of the error. Error descriptions are in "VTOC Index Error Message and Associated Codes" on page 127.
8 (X'08')	Invalid VTOC index structure. CVSTAT contains an indication of the cause of the error. Error descriptions are in "VTOC Index Error Message and Associated Codes" on page 127.
12 (X'0C')	The CVPL (CVAF parameter list) is not in your protect key, or is not valid (the ID is not valid, or the length field is incorrect, or the CVFCTN field is not valid). The CVPL has not been modified.
16 (X'10')	An I/O error was encountered.

Example of using the CVAFSEQ macro with an indexed VTOC

This example uses the CVAFSEQ macro to count the number of VSAM data sets whose data set names are within the range defined by two supplied data set names. The addresses of the two data set names are supplied to the program in registers 6 and 7, labeled RDSN1 and RDSN2, respectively. The address of a DEB open to the VTOC is supplied in register 4, labeled RDEB.

The CVAF parameter list is expanded by a list form of the CVAFSEQ macro. ACCESS=GTEQ is specified on the list form of macro and is, therefore, not coded in the first execution of the CVPL. Subsequent executions of the CVPL (at label RELOOP) specify ACCESS=GT.

End of data is tested by comparing the CVSTAT field to the value of STAT032, which is an equate in the ICVAFPL mapping macro.

The count of VSAM DSCBs matching the data set name criterion is returned in register 15, unless an error is encountered, in which case a negative 1 is returned in register 15.

```
SEQXMP1  CSECT
         STM   14,12,12(13)
         BALR  12,0
         USING *,12
         ST    13,SAVEAREA+4
         LA    RWORK,SAVEAREA
         ST    RWORK,8(,13)
         LR    13,RWORK
*************************************************************
*
*        REGISTERS
*
*************************************************************
REG1     EQU   1                  REGISTER 1
RWORK    EQU   3                  WORK REGISTER
RDEB     EQU   4                  DEB ADDRESS
RDSN1    EQU   6                  ADDRESS OF DATA SET NAME 1
RDSN2    EQU   7                  ADDRESS OF DATA SET NAME 2
REG15    EQU   15                 RETURN CODE REGISTER 15
```

```
*************************************************************
*
*        COUNT THE NUMBER OF VSAM DATA SETS WHOSE DATA SET NAMES ARE
*        BETWEEN DSN1 AND  DSN2 INCLUSIVELY.
*        RDSN1 CONTAINS ADDRESS OF DSN1.
*        RDSN2 CONTAINS ADDRESS OF DSN2.
*        ADDRESS OF DEB OPEN TO VTOC SUPPLIED IN RDEB.
*
*************************************************************
         XC    BUFLIST(BFLHLN+BFLELN),BUFLIST ZERO BUFFER LIST
         OI    BFLHFL,BFLHDSCB       DSCBS TO BE READ WITH BUFFER LIST
         MVI   BFLHNOE,1             ONE BUFFER LIST ENTRY
         LA    RWORK,DS1FMTID        ADDRESS OF DSCB BUFFER
         ST    RWORK,BFLEBUF         PLACE IN BUFFER LIST
         MVI   BFLELTH,DSCBLTH       DATA PORTION OF DSCB READ - DSN   *
                                     SUPPLIED IN CVPL
         MVC   DS1DSNAM,0(RDSN1)     MOVE IN STARTING DATA SET NAME TO *
                                     WORKAREA
         XR    RWORK,RWORK           ZERO COUNT
         CVAFSEQ DEB=(RDEB),         FIND FIRST DATA SET WHOSE DATA SET*
               BUFLIST=BUFLIST,      NAME IS GREATER THAN OR EQUAL TO  *
               EADSCB=OK,            THAT OF DSN1                      *
               MF=(E,CVPL)
LOOP     EQU   *                    LOOP UNTIL END OF DATA OR DATA SET *
                                    NAME GREATER THAN DSN2
         LTR   REG15,REG15          ANY ERROR
         BZ    TESTDSN              BRANCH IF NOT-CHECK DSN LIMIT
*************************************************************
*
*        DETERMINE WHAT ERROR IS
*
*************************************************************
         C     REG15,ERROR4         IS RETURN CODE 4
         BNE   OTHERERR             BRANCH IF NOT 4
         CLI   CVSTAT,STAT032       IS IT END OF DATA?
         BNE   OTHERERR             BRANCH IF NOT
*************************************************************
*
*        END OF DATA
*
*************************************************************
         B     RELEASE              RELEASE CVAF RESOURCES AND RETURN
TESTDSN  EQU   *                    IS DATA SET NAME GREATER THAN DSN2
         CLI   DS1FMTID,DS1IDC      IS THIS A FORMAT 1 DSCB?
         BE    CKVSAM               BRANCH IF FORMAT 1
         CLI   DS1FMTID,DS8IDC      IS THIS A FORMAT 8 DSCB?
         BNE   CKLAST               BRANCH IF NO. CAN NOT BE VSAM.
CKVSAM   EQU   *                    CHECK VSAM
         CLC   DS1DSNAM,0(RDSN2)    HAS LIMIT BEEN REACHED?
         BH    RELEASE              BRANCH IF HIGH TO RELEASE RESOURCES
         TM    DS1DSORG+1,DS1ORGAM IS DATA SET VSAM
         BZ    CKLAST               BRANCH IF NO-DO NOT COUNT IT
         LA    RWORK,1(,RWORK)      INCREMENT COUNT BY ONE
CKLAST   EQU   *                    CHECK IF LAST DATA SET NAME (DSN2)
         CLC   DS1DSNAM,0(RDSN2)    HAS LIMIT BEEN REACHED?
         BNH   RELOOP               BRANCH IF NO-READ NEXT ONE
         B     RELEASE              RELEASE CVAF RESOURCES AND RETURN
RELOOP   EQU   *                    READ NEXT DSCB
         CVAFSEQ ACCESS=GT,               GET DSCB WITH DATA SET NAME     *
               EADSCB=OK,                 GREATER THAN THE ONE LAST READ  *
               MF=(E,CVPL)
         B     LOOP                 CHECK RESULTS OF CVAFSEQ
OTHERERR EQU   *                    UNEXPECTED ERROR
*************************************************************
*
*        UNEXPECTED ERROR PROCESSING
*
*************************************************************
         LA    RWORK,1(0,0)         ONE IN RWORK
         LNR   RWORK,RWORK          SET NEGATIVE COUNT INDICATING ERROR
RELEASE  CVAFDIR ACCESS=RLSE,       RELEASE CVAF BUFFERS/IOAREA          *
               BUFLIST=0,           DO NOT RELEASE USER BUFFER LIST      *
               IXRCDS=NOKEEP,       RELEASE CVAF VIER BUFFERS            *
               MF=(E,CVPL)          RELEASE CVAF I/O AREA
         LR    REG15,RWORK          CURRENT COUNT IS RETURN CODE
         L     13,SAVEAREA+4
         RETURN (14,12),RC=(15)     RETURN CURRENT COUNT
ERROR4   DC    F'4'                 ERROR RETURN CODE 4
BUFLIST  ICVAFBFL DSECT=NO          BUFFER LIST
         IECSDSL1 (1)               FORMAT 1 DSCB DATASET NAME AND       *
                                    BUFFER
DSCBLTH  EQU   *-IECSDSL1-L'DS1DSNAM  LENGTH OF DATA PORTION OF DSCB
```

```
SAVEAREA DS    18F                SAVE AREA
CVPL     CVAFSEQ ACCESS=GTEQ,     READ DSCB WITH DSN GTEQ SUPPLIED DSN *
               IXRCDS=KEEP,       KEEP VIERS IN STORAGE DURING CALLS   *
               DSN=DS1DSNAM,      SUPPLIED DATA SET NAME               *
               BUFLIST=BUFLIST,                                        *
               EADSCB=OK,                                              *
               MF=L
         ORG   CVPL               EXPAND MAP OVER LIST
CVPLMAP  ICVAFPL DSECT=NO         CVPL MAP

         END
```

Example of using the CVAFSEQ macro to process a volume in physical sequential order

This example will use the CVAFSEQ macro to read through the DSCBs in physical sequential order. Although CVAFSEQ can be used to process both an indexed and non indexed volume in physical sequential order this example uses a non indexed volume. The CVAFSEQ call will return DSCBs where BFLEARG is set to a starting CCHHR. This value is initially set to zero and the CVAFSEQ call uses ACCESS=GT. A buffer list with five buffer list entries is contained within the program and is used to read up to five DSCBs at a time.

Output from this program will be to OUTDD. It will be a list of all the datasets on the volume and their corresponding CCHHRs. The output from this program is based on the volume and dataset information detailed within the source code example. If the output received is different it can be verified using the IEHLIST utility.

This program must be APF authorized because it uses the UCB= and BRANCH=(YES,PGM) parameters. It is bad programming practice to give a program APF authorization unnecessarily. In this case these two options just give a slight performance improvement. To remove APF authorization from this example, perform these steps:

- Remove the SETCODE and ENTRY statements for the binder.
- Replace the UCB parameter with DEB=(reg). Precede the CVAFSEQ macro with an instruction to load the specified register from DEBADD.
- Remove the BRANCH parameter on CVAFSEQ.

Sample JCL for CVAFSEQ macro to process a volume in sequential order

The following is the sample JCL used to Assemble, Link, and Execute the example source. Changes will have to be made to this JCL, as appropriate, for each customer environment.

```
//SEQXMP2  JOB ,MSGCLASS=X,TIME=(,10),
//             NOTIFY=&SYSUID
//*
//STEP01  EXEC PROC=ASMACLG
//SYSIN     DD *
    (INCLUDE EXAMPLE SOURCE HERE)
/*
//*
//L.SYSLMOD DD DSN=YOUR.AUTH.LINKLIB(SEQXMP2),DISP=SHR
//L.SYSIN   DD *
  SETCODE AC(1)
  ENTRY   SEQXMP2
/*
//*
//G.SYSABEND DD  SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//*
//G.CVAFDD  DD DISP=SHR,UNIT=3390,VOL=SER=339003
//G.OUTDD   DD DSN=CVAFSEQ1.OUTPUT,
//             DISP=(NEW,CATLG),
//             UNIT=3390,VOL=SER=339001,
//             SPACE=(TRK,(2,2)),
//             DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//*
//
```

Code example of the CVAFSEQ macro to process a volume in sequential order

```
SEQXMP2 TITLE 'CVAF CVAFSEQ - SEQXMP2 EXAMPLE'
SEQXMP2 CSECT
SEQXMP2 AMODE 24
SEQXMP2 RMODE 24
*
*********************************************************************
*                                                                   *
*    SEQXMP2 - THIS MODULE WILL READ THROUGH THE DSCBS ON A         *
*              NONINDEXED VOLUME.  IT WILL USE THE CVAFSEQ MACRO    *
*              TO READ UP TO 5 DSCBS AT A TIME IN PHYSICAL          *
*              SEQUENTIAL ORDER.                                    *
*              OUTPUT FROM THIS MODULE WILL BE TO OUTDD.  IT WILL   *
*              LIST ALL OF THE DATASETS ON THE VOLUME AND THEIR     *
*              CORRESPONDING CCHHR'S.  THE FMT4, FMT5, AND FMT7 DSCBS *
*              AND THEIR CORRESPONDING CCHHRS WILL NOT BE LISTED.   *
*                                                                   *
*              THE FOLLOWING DATASETS WERE ALLOCATED IN THE FOLLOWING *
*              ORDER ON A NONINDEXED VOLUME.                        *
*              10 SEQUENTIAL DATASETS: CVAFSEQ1.SEQ01-CVAFSEQ1.SEQ10  *
*              5 PDS DATASETS: CVAFSEQ1.PDS01-CVAFSEQ1.PDS05        *
*              5 PDSE DATASETS: CVAFSEQ1.PDSE01-CVAFSEQ1.PDSE05     *
*              2 VSAM DATASETS: CVAFSEQ1.VSAM01-CVAFSEQ1.VSAM02     *
*                                                                   *
*    NOTE: THE OUTPUT FROM THIS EXAMPLE IS BASED ON USING A NON SMS *
*          MANAGED NON INDEXED VOLUME FOR ALLOCATING THE TEST       *
*          DATASETS (CVAFDD DD).                                    *
*          IF A SMS MANAGED NON INDEXED VOLUME IS USED THE POSITION *
*          OF THE VVDS DATASET WILL BE THE SECOND ENTRY IN THE LIST. *
*                                                                   *
*    DASD VOLUMES USED IN THIS EXAMPLE:                             *
*    - 1 3390-3 WHERE THE OUTDD DATASET IS DEFINED                  *
*    - 1 3390-3 WHERE THE TEST DATASETS DETAILED ABOVE ARE DEFINED  *
*        THIS VOLUME INITIALIZED USING:   VTOC(3326,0,195) NOINDEX  *
*                                                                   *
*                                                                   *
*                                                                   *
*          OUTPUT IN OUTDD DATASET SHOULD BE THE FOLLOWING:         *
*-------------------------------------------------------------------*
*                                                                   *
*                                                                   *
* SEQXMP2 START OF OUTPUT MESSAGES                                  *
*                                                                   *
* DSN: CVAFSEQ1.SEQ01                          CCHHR: 0CFE000003  *
* DSN: CVAFSEQ1.SEQ02                          CCHHR: 0CFE000004  *
* DSN: CVAFSEQ1.SEQ03                          CCHHR: 0CFE000005  *
* DSN: CVAFSEQ1.SEQ04                          CCHHR: 0CFE000006  *
* DSN: CVAFSEQ1.SEQ05                          CCHHR: 0CFE000007  *
* DSN: CVAFSEQ1.SEQ06                          CCHHR: 0CFE000008  *
* DSN: CVAFSEQ1.SEQ07                          CCHHR: 0CFE000009  *
* DSN: CVAFSEQ1.SEQ08                          CCHHR: 0CFE00000A  *
* DSN: CVAFSEQ1.SEQ09                          CCHHR: 0CFE00000B  *
* DSN: CVAFSEQ1.SEQ10                          CCHHR: 0CFE00000C  *
* DSN: CVAFSEQ1.PDS01                          CCHHR: 0CFE00000D  *
* DSN: CVAFSEQ1.PDS02                          CCHHR: 0CFE00000E  *
* DSN: CVAFSEQ1.PDS03                          CCHHR: 0CFE00000F  *
* DSN: CVAFSEQ1.PSD04                          CCHHR: 0CFE000010  *
* DSN: CVAFSEQ1.PDS05                          CCHHR: 0CFE000011  *
* DSN: CVAFSEQ1.PDSE01                         CCHHR: 0CFE000012  *
* DSN: CVAFSEQ1.PDSE02                         CCHHR: 0CFE000013  *
* DSN: CVAFSEQ1.PDSE03                         CCHHR: 0CFE000014  *
* DSN: CVAFSEQ1.PDSE04                         CCHHR: 0CFE000015  *
* DSN: CVAFSEQ1.PDSE05                         CCHHR: 0CFE000016  *
* DSN: CVAFSEQ1.VSAM01.DATA                    CCHHR: 0CFE000017  *
* DSN: SYS1.VVDS.V339003                       CCHHR: 0CFE000018  *
* DSN: CVAFSEQ1.VSAM02.DATA                    CCHHR: 0CFE000019  *
*                                                                   *
* END OF DATA REACHED - ALL DATA SETS PROCESSED                     *
*                                                                   *
* SEQXMP2 END OF OUTPUT MESSAGES                                    *
*                                                                   *
*                                                                   *
*-------------------------------------------------------------------*
*                                                                   *
*********************************************************************
*                                                                   *
*    SEQXMP2 - LOGIC NOTES                                          *
*                                                                   *
*    THIS MODULE WILL PERFORM THE FOLLOWING:                        *
*                                                                   *
```

```
*    INITIALIZATION                                                 *
*    - INITIALIZE STARTING CCHHR VALUE TO 0 (GT ZERO USED IN CVAFSEQ) *
*    - OBTAIN THE NECESSARY INFORMATION FROM THE DASD VOLUME        *
*    - OPEN AN OUTPUT FILE                                          *
*    - WRITE NECESSARY MESSAGES TO THE OUTPUT FILE                  *
*                                                                   *
*    MAINLINE                                                       *
*      DO WHILE MORE DATASETS ON VOLUME TO PROCESS                  *
*      - LOAD 5 ENTRY TABLE WITH DSCB ADDRESS                       *
*      - INIT BUFFER LIST HDR TO READ DSCBS AND STARTING CCHHR      *
*      - INIT BUFFER LIST ENTRY WITH DSCB ADDRESS AND LENGTH        *
*      - ISSUE CVAFSEQ MACRO TO READ 5 ENTRIES                      *
*      - LOAD CCHHR RETURNED FROM CVAFSEQ INTO TABLE                *
*      - PROCESS TABLE ENTRIES AND PRODUCE OUTPUT: DSN/CCHHR LIST   *
*                                                                   *
*    FINALIZATION                                                   *
*    - WRITE NECESSARY MESSAGES TO THE OUTPUT FILE                  *
*    - CLOSE THE OUTPUT FILE                                        *
*    - EXIT                                                         *
*                                                                   *
*                                                                   *
*    SEQXMP2 - JOB INFORMATION                                      *
*                                                                   *
*    NORMAL END OF JOB:                                             *
*    - RC=00 AND OUTDD OUTPUT AS DETAILED ABOVE                     *
*                                                                   *
*                                                                   *
*    ABNORMAL END OF JOB:                                           *
*    - ABEND 100 - ERROR OPENING VTOC ON THE DASD VOLUME THAT IS    *
*                  ASSOCIATED WITH THE CVAFDD DD STATEMENT          *
*    - ABEND 101 - ERROR OPENING THE OUTDD DATASET                  *
*    - ABEND 102 - ERROR CLOSING THE OUTDD DATASET                  *
*                                                                   *
*                                                                   *
*                                                                   *
*********************************************************************
*
*********************************************************************
*                                                                   *
* HOUSEKEEPING                                                      *
* - SAVE CALLER'S REGISTERS AND ESTABLISH A NEW REGISTER SAVE AREA   *
*                                                                   *
*********************************************************************
*
         STM   R14,R12,12(R13)    STANDARD LINKAGE CONVENTION
         BALR  R11,0              DCL R11 AS IMPLIED BASE REG
         USING BASE,R11,R12       R12 IS ALSO BASE REG
BASE     L     R12,BASEADDR       SET UP ADDRESSING FOR R12
         B     CVAFSQ00           BRANCH AROUND DECLARES
BASEADDR DC    A(BASE+4096)       ADDRESSING FOR R12
CVAFSQ00 DS    0H                 CONTINUE...
         ST    R13,SAVE+4         SAVE PTR TO CALLER'S SAVE AREA
         LA    R14,SAVE           GET ADDRESS OF THE NEW SAVE AREA
         ST    R14,8(,R13)        CHAIN CALLER'S AREA TO OURS
         LR    R13,R14            ESTABLISH THE NEW SAVE AREA
*
*********************************************************************
*                                                                   *
* INITIALIZATION                                                    *
*                                                                   *
*********************************************************************
*
INITIAL  DS    0H                 INITIALIZATION SECTION
         MVC   CCHHRS,CCHHR0      INIT CCHHR START TO ZERO
         BAL   R14,IDVOLRTN       INVOKE RTN TO IDENTIFY THE VOLUME(S)
         BAL   R14,OPENRTN        INVOKE OPEN OUTPUT DATASET RTN
         MVC   PDETLINE(133),STRTMSG  MOVE START MSG TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         MVC   PDETLINE(133),BLNKLNE  MOVE BLANK LINE TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
*
*********************************************************************
*                                                                   *
* MAINLINE                                                          *
*                                                                   *
*********************************************************************
*
MAINLINE DS    0H                 MAINLINE SECTION
         MVI   SWEOD,NOEOD        SET SWITCH TO NO EOD INITIALLY
         BAL   R14,LDTABRTN       INVOKE LDTABRTN TO LOAD TABLE
         BAL   R14,INITBRTN       INVOKE INITBRTN TO INIT BUFFER LIST
         BAL   R14,CVAFSRTN       INVOKE CVAFSRTN TO ISSUE CVAFSEQ CALL
```

```
         BAL   R14,LODCRTN        INVOKE LODCRTN TO LOAD CCHHR IN TBL
         BAL   R14,PRTBRTN        INVOKE PRTBRTN TO PROCESS TBL ENTRIES
         CLI   SWEOD,EOD          DID WE REACH THE END OF DATA?
         BNE   MAINLINE           NO, PROCESS MORE DATA
         L     R15,RETCODE        LOAD R15 WITH RETURN CODE
         CH    R15,RCODE00        IF RC WAS 00?
         BNE   FINAL                 NO - DO NOT PRINT EOD MESSAGE
         MVC   PDETLINE(133),BLNKLNE YES- MOVE BLANK LINE TO LINE
         PUT   OUTFILE,PDETLINE           WRITE THE REC TO OUTPUT FILE
         MVC   PDETLINE(133),EODMSG       MOVE EOD MSG TO LINE
         PUT   OUTFILE,PDETLINE           WRITE THE REC TO OUTPUT FILE
*
**************************************************************************
*                                                                        *
* FINALIZATION                                                           *
*                                                                        *
**************************************************************************
*
FINAL    DS    0H                 FINALIZATION SECTION
         MVC   PDETLINE(133),BLNKLNE  MOVE BLANK LINE TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         MVC   PDETLINE(133),ENDMSG  MOVE END MSG TO LINE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         BAL   R14,CLOSERTN       INVOKE CLOSE OUTPUT DATASET RTN
         L     R13,4(R13)         RESTORE REGISTER
         LM    R14,R12,12(R13)    RESTORE CALLERS REGISTERS
         LA    R15,0              SET RC TO 0
         BR    R14                RETURN TO CALLER
*
**************************************************************************
*                        OPENRTN                                         *
*          - ROUTINE TO OPEN OUTPUT FILE USED BY THIS MODULE             *
**************************************************************************
*
OPENRTN  DS    0H                 OPEN FILES ROUTINE
         ST    R14,OPENSAVE       STORE C(R14) INTO SAVE AREA
         OPEN  (OUTFILE,(OUTPUT)) OPEN THE OUTDD OUTPUT FILE FOR MSGS
         TM    OUTFILE+(DCBOFLGS-IHADCB),DCBOFOPN   IS FILE OPEN
         BO    OPENEXIT           FILE OPEN OK - EXIT OPEN RTN
         LA    R1,EABN101         OUTPUT FILE NOT OPEN-USER ABEND 101
         BAL   R14,ABENDRTN       INVOKE ABEND ROUTINE
OPENEXIT DS    0H                 EXIT FROM OPEN ROUTINE
         L     R14,OPENSAVE       LOAD C(OPENSAVE) INTO R14
         BR    R14                EXIT
*
**************************************************************************
*                       CLOSERTN                                         *
*          - ROUTINE TO CLOSE OUTPUT FILE USED BY THIS MODULE            *
**************************************************************************
*
CLOSERTN DS    0H                 CLOSE FILES ROUTINE
         ST    R14,CLOSSAVE       STORE C(R14) INTO SAVE AREA
         CLOSE (OUTFILE)          CLOSE OUTPUT FILE
         LTR   R15,R15            CHECK IF CLOSED OK
         BZ    CLOSEXIT           IF OK BRANCH TO CLOSEXIT
         LA    R1,EABN102         ELSE SETUP FOR USER ABEND 102
         BAL   R14,ABENDRTN       INVOKE ABEND ROUTINE
CLOSEXIT DS    0H                 EXIT FROM CLOSE ROUTINE
         L     R14,CLOSSAVE       LOAD C(CLOSSAVE) INTO R14
         BR    R14                EXIT
*
**************************************************************************
*                       ABENDRTN                                         *
*                  - FORCE AN ABEND ROUTINE                              *
**************************************************************************
*
ABENDRTN DS    0H                 ABEND ROUTINE
         ST    R14,ABENSAVE       STORE C(R14) INTO SAVE AREA
         ABEND (R1),DUMP          ISSUE USER ABEND WITH DUMP
ABENEXIT DS    0H                 EXIT FROM ABEND ROUTINE
         L     R14,ABENSAVE       LOAD C(ABENSAVE) INTO R14
         BR    R14                EXIT
*
**************************************************************************
*                         IDVOLRTN                                       *
*     - OBTAIN THE NECESSARY INFORMATION FROM THE DASD VOLUME            *
**************************************************************************
*
IDVOLRTN DS    0H                 IDENTIFY VOLUME ROUTINE
         ST    R14,IDVLSAVE       STORE C(R14) INTO SAVE AREA
         RDJFCB (VTOCDCB,(INPUT)) READ JFCB / OPEN VTOC
         MVI   JFCB1,X'04'        PUT IN ID FOR FORMAT 4
```

```
         MVC   JFCB1+1(43),JFCB1  SETUP FOR VTOC OPEN
         OPEN  (VTOCDCB,(INPUT)),TYPE=J  OPEN VTOC (OPEN TYPE=J)
         TM    VTOCDCB+(DCBOFLGS-IHADCB),DCBOFOPN  TEST FOR GOOD OPEN
         BO    IDVOL010           BRANCH TO IDVOL010 - GOOD OPEN
         LA    R1,EABN100         ELSE SETUP FOR USER ABEND 100
         BAL   R14,ABENDRTN       INVOKE ABEND ROUTINE
IDVOL010 DS    0H                 GOOD OPEN - OBTAIN VOLUME INFORMATION
         SLR   RDEB,RDEB          INIT REG1 FOR DEB PTR
         SLR   RUCB,RUCB          INIT REG4 FOR UCB PTR
         ICM   RDEB,B'0111',VTOCDCB+(DCBDEBA-IHADCB) GET DEB ADDRESS
         ST    RDEB,DEBADD        SAVE DEB ADDRESS INTO R1
         ICM   RUCB,B'0111',(DEBBASND-DEBBASIC)+(DEBUCBA-DEBDASD)(RDEB)
         ST    RUCB,UCBADD        SAVE UCB ADDRESS INTO R4
IDVLEXIT DS    0H                 EXIT FROM IDVOLRTN
         L     R14,IDVLSAVE       LOAD C(IDVLSAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                         LDTABRTN                                  *
*  - LOAD 5 ENTRY TABLE WITH DSCB ADDRESS TO USE                    *
*********************************************************************
*
LDTABRTN DS    0H                 LOAD TABLE ROUTINE
         ST    R14,LDTBSAVE       STORE C(R14) INTO SAVE AREA
         LA    R6,DSCB01          LOAD R6 WITH ADDRESS OF DSCB01
         ST    R6,TDSCB01         STORE ADDRESS OF DSCB01 INTO TABLE
         LA    R6,DSCB02          LOAD R6 WITH ADDRESS OF DSCB02
         ST    R6,TDSCB02         STORE ADDRESS OF DSCB02 INTO TABLE
         LA    R6,DSCB03          LOAD R6 WITH ADDRESS OF DSCB03
         ST    R6,TDSCB03         STORE ADDRESS OF DSCB03 INTO TABLE
         LA    R6,DSCB04          LOAD R6 WITH ADDRESS OF DSCB04
         ST    R6,TDSCB04         STORE ADDRESS OF DSCB04 INTO TABLE
         LA    R6,DSCB05          LOAD R6 WITH ADDRESS OF DSCB05
         ST    R6,TDSCB05         STORE ADDRESS OF DSCB05 INTO TABLE
LDTBEXIT DS    0H                 EXIT FROM LDTABRTN
         L     R14,LDTBSAVE       LOAD C(LDTBSAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                         INITBRTN                                  *
*  - INITIALIZE THE BUFFER LIST                                     *
*********************************************************************
*
INITBRTN DS    0H                 INITIALIZE BUFFER LIST ROUTINE
         ST    R14,INITSAVE       STORE C(R14) INTO SAVE AREA
         LA    R7,BUFLISTE        LOAD R7 WITH ADDRESS OF BUFLIST ENTRY
         USING BFLE,R7            ESTABLISH ADDRESSABILITY TO BFLE
         LA    R8,BUFLISTH        LOAD R8 WITH ADDRESS OF BUFLIST HDR
         USING BFLHDR,R8          ESTABLISH ADDRESSABILITY TO BFLHDR
         LA    R2,TABLE           LOAD R2 WITH ADDRESS OF TABLE
         USING TBLMAP,R2          ESTABLISH ADDRESSABILITY USING TBLMAP
         XC    BFLHDR(BFLHLN+TBLNBR*BFLELN),BFLHDR CLEAR BUFLIST
         OI    BFLHFL,BFLHDSCB    DSCBS TO BE READ WITH BUFFER LIST
         MVC   BFLEARG,CCHHRS     MOVE STARTING CCHHR TO ARG
         MVI   BFLHNOE,TBLNBR     MOVE NBR OF TBL ENTRIES TO BUFE NBR
         SLR   R10,R10            INIT R10 WITH ZERO
         IC    R10,BFLHNOE        NBR OF BUFFER ENTRIES IN R10
         ST    R10,COUNT          NBR OF BUFFER ENTRIES IN COUNT
*
INIT0010 DS    0H                 INIT BUFFER LIST WITH DSCB ADDR/LENG
         L     R6,DSCBA           LOAD R6 WITH DSCB ADDRESS FROM TABLE
         ST    R6,BFLEBUF-BFLE(,R7)      PLACE IN BUFFER LIST
         MVI   BFLELTH-BFLE(R7),DSCBLTH  FULL DSCB READ
         OI    BFLEFL,BFLECHR     CCHHR TO BE RETURNED
         LA    R2,TBLLNG(,R2)     POINT TO NEXT TABLE ENTRY
         LA    R7,BFLELN(,R7)     POINT TO NEXT BUFFER LIST ENTRY
         BCT   R10,INIT0010       BRANCH TO INIT0010 IF C(R10) GT ZERO
         DROP  R2                 DROP R2
INITEXIT DS    0H                 EXIT FROM INITBRTN
         L     R14,INITSAVE       LOAD C(INITSAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                         CVAFSRTN                                  *
*  - CVAFSEQ REQUEST TO READ UP TO 5 ENTRIES AND PLACE DATA FOR EACH  *
*    ENTRY STARTING AT THE CORRESPONDING DSCB ADDRESS IN THE BUFFER   *
*    LIST ENTRY.                                                    *
*********************************************************************
*
CVAFSRTN DS    0H                 CVAFSEQ REQUEST ROUTINE
         ST    R14,CVAFSAVE       STORE C(R14) INTO SAVE AREA
         L     RUCB,UCBADD        LOAD R4 WITH UCB ADDRESS
```

```
         CVAFSEQ UCB=(RUCB),      CVAFSEQ MACRO INVOCATION            X
               EADSCB=OK,                                            X
               BRANCH=(YES,PGM),                                     X
               MF=(E,CVPL)
         ST    R15,RETCODE        STORE RC INTO RETCODE
         LTR   R15,R15            TEST RC RETURNED FROM CVAFSEQ
         CH    R15,RCODE00        IS IT A RC00?
         BZ    CVAFEXIT             YES - BRANCH TO EXIT ROUTINE
         CH    R15,RCODE04        IS IT A RC04?
         BE    CVAFSR04             YES - BRANCH TO PROCESS RC04
         CH    R15,RCODE08        IS IT A RC08?
         BE    CVAFSR08             YES - BRANCH TO PROCESS RC08
         CH    R15,RCODE12        IS IT A RC12?
         BE    CVAFSR12             YES - BRANCH TO PROCESS RC12
         CH    R15,RCODE16        IS IT A RC16?
         BE    CVAFSR16               YES - BRANCH TO PROCESS RC16
         MVC   PDETLINE(133),RCERMSG ELSE FORMAT UNEXPECTED RC
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         B     CVAFEXIT           EXIT ROUTINE
CVAFSR04 DS    0H                 RETURN CODE 04
         MVC   PDETLINE(133),RC04MSG FORMAT RC04 MESSAGE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         CLI   CVSTAT,STAT032     IS CVSTAT CODE STAT032?
         BE    CVAFSS32             YES - BRANCH TO PROCESS STAT032
         MVC   PDETLINE(133),STAT2MSG NO -  FORMAT STAT2MSG
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         B     CVAFEXIT           BRANCH TO EXIT ROUTINE
CVAFSS32 DS    0H                 CVSTAT CODE = STAT032
         MVC   PDETLINE(133),STAT1MSG FORMAT STAT1MSG
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         B     CVAFEXIT           BRANCH TO EXIT ROUTINE
CVAFSR08 DS    0H                 RETURN CODE 08
         MVC   PDETLINE(133),RC08MSG FORMAT RC08 MESSAGE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         B     CVAFEXIT           BRANCH TO EXIT ROUTINE
CVAFSR12 DS    0H                 RETURN CODE 12
         MVC   PDETLINE(133),RC12MSG FORMAT RC12 MESSAGE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
         B     CVAFEXIT           BRANCH TO EXIT ROUTINE
CVAFSR16 DS    0H                 RETURN CODE 16
         MVC   PDETLINE(133),RC16MSG FORMAT RC16 MESSAGE
         PUT   OUTFILE,PDETLINE   WRITE A RECORD TO THE OUTPUT FILE
CVAFEXIT DS    0H                 EXIT FROM CVAFSRTN
         L     R14,CVAFSAVE       LOAD C(CVAFSAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                          LODCRTN                                  *
*  - LOAD CCHHR FROM BUFFER LIST ENTRY INTO PROCESSING TABLE.  THIS  *
*    VALUE RETURNED FROM CVAFSEQ CALL.                              *
*********************************************************************
*
LODCRTN  DS    0H                 LOAD CCHHR INTO PROCESSING TABLE
         ST    R14,LODCSAVE       STORE C(R14) INTO SAVE AREA
         DROP  R7,R8              DROP R7 AND R8
         LA    R7,BUFLISTE        LOAD R7 WITH ADDRESS OF BUFLIST ENTRY
         USING BFLE,R7            ESTABLISH ADDRESSABILITY TO BFLE
         LA    R2,TABLE           LOAD R2 WITH ADDRESS OF TABLE
         USING TBLMAP,R2          ESTABLISH ADDRESSABILITY USING TBLMAP
         L     R10,COUNT          LOAD R10 WITH TABLE ENTRY COUNT
*
LODC0010 DS    0H                 PROCESS CCHHR VALUE RETURNED
         LA    R8,BFLEARG         LOAD R8 WITH ADDRESS OF BFLEARG
         ST    R8,CCHHRA          STORE CCHHR VALUE INTO TABLE
         LA    R2,TBLLNG(,R2)     POINT TO NEXT TABLE ENTRY
         LA    R7,BFLELN(,R7)     POINT TO NEXT BUFFER LIST ENTRY
         BCT   R10,LODC0010       BRANCH TO LODC0010 IF C(R10) GT ZERO
         DROP  R2,R7              DROP R2 AND R7
LODCEXIT DS    0H                 EXIT FROM LODCRTN
         L     R14,LODCSAVE       LOAD C(LODCSAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                          PRTBRTN                                  *
*  - PROCESS TABLE WHICH CONTAINS ADDRESS OF DSCB AND ADDRESS OF    *
*    CCHHR FOR EACH ENTRY RETURNED FROM CVAFSEQ CALL.  TABLE IS     *
*    CURRENTLY 5 ENTRIES.                                           *
*********************************************************************
*
PRTBRTN  DS    0H                 PROCESS TABLE ENTRIES (DSCBA/CCHHRA)
         ST    R14,PRTBSAVE       STORE C(R14) INTO SAVE AREA
         L     R10,COUNT          LOAD COUNT IN R10
```

```
         LA    R2,TABLE           LOAD ADDRESS OF TABLE INTO R2
         USING TBLMAP,R2          ESTABLISH ADDRESSABILITY TO TABLE
PRTB0000 DS    0H                 PROCESS ENTRIES
         L     R3,DSCBA           ADDRESSABILITY TO DSCBA
         L     R4,CCHHRA          ADDRESSABILITY TO CCHHRA
         CLC   0(1,R3),FMT4       IS IT A FMT4?
         BE    PRTB0020           YES, BRANCH TO POINT TO NEXT ENTRY
         CLC   0(1,R3),FMT5       IS IT A FMT5?
         BE    PRTB0020           YES, BRANCH TO POINT TO NEXT ENTRY
         CLC   0(1,R3),FMT7       IS IT A FMT7?
         BE    PRTB0020           YES, BRANCH TO POINT TO NEXT ENTRY
         CLC   0(1,R3),FMT9       IS IT A FMT9?
         BE    PRTB0020           YES, BRANCH TO POINT TO NEXT ENTRY
* DETERMINE IF END OF DATA WAS REACHED
         CLC   0(1,R3),NODSN      IS THERE '00' IN FIRST BYTE
         BNE   PRTB0010           NO, THEN CONTINUE TO PROCESS DSN
         MVI   SWEOD,EOD          YES, SET SWITCH TO END OF DATA
         B     PRTBEXIT           EXIT OUT OF ROUTINE
PRTB0010 DS    0H                 PROCESS DSN - FORMAT
         MVC   DSNMSG(44),0(R3)   MOVE DSN TO PRINT LINE
* PROCESS / FORMAT CCHHR
         MVC   CCHHRS(5),0(R4)    MOVE CCHHR TO CCHHRS START VARIABLE
         UNPK  CCHHRM(L'CCHHRM+1),CCHHRS(6)  UNPACK CCHHR
         TR    CCHHRM,TCHAR1      CONVERT TO PRINTABLE HEX
         PUT   OUTFILE,MSG1 PRINT THE DSN LINE
PRTB0020 DS    0H                 INCREMENT COUNTER FILEEDIT
         LA    R2,TBLLNG(R2)      POINT TO NEXT TABLE ENTRY
         BCT   R10,PRTB0000       BRANCH TO PRTB0000 IF C(R10) GT ZERO
PRTBEXIT DS    0H                 EXIT FROM PRTBRTN
         L     R14,PRTBSAVE       LOAD C(PRTBSAVE) INTO R14
         BR    R14                EXIT
*
*********************************************************************
*                      WORKING  STORAGE                             *
*********************************************************************
*
         DS    0D
         DC    CL36'SEQXMP2-WORKING STORAGE BEGINS HERE'
*
*********************************************************************
*                         EQUATES                                   *
*********************************************************************
*
EABN100  EQU   100                USER ABEND CODE 100-VTOC OPEN ERROR
EABN101  EQU   101                USER ABEND CODE 101-OUTDD OPEN ERROR
EABN102  EQU   102                USER ABEND CODE 102-OUTDD CLOSE ERROR
R0       EQU   0
R1       EQU   1
RDEB     EQU   1                  REG1 FOR DEB ADDRESS
R2       EQU   2
R3       EQU   3
R4       EQU   4
RUCB     EQU   4                  REG4 FOR UCB ADDRESS
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
*
*********************************************************************
*                        SAVE AREAS                                 *
*********************************************************************
*
SAVE     DC    18F'0'              MAIN PROGRAM SAVE AREA
OPENSAVE DC    F'0'                OPEN FILES ROUTINE SAVE AREA
CLOSSAVE DC    F'0'                CLOSE FILES ROUTINE SAVE AREA
ABENSAVE DC    F'0'                ABEND ROUTINE SAVE AREA
IDVLSAVE DC    F'0'                IDENTIFY VOLUME ROUTINE SAVE AREA
LDTBSAVE DC    F'0'                LOAD TABLE ROUTINE SAVE AREA
INITSAVE DC    F'0'                INIT BUFFER ROUTINE SAVE AREA
PRTBSAVE DC    F'0'                PROCESS TABLE ROUTINE SAVE AREA
CVAFSAVE DC    F'0'                CVAFSEQ REQUEST ROUTINE SAVE AREA
LODCSAVE DC    F'0'                LOAD CCHHR ROUTINE SAVE AREA
*
*********************************************************************
*                        CONSTANTS                                  *
```

```
*********************************************************************
*
RETCODE  DC    F'0'               RETURN CODE SAVE FIELD
RCODE00  DC    H'0'               RETURN CODE 0 - HALFWORD
RCODE04  DC    H'4'               RETURN CODE 4 - HALFWORD
RCODE08  DC    H'8'               RETURN CODE 8 - HALFWORD
RCODE12  DC    H'12'              RETURN CODE 12 - HALFWORD
RCODE16  DC    H'16'              RETURN CODE 16 - HALFWORD
CCHHR0   DC    XL5'0000000000'    INIT WITH ZERO
FMT4     DC    XL1'04'            FMT4?
FMT5     DC    XL1'05'            FMT5?
FMT7     DC    XL1'07'            FMT7? (ONLY CERTAIN DEVICE TYPES)
FMT9     DC    XL1'09'            FMT9? (ONLY CERTAIN DEVICE TYPES)
NODSN    DC    XL1'00'            END OF DATA?
BLNKLNE  DC    CL133' '
STRTMSG  DC    CL133' SEQXMP2 START OF OUTPUT MESSAGES               '
ENDMSG   DC    CL133' SEQXMP2 END OF OUTPUT MESSAGES                 '
RC00MSG  DC    CL133' RC00 RETURNED FROM SEQXMP2                      '
RC04MSG  DC    CL133' RC04 RETURNED FROM SEQXMP2                      '
RC08MSG  DC    CL133' RC08 RETURNED FROM SEQXMP2                      '
RC12MSG  DC    CL133' RC12 RETURNED FROM SEQXMP2                      '
RC16MSG  DC    CL133' RC16 RETURNED FROM SEQXMP2                      '
RCERMSG  DC    CL133' UNEXPECTED RC (??) RETURNED FROM SEQXMP2        '
STAT1MSG DC    CL133' CVSTAT CODE STAT032 ENCOUNTERED                 '
STAT2MSG DC    CL133' UNEXPECTED CVSTAT CODE RETURNED FROM SEQXMP2    '
EODMSG   DC    CL133' END OF DATA REACHED - ALL DATA SETS PROCESSED   '
MSG1     DS    0CL133
         DC    CL6' DSN: '
DSNMSG   DS    CL44' '
         DC    CL7'CCHHR: '
CCHHRM   DS    CL10' '
         DC    CL66' '
*********************************************************************
*                         WORK AREAS                                *
*********************************************************************
*
BUFLST   DS    0F                 BUFFER LIST WORK AREA
BUFLISTH DC    2F'0'              BUFFER LIST HEADER
BUFLISTE DC    15F'0'             5 BUFFER LIST ENTRIES
*
UCBADD   DC    F'0'               UCB ADDRESS SAVE AREA
DEBADD   DC    F'0'               DEB ADDRESS SAVE AREA
COUNT    DC    F'0'               TABLE COUNTER
         DS    D
DSCB01   DS    XL140              DSCB01 BUFFER AREA
DSCB02   DS    XL140              DSCB02 BUFFER AREA
DSCB03   DS    XL140              DSCB03 BUFFER AREA
DSCB04   DS    XL140              DSCB04 BUFFER AREA
DSCB05   DS    XL140              DSCB05 BUFFER AREA
*
CCHHRS   DC    XL5'0000000000'    STARTING CCHHR
*
*********************************************************************
*                         PRINT LINES                               *
*********************************************************************
*
         DS    0D
PDETLINE DS    0CL133             DETAIL LINE
         DC    CL133' '
EPDETLEN EQU   *-PDETLINE         LENGTH OF DETAIL LINE
*
*********************************************************************
*                   DCB - OUTPUT FILE (OUTFILE)                     *
*********************************************************************
*
OUTFILE  DCB   DDNAME=OUTDD,                                        X
               DSORG=PS,                                            X
               RECFM=FBA,                                           X
               LRECL=133,                                           X
               MACRF=PM
*
*********************************************************************
*                    VTOC DCB AREA                                  *
*********************************************************************
*
VTOCDCB  DCB   DDNAME=CVAFDD,MACRF=E,EXLST=XLST1,DSORG=PS,DCBE=VTOCDCBE
XLST1    DC    X'87'
         DC    AL3(JFCB1)
JFCB1    DS    0CL176
TESTNAME DS    CL44
         DS    CL8
         DS    BL1
```

```
         DS    CL123
VTOCDCBE DCBE  EADSCB=OK
*
*********************************************************************
*                     TABLES                                        *
*********************************************************************
*
TABLE    DC    0H                 START OF TABLE DSCB ADDR / CCHHR ADDR
*
TDSCB01  DS    F'0'               DSCB01 ADDRESS (140 BYTE DSCB)
TCHR01   DS    F'0'               CCHHR ADDRESS FOR DSCB01 - RETURNED
*
TBLLNG   EQU   *-TABLE            LENGTH OF TABLE ENTRY
*
TDSCB02  DS    F'0'               DSCB02 ADDRESS (140 BYTE DSCB)
TCHR02   DS    F'0'               CCHHR ADDRESS FOR DSCB02 - RETURNED
*
TDSCB03  DS    F'0'               DSCB03 ADDRESS (140 BYTE DSCB)
TCHR03   DS    F'0'               CCHHR ADDRESS FOR DSCB03 - RETURNED
*
TDSCB04  DS    F'0'               DSCB04 ADDRESS (140 BYTE DSCB)
TCHR04   DS    F'0'               CCHHR ADDRESS FOR DSCB04 - RETURNED
*
TDSCB05  DS    F'0'               DSCB05 ADDRESS (140 BYTE DSCB)
TCHR05   DS    F'0'               CCHHR ADDRESS FOR DSCB05 - RETURNED
*
TBLNBR   EQU   (*-TABLE)/TBLLNG   NBR OF TABLE ENTRIES
*
*********************************************************************
*                       TABLES (CONT)                               *
*********************************************************************
*
TCHAR1   EQU   *-C'0'             TABLE TO TRANSLATE TO PRINTABLE HEX
         DC    C'0123456789ABCDEF'
*
*********************************************************************
*                      SWITCHES                                     *
*********************************************************************
SWEOD    DC    XL1'00'            SWITCH - END OF DATA ?
EOD      EQU   X'FF'              END OF DATA DETECTED
NOEOD    EQU   X'00'              END OF DATA NOT DETECTED
*
*********************************************************************
*                           DSECTS                                  *
*********************************************************************
TBLMAP   DSECT                    DUMMY CONTROL SECTION FOR TABLE MAP
DSCBA    DS    F                  DSCB ADDRESS ENTRY
CCHHRA   DS    F                  CCHHR ADDRESS ENTRY
*
*********************************************************************
*                         MACROS / INCLUDES                         *
*********************************************************************
         DCBD DSORG=XE,DEVD=DA    MAP OF DCB
         IEZDEB                   MAP OF DEB
         ICVAFBFL                 BUFFER LIST WITH ONE ENTRY
DSCB     DSECT
         IECSDSL1 (1)             FORMAT 1 DSCB
DSCBLTH  EQU   *-IECSDSL1          LENGTH OF DSCB
*
*********************************************************************
*     CVAF PARAMETER LISTS                                          *
*********************************************************************
*
SEQXMP2  CSECT
CVPL     CVAFSEQ ACCESS=GT,       CVAFSEQ MACRO REQUEST               X
               BUFLIST=BUFLISTH,  ADDRESS OF BUFFER LIST              X
               MF=L
         ORG   CVPL
CVPLMAP  ICVAFPL DSECT=NO         CVAF PARM LIST MAP
*
*
         END   SEQXMP2            END OF SEQXMP2
```

CVAFTST Macro Overview and Specification

The CVAFTST macro determines whether the system supports an indexed VTOC, and, if it does, whether the VTOC on the unit whose UCB is supplied is indexed or nonindexed.

When you issue CVAFTST, register 13 must contain the address of a standard 18 word save area.

You will get a return code of 12 if CVAFTST cannot determine whether an indexed or nonindexed VTOC is on the unit's volume. You should not receive a return code of 12 from CVAFTST if you have opened a data set (including the VTOC) on the volume.

You need no authorization to issue the CVAFTST macro.

See "CVAFDSM Macro Overview and Specification" on page 88 for an example of using the CVAFTST macro with the CVAFDSM macro.

The format of the CVAFTST macro is:

UCB: Specify the VTOC to Be Tested

UCB=(*reg*)

Supplies the address of the UCB for the volume whose VTOC is to be tested. The UCB address can be for a captured UCB, or for an actual UCB above or below the 16 MB line. If your program is in 31-bit mode, this address must be in 31-bit address; the high order byte is part of the address.

Recommendation: Code the address of the UCB parameter as register (2-12). Coding an RX-type address gives unpredictable results.

The CVAFTST macro accepts the address of a UCB or UCB copy. Unauthorized programs can get a copy of the UCB by using the UCBSCAN macro and specifying the COPY, UCBAREA, CMXTAREA, and DCEAREA keywords. The UCB copy and common extension copy must be below the 16 MB line and on a word boundary. Data accessed with DCEAREA can be above the 16 MB line. Refer to *z/OS HCD Planning* for details.

Return Codes from CVAFTST

On return from CVAF, register 15 contains one of the following return codes:

Return Code	Meaning
0 (X'00')	The system does not support an indexed VTOC. The volume should be considered to have a nonindexed VTOC. The UCB was not inspected to determine its validity or status.
4 (X'04')	The system supports an indexed VTOC, but the volume has a nonindexed VTOC.
8 (X'08')	The system supports an indexed VTOC and the volume has an indexed VTOC.
12 (X'0C')	The system supports an indexed VTOC, but the volume is not mounted or the VIB is not initialized for it; thus, the status (indexed or nonindexed) of the VTOC cannot be determined.
16 (X'10')	The system supports an indexed VTOC, but the unit is not a DASD or has a VIO UCB, or the UCB address is not valid. The address of a UCB copy is not valid without a CMXTAREA and a DCEAREA.

VTOC Index Error Message and Associated Codes

Error Message

When CVAF finds an error in a VTOC index, it issues the following message:

IEC606I VTOC INDEX DISABLED ON *dev,volser,code,*[*rba*[*,secno,offset*]]

In addition, CVAF puts a return code in the CVSTAT field of the CVPL.

Explanation

The Common VTOC Access Facility (CVAF) detected a VTOC index error on the device *dev* with volume serial number *volser*. A number that represents the kind of VTOC index error is provided in the *code* field. The RBA of the VIR in the VTOC index that contains the structure error indicated by *code* is provided in the *rba* field. If the VIR is a VIER, the section number in the VIER containing the VTOC index entry is supplied in the *secno* field, and the offset into the section of that VTOC index entry is supplied in the *offset* field.

System Action

The VTOC index is disabled. The VTOC will be converted to nonindexed format when DADSM next allocates space on the volume. A system dump is written to the SYS1.DUMP data set, and an entry is made in the SYS1.LOGREC data set. The message IEC604I (which indicates that the VTOC convert routines have been used) will be issued later.

Programmer Response

Examine the system dump and a print of the VTOC index, and use the information in message IEC606I to determine the cause of the VTOC index structure error.

Routing and Descriptor Codes

The routing codes are 4 (direct access pool) and 10 (system/error maintenance), and the descriptor code is 4 (system status).

Codes Put in the CVSTAT Field

If you are diagnosing an error and require a description of the CVSTAT field codes, see *z/OS DFSMSdfp Diagnosis*.

VTOC Error Responses

The following actions are taken if an error occurs in the VTOC:

- If an index structure error is detected and if the address space is enqueued on the VTOC, then DADSM or CVAF disables the VTOC index. The indexed VTOC bit is zeroed in the format-4 DSCB, a software error record is written to SYS1.LOGREC, and a system dump is taken at the next call to DADSM create or extend. The VTOC is converted to a nonindexed format at the next DADSM create or extend call.
- If a program check, machine check, or other error occurs while using a VTOC access macro, a SYS1.LOGREC record is written, and a system dump is taken.
- An error code is put in the CVSTAT field of the CVPL. The values and explanations of these error codes are listed in "VTOC Index Error Message and Associated Codes" on page 127.

Recovering from System or User Errors

Because an unauthorized user cannot modify a VTOC, neither the VTOC nor the VTOC index need be recovered from an error caused by an unauthorized user.

A system error can affect a VTOC and VTOC index by interrupting DADSM while it is updating, leaving the VTOC or the VTOC index (or both) in a partially-updated state. Both the VTOC and the VTOC index allow DADSM to recover from such an interruption.

For a nonindexed VTOC (or a VTOC with an index that has been disabled), a subsequent call to DADSM create or extend, causes VTOC convert routines to reestablish the free space DSCB chain.

For an indexed VTOC, a subsequent call to any DADSM function causes the recovery of the previous interrupt (either by backing out or completing the interrupted function).

GTF Trace

A trace function exists to trace all CVAF calls for VTOC index output I/O, all VTOC output I/O, and all VTOC index and space map modifications. For information on this function, see *z/OS DFSMSdfp Diagnosis*.

VTOC and VTOC Index Listings

You can obtain dump, formatted, or abridged listings of the VTOC and the VTOC index by using the LISTVTOC command of the IEHLIST utility program. The DFSMSdss print command also provides VTOC and index listing options. The ISMF data set application displays information about the data sets represented in a VTOC.

Chapter 2. Managing the Volume Table of Contents

This information covers the programs and procedures used to create and manage a volume table of contents (VTOC).

Creating the VTOC and VTOC Index

To prepare a volume for activity by initializing it, use the Device Support Facilities (ICKDSF) utility to build the VTOC. You can create a VTOC index when initializing a volume by using the ICKDSF INIT command and specifying the INDEX keyword.

To convert a non-indexed VTOC to an indexed VTOC, use the BUILDIX command with the IXVTOC keyword. The reverse operation can be performed by using the BUILDIX command and specifying the OSVTOC keyword.

To refresh a volume VTOC and INDEX in its current format, use the ICKDSF command REFORMAT with the RVTOC keyword. To optionally extend the VTOC and INDEX, use the ICKDSF command REFORMAT with the EXTVTOC and EXTINDEX keywords.

See *Device Support Facilities (ICKDSF) User's Guide and Reference* for details.

Protecting the VTOC and VTOC Index

Use the following methods in order to protect the VTOC and VTOC index:

RACF®

You can protect the VTOC and VTOC index using the Resource Access Control Facility (RACF), a component of the Security Server for z/OS, by defining the volume serial entity under the RACF DASDVOL class. For you to modify a protected VTOC and VTOC index, programs must be authorized at the following levels:

- UPDATE level to open for output processing a VTOC or any data set with a name beginning with SYS1.VTOCIX.
- ALTER level to allocate, rename, or scratch any data set with a name beginning with SYS1.VTOCIX or to rename a data set to a name beginning with SYS1.VTOCIX.

Neither the VTOC nor the VTOC index is protected from being opened for input processing by the DASDVOL/volume serial entity. Neither the VTOC nor the VTOC index can be protected through the RACF DATASET class. For additional information on using RACF, see *z/OS Security Server RACF Security Administrator's Guide*.

APF

The authorized program facility (APF) must authorize a program in order for you to:

- Open a VTOC for output processing
- Open for output processing, allocate, rename, or scratch any data set whose name begins with SYS1.VTOCIX
- Rename a data set to a name that begins with SYS1.VTOCIX.

For additional information on using APF, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

Password Protection

The VTOC index data set can be password protected. Protection is the same as for a password-protected data set. Password checking is bypassed if the volume containing the VTOC index is protected by RACF with the DASDVOL class. For additional information, see Chapter 6, “Using Password Protected Data Sets,” on page 237.

Copying/Restoring/Initializing the VTOC

The following topics discuss VTOC considerations when updating volumes:

- “Volumes Containing a Nonindexed VTOC” on page 132
- “Volumes Containing an Indexed VTOC” on page 132

Volumes Containing a Nonindexed VTOC

When updating volumes containing a non-indexed VTOC keep the following considerations in mind:

- *Restoring a Volume from a Dump Tape*: There are no operational requirements if you change the volume serial number or do a partial restore that does not modify the VTOC. If you do a restore and change the VTOC size without changing the volume serial number, the system can automatically update the UCB with the new VTOC location and new volume serial number following a Restore or Copy Volume operation if you have the REFUCB function enabled. To enable this function, do one of the following:
  - Specifiy ENABLE(REFUCB) in the DEVSUPxx parmlib member
  - Issue the MODIFY command as follows:

    ```
    F DEVMAN,ENABLE(REFUCB)
    ```

  If you do not enable REFUCB, you cannot restore on a volume with an indexed VTOC. In that case, you must vary the volume offline after it is restored.
- *Copying a Volume*: There are no operational requirements if you change the volume serial number or do not modify the VTOC of the receiving volume. If you do a copy and change the VTOC size without changing the volume serial number, you must vary the volume offline after it is copied. Do not attempt to copy from a volume with an indexed VTOC.
- *Shared DASD Considerations*:In shared DASD environments, if the VTOC index is relocated or the volume is changed from indexed VTOC to nonindexed VTOC or from nonindexed VTOC to indexed VTOC, it generally is advisable to vary the device offline to the sharing system or systems before beginning the operation. However it is not necessary to vary the volume offline on other systems in the same sysplex z/OS Version 1 Release 5 or higher using ICKDSF release 17.

Volumes Containing an Indexed VTOC

Use Device Support Facilities (ICKDSF) to convert a VTOC to a non-indexed format to update the volume. If you do not, keep the following considerations in mind:

Initializing a Volume: If you do not change the volume serial number, you must vary the volume offline before starting the job.

Restoring a Volume from a Dump Tape: There are no operational requirements if you change the volume serial number or do a partial restore that does not modify the VTOC or VTOC index. If you do a restore, and modify the VTOC or VTOC index without changing the volume serial number, you must vary the volume offline after it is restored.

Copying a Volume: There are no operational requirements if you change the volume serial number of the receiving volume or do a partial dump without modifying the VTOC or VTOC index. If you modify the VTOC or VTOC index without changing the volume serial number, you must vary the receiving volume offline after it is copied.

Deleting a Data Set from the VTOC

You can use the SCRATCH and CAMLST macro to delete a non-VSAM data set or a temporary VSAM data set. SCRATCH processing makes the space occupied by the data set available for reallocation. This process does not automatically erase data from the disk. See “Erasing Sensitive Data” on page 133 for further information.

Specifying the Volumes Affected

When deleting a data set, build a volume list in virtual storage. The volume list consists of an entry for each volume on which the data set resides. If you are deleting an SMS-managed data set, specify at least one SMS-managed volume in the list. The first two bytes of the list indicate the number of entries in the list. Each 12-byte entry consists of a 4-byte device code (the UCBTYP field from the volume's UCB), a 6-byte volume serial number, and 2 bytes of scratch status information consisting of a secondary status code and a status code, both of which must be initialized to zero.

Volumes are processed according to their order in the volume list. If a volume is not mounted, a message is issued to the operator requesting that the volume be mounted. This only occurs when you indicate the direct access device on which unmounted volumes are to be mounted by loading register 0 with the address of the UCB of the device. (The device must be allocated to your job.) **If you do not load register 0 with a UCB address, its contents must be zero**, and at least one of the volumes in the volume list must be mounted before the SCRATCH macro instruction is issued. Use the address of a UCB, not a UCB copy, in register 0 with this macro. 31-bit programs must pass a clean UCB addr on RENAME, when applicable.

If the requested volume cannot be mounted, the operator replies by indicating that the request cannot be fulfilled. A status code is then set in the last byte of the volume list entry (the second byte of the scratch status code) for the unavailable volume, and the next volume in the volume list is processed.

Erasing Sensitive Data

You should erase data sets that contain sensitive data by overwriting them with zeros before their space is made available. This can either be done before issuing the SCRATCH macro, or be requested in scratch processing by performing one of the following:

- Providing an associated RACF profile ERASE attribute
- Activating bit 21 (X'00 00 04 00') of the SCRATCH parameter list. See Table 22 on page 135.

Authorized callers of SCRATCH can prevent erasure of the data by setting bit 22 to 1, which overrides the RACF profile ERASE attribute.

System-Managed-Storage Considerations

SMS screens all data set SCRATCH requests. If the volumes in your volume list are SMS managed, SMS does a catalog LOCATE to determine the actual volume serial numbers, and deletes the data set from all volumes on which it resides. SMS coordinates the required changes to the VTOC, the VTOC index, and the catalog.

If DADSM encounters a processing error when SMS is active and all the volumes in your list are SMS managed, SMS determines the volume on which the failure occurred. The first entry in your list will be overlaid with the entry for the volume on which the request failed.

You might find that a volume indicated as being in error was not specified in the volume list your program provided. This occurs if the volumes in your list are different from the volumes in the data set's catalog entry.

If SMS is not active, you cannot delete SMS-managed data sets.

You can delete SMS-managed VSAM data sets using the access method services DELETE command. See *z/OS DFSMS Access Method Services Commands* for further information.

General Considerations and Restrictions

A data set cannot be deleted if the expiration date in the format-1 DSCB has not passed unless you override the expiration date. You can request SCRATCH to ignore the expiration date by specifying the OVRD option in the CAMLST macro instruction. SCRATCH processing supports three never-scratch dates. To prevent a data set from being scratched, specify one of the following expiration dates:

1999.365
1999.366
1999.999

To delete a virtual input/output (VIO) data set, the data set must be allocated for use by your job step.

You cannot use the SCRATCH macro with either a SYSIN or SYSOUT data set or an z/OS UNIX file. You will receive unpredictable results if you use SCRATCH for z/OS UNIX files.

If you attempt to delete a password-protected data set that is not also RACF protected, the operating system issues message IEC301A to the operator at the console, or the terminal operator of a TSO console, to enter the password. The data set will be scratched if the password supplied is associated with a WRITE protection mode indicator. The protection mode indicator is described in Chapter 6, “Using Password Protected Data Sets,” on page 237.

If a data set is RACF-defined (indicated in its format-1 DSCB or described by a RACF profile) or the volume upon which it resides is RACF-defined, you can scratch the data set only if you have ALTER access authority to either the data set/volume serial in the DATASET class or to the volume serial in the DASDVOL class.

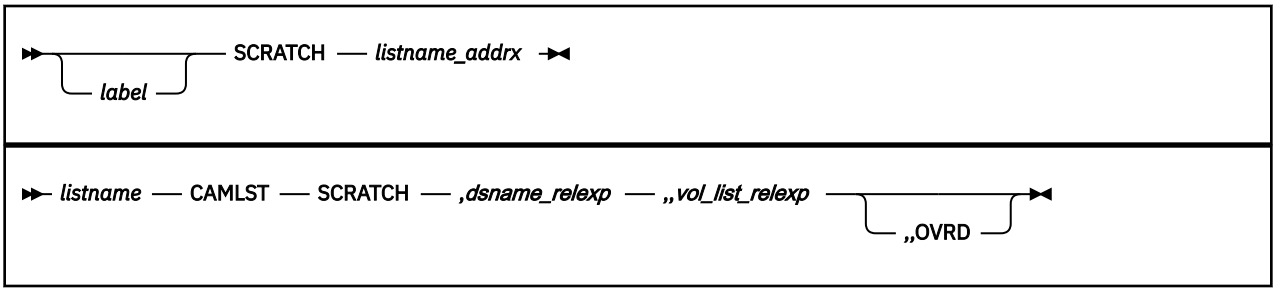
Requirement: For an SMS-managed non-VSAM data set, you need RACF authority to the data set or to the catalog to delete it.

For a non-VSAM data set that is not SMS-managed, DADSM invokes RACF to verify authorization. If you have ALTER access authority to the data set/volume serial in the DATASET class, DADSM deletes the data set from the volume. If you have ALTER authority to the data set/volume serial in the DATASET class, or UPDATE access authority to the catalog/volume serial in the DATASET class, you can delete the catalog entry.

Use the STOW macro to delete or rename a member of a PDS or PDSE. STOW is described in *z/OS DFSMS Macro Instructions for Data Sets* and *z/OS DFSMS Using Data Sets*. You can also use the IEHPROGM utility to delete a member (see *z/OS DFSMSdfp Utilities*).

SCRATCH and CAMLST Macro Specification

The format of the SCRATCH and CAMLST macros is:



listname_addrx
Points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

SCRATCH
Code this operand as shown.

dsname_relexp
Specifies the virtual storage location of a fully-qualified data set name. The area that contains the name must be 44 bytes long.

vol list_relexp
Specifies the virtual storage location of an area that contains a volume list. The area must begin on a halfword boundary.

OVRD
When coded as shown, specifies that the expiration date in the DSCB should be ignored.

Example

In the following example, data set A.B.C is deleted from two volumes. The expiration date in the identifier (format-1) DSCB is ignored.

```
          SR       0,0              SHOW NO UCB IS SUPPLIED
          SCRATCH  DELABC           DELETE DATA SET A.B.C
*                                   FROM TWO VOLUMES,
*                                   IGNORING EXPIRATION
*                                   DATE IN THE DSCB

DELABC    CAMLST   SCRATCH,DSABC,,VOLIST,,OVRD
DSABC     DC       CL44'A.B.C'      DATA SET NAME
VOLIST    DC       H'2'             NUMBER OF VOLUMES
          DC       X'3030200F'      3390 DISK DEVICE CODE
          DC       CL6'000017'      VOLUME SERIAL NO.
          DC       H'0'             SCRATCH STATUS CODE
          DC       X'3030200F'      3390 DISK DEVICE CODE
          DC       CL6'000018'      VOLUME SERIAL NO.
          DC       H'0'             SCRATCH STATUS CODE
```

Recommendation: Check the return codes and SCRATCH status codes.

The SCRATCH macro instruction points to the CAMLST macro instruction. The SCRATCH operand specifies that a data set be deleted. DSABC specifies the virtual storage location of a 44-byte area containing the fully-qualified name of the data set to be deleted. VOLIST specifies the virtual storage location of the volume list you have built. OVRD specifies that the expiration date in the DSCB of the data set to be deleted should be ignored.

SCRATCH Parameter List

The CAMLST macro generates the SCRATCH parameter list, but your code can set several options that CAMLST does not support. See Table 22 on page 135.

Table 22. SCRATCH Parameter List

Offset	Length or Bit Pattern	Description
0(0)	1	Flags. Always X'41'.
1(1)	1	Flags.
	.1..	Do not delete the Resource Access Control Facility (RACF) profile. Has an effect only if JSCBPASS is on. JSCBPASS is on if the program properties table gives authority to bypass security. See the NOPASS option on the primary POI task (PPT) statement in the SCHED*xx* member of SYS1.PARMLIB as described in *z/OS MVS Initialization and Tuning Reference*.
	x.xx xxxx	Reserved.
2(2)	1	Flags.

Table 22. SCRATCH Parameter List (continued)		
Offset	**Length or Bit Pattern**	**Description**
	1...	SYSZTIOT already is enqueued. SCRATCH bypasses enqueueing on SYSZTIOT if this bit is on and the caller is APF-authorized or is running in a system key or supervisor state. If SYSZTIOT is not already enqueued, system damage might result.
	.10. ...0	Always set for SCRATCH.
	...1	OVRD coded on CAMLST.
	.... 1...	Bypass RACF profile checking. SCRATCH bypasses RACF profile checking if this bit is on and the caller is APF-authorized or is running in a system key or supervisor state.
	.... .1..	Erase all allocated space for the data set.
	.... ..1.	Do not erase allocated space. SCRATCH bypasses space erasure if this bit is on and the caller is APF-authorized or is running in a system key or supervisor state.
3(3)	1	Reserved.
4(4)	4	Address of 44-byte data set name.
8(8)	4	Reserved.
12(C)	4	Address of volume list.

Return Codes from SCRATCH

Control returns to the instruction following the instructions generated by the SCRATCH macro. Register 15 contains the SCRATCH return code as shown in Table 23 on page 136.

SCRATCH returns 4 bytes of diagnostic information in register 0. If an error occurs, DADSM issues message IEC614I, consisting of failure-related information including the return code and the 4 bytes of diagnostic information. See *z/OS DFSMSdfp Diagnosis* for a description of this information.

The last two bytes of a volume list entry contain the secondary status code and the scratch status code. The secondary status codes are shown in Table 25 on page 138 and the scratch status codes are shown in Table 24 on page 137. To determine if the data set has been deleted from each volume, check the scratch status code. (Even if the scratch status code is zero, the secondary status code might be nonzero for the first entry in the volume list.)

Table 23 on page 136 describes the conditions indicated by the SCRATCH return code.

Table 23. SCRATCH Return Codes	
Return Code	**Description**
000 (X'00')	If the 4 bytes of diagnostic information returned in register 0 are all zeros, the data set was successfully deleted. If they are nonzero, use the SCRATCH diagnostic information tables in *z/OS DFSMSdfp Diagnosis* to determine the failure-related conditions.
004 (X'04')	No volume containing any part of the data set was mounted. The data set might be a VIO data set that was not allocated to your jobstep.
008 (X'08')	An unusual condition was encountered on one or more volumes.

Table 23. SCRATCH Return Codes (continued)

Return Code	Description
012 (X'0C')	One of the following conditions occurred: • The SCRATCH parameter list is not valid. • The volume list is not valid. • At entry to SCRATCH, register 0 was not zero and did not point to a valid UCB. The address must be that of a UCB, not a UCB copy. The SCRATCH status code will not have been set.

Status Codes from SCRATCH

After the SCRATCH macro instruction is executed (for SCRATCH return codes 0, 4, and 8 only), the last byte of each 12-byte entry in the volume list indicates one of the following conditions:

Table 24. SCRATCH Status Codes

Status Code	Meaning
0 (X'00')	The data set has been deleted from this volume.
1 (X'01')	The VTOC of this volume does not contain the format-1 or format-8 DSCB to be deleted.
2 (X'02')	One of the following conditions occurred: • The data set could not be scratched because the correct password was not specified in the two attempts allowed. • The user tried to scratch a VSAM data space or an integrated catalog facility VSAM data set. • The user tried to scratch the VTOC index data set. • An SMS-validation failure occurred. • The verify of the last referenced date failed.
3 (X'03')	The data set was not deleted because either the OVRD option was not specified or the retention cycle had not expired.
4 (X'04')	One of the following conditions occurred: • An invalid format-1 or format-8 DSCB was encountered when processing this volume. • An unexpected CVAF error return code was encountered. • An installation exit rejected the request. • An I/O error occurred while the DASD tracks occupied by the data set were being erased. Either the ERASE option was specified in the scratch parameter list or the ERASE attribute was specified for an RACF-defined data set.
5 (X'05')	It could not be verified that this volume was mounted, nor was there a unit available for mounting the volume.
6 (X'06')	The operator was unable to mount this volume.
7 (X'07')	The data set was not deleted because it was open.
8 (X'08')	The format-1 or format-8 DSCB indicates the data set is defined to RACF, but either you are not authorized to the data set or volume, or the data set is a VSAM data space.
9 (X'09')	The data set is on a read-only volume and cannot be deleted on this volume.

After the SCRATCH macro instruction is executed, the next to last byte of the first entry in the volume list indicates one of the following conditions:

Table 25. Secondary Status Codes

Status Code	Meaning
0 (X'00')	No secondary status for this volume.
128 (X'80')	The data set was RACF protected and the calling program was authorized by the RACF DATASET class to scratch the data set. This means that at least one volume entry was protected.

Renaming a Data Set in the VTOC

You can use the RENAME and CAMLST macro to rename a non-VSAM data set. Rename processing causes the data set name in all format-1 or format-8 DSCBs to be replaced with the new name you supply. The new data set name must conform to standard data set naming conventions.

Specifying the Volumes Affected

When renaming a data set, build a volume list in virtual storage. The volume list consists of an entry for each volume on which the data set resides. If you are renaming an SMS-managed data set, specify at least one SMS-managed volume in the list. The first two bytes of the list indicate the number of entries in the list. Each 12-byte entry consists of a 4-byte device code (the UCBTYP field from the volume's UCB), a 6-byte volume serial number, and a 2-byte rename status code that should be initialized to zero.

Volumes are processed according to their order in the volume list. If a volume is not mounted, a message is issued to the operator requesting that the volume be mounted. This only occurs when you indicate the direct access device on which unmounted volumes are to be mounted by loading register 0 with the address of the UCB of the device. (The device must be allocated to your job.) **If you do not load register 0 with a UCB address, its contents must be zero**, and at least one of the volumes in the volume list must be mounted before the RENAME macro instruction is issued. Use the address of a UCB, not a UCB copy, in register 0 with this macro. 31-bit programs must pass a clean UCB addr on RENAME, when applicable.

If the requested volume cannot be mounted, the operator replies by indicating that the request cannot be fulfilled. A status code is then set in the last byte of the volume list entry (the second byte of the rename status code) for the unavailable volume, and the next volume indicated in the volume list is processed.

System-Managed-Storage Considerations

SMS screens all data set RENAME requests. If the volumes specified in your volume list are SMS managed, SMS does a catalog LOCATE to determine the actual volume serial numbers, and coordinates the required changes to the VTOC, the VTOC index, and the catalog.

If DADSM encounters a processing error while SMS is active and all the volumes in your list are SMS managed, SMS determines the volume on which the failure occurred. The first entry in your list will be overlaid with the entry for the volume on which the request failed.

You might find that a volume indicated as being in error was not specified in the volume list your program provided. This occurs if the volumes in your list are different from the volumes in the data set's catalog entry.

If SMS is not active, you cannot rename SMS-managed data sets.

General Considerations and Restrictions

Multivolume Considerations

To rename a data set that is stored on more than one volume, all volumes must be mounted.

Unrenamable Data Sets and UNIX Files

You cannot use the RENAME macro with either a SYSIN or SYSOUT data set or UNIX file (such as an z/OS UNIX file). You will receive unpredictable results if you use RENAME for UNIX files.

You cannot rename VIO data sets.

Data Set Security

You can rename a RACF-defined data set only if you have ALTER access authority to the data set in the DATASET class.

If you attempt to rename a password-protected data set, the operating system issues message IEC301A asking the operator or TSO operator to verify the password. The data set will be renamed if the password supplied is associated with a WRITE protection mode indicator. The protection mode indicator is described in Chapter 6, “Using Password Protected Data Sets,” on page 237.

Renaming a Data Set That Might be in Use

You can rename a data set that is allocated to the current address space but it cannot be open.

In general, you cannot rename a data set whose name is the same as any data set that is allocated to another address space in the same system or in the scope of the SYSDSN enqueue. The system bypasses this restriction if all of the following are true:

- Your program sets on a certain bit in the CAMLST macro expansion. You can code this instruction: OI listname+2,X'10'.
- You have at least read authority to the RACF facility class named STGADMIN.DPDSRN.*olddsname*, where *olddsname* is up to 23 characters of the existing data set name. You can use a generic class name such as STGADMIN.DPDSRN.SYS2.*. IBM recommends that no one have authority to STGADMIN.DPDSRN.* because it is too broad.
- The data set is not SMS-managed.

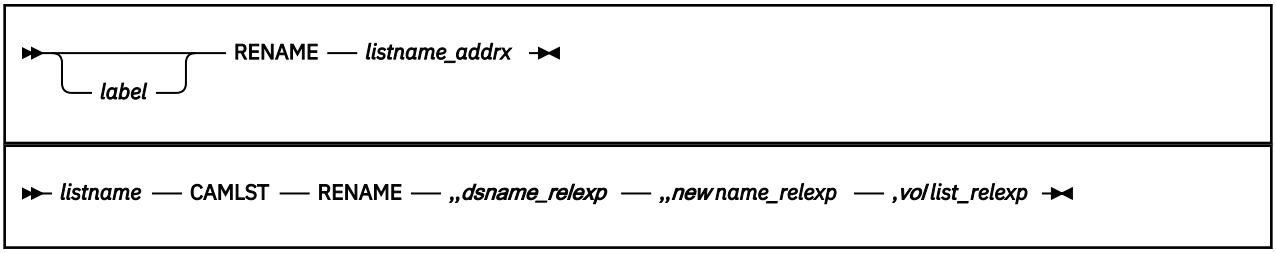
Alternatively, you can use the data set rename option of PDF. If you attempt to rename a non-SMS-managed, non-VSAM data set, the data set name is in use and you have the appropriate RACF facility class authority, then PDF asks whether you wish to proceed because you know that the data set is not actually open. Let the rename proceed only if you know the data set being renamed is not open on any system.

Attention: This option should be used with extreme caution. Very few people should have RACF authority to STGADMIN.DPDSRN.*olddsname*. Do not use this option unless you know the data set is not open on any system. After the data set is renamed, someone could delete it in a different address space. If someone has it open by the old name, new data sets will appear at those places on the disk. This would be a security violation that the system does not detect.

The data set rename function writes a type 18 SMF record to provide information to storage administrators, system programmers, and auditors. The record contains an indicator of whether it was successful due to the use of this duplicate name override function. If you request the option in the CAMLST macro expansion but the data set name is not in use, then the SMF indicator will not be on.

RENAME and CAMLST Macro Specification

The format of the RENAME and CAMLST macros is:



listname_addrx
Points to the parameter list (labeled *listname*) set up by the CAMLST macro instruction.

RENAME
Code this operand as shown.

dsname_relexp
Specifies the virtual storage location of a fully-qualified data set name to be replaced. The area containing the name must be 44 bytes long.

new name_relexp
Specifies the virtual storage location of a fully-qualified data set name that is to be used as the new name. The area containing the name must be 44 bytes long.

vol list_relexp
Specifies the virtual storage location of an area that contains a volume list. The area must begin on a halfword boundary.

Example

In the following example, data set A.B.C is renamed D.E.F. The data set resides on two volumes.

```
         SR      0,0             SET REG 0 TO ZERO
         RENAME  DSABC           CHANGE DATA SET
                                 NAME A.B.C TO D.E.F

DSABC    CAMLST  RENAME,OLDNAME,NEWNAME,VOLIST
OLDNAME  DC      CL44'A.B.C'     OLD DATA SET NAME
NEWNAME  DC      CL44'D.E.F'     NEW DATA SET NAME
VOLIST   DC      H'2'            TWO VOLUMES
         DC      X'3030200F'      3390 DISK DEVICE CODE
         DC      CL6'000017'     VOLUME SERIAL NO.
         DC      H'0'            RENAME STATUS CODE
         DC      X'3030200F'      3390 DISK DEVICE CODE
         DC      CL6'000018'     VOLUME SERIAL NO.
         DC      H'0'            RENAME STATUS CODE
```

Recommendation: Check the return codes and RENAME status codes.

The RENAME macro instruction points to the CAMLST macro instruction. The RENAME operand specifies that a data set be renamed. OLDNAME specifies the virtual storage location of a 44-byte area where you have placed the fully-qualified name of the data set to be renamed. NEWNAME specifies the virtual storage location of a 44-byte area where you have placed the new name of the data set. VOLIST specifies the virtual storage location of the volume list you have built.

RENAME Parameter List

The CAMLST macro generates the RENAME parameter list but your code can set several options that CAMLST does not support. See Table 26 on page 140.

Table 26. RENAME Parameter List generated by CAMLST

Offset	Length or Bit Pattern	Description
0(0)	1	Flags. Always X'C1'
1(1)	1	Flags.
	.1..	Do not define RACF profile.
	..1.	Do not update the changed bit (DS1DSCHA) in the format 1 DSCB.
	x..x xxxx	Reserved.
2(2)	1	Flags.
	..1.	Always set for RENAME.

Table 26. RENAME Parameter List generated by CAMLST (continued)		
Offset	**Length or Bit Pattern**	**Description**
	...1	See "Renaming a Data Set That Might be in Use" on page 139.
	xx.. xxxx	Reserved.
3(3)	1	Reserved.
4(4)	4	Address of 44-byte existing data set name.
8(8)	4	Address of 44-byte new data set name.
12(C)	4	Address of volume list.

Return Codes from RENAME

Control returns to the instruction following the instructions generated by the RENAME macro. Register 15 contains the DADSM return code as shown in Table 27 on page 141.

DADSM RENAME returns 4 bytes of diagnostic information in register 0. If an error occurs, DADSM issues message IEC614I, consisting of failure-related information including the return code and the 4 bytes of diagnostic information. See *z/OS DFSMSdfp Diagnosis* for a description of this information.

Each volume's volume list entry contains the rename status code as shown in Table 28 on page 142. To determine whether the data set has been successfully renamed on each volume, check the rename status code, contained in the last byte of each entry in the volume list.

Table 27 on page 141 describes the conditions indicated by the DADSM return code.

Table 27. DADSM RENAME Return Codes

Return Code	Description
000 (X'00')	If the 4 bytes of diagnostic information returned in register 0 are all zeros, the data set has been successfully renamed. If they are nonzero, use the DADSM RENAME diagnostic information tables in *z/OS DFSMSdfp Diagnosis* to determine the failure-related conditions.
004 (X'04')	No volume containing any part of the data set was mounted. The data set can be a VIO data set but cannot be renamed.
008 (X'08')	An unusual condition was encountered on one or more volumes. The diagnostic information is in register 0. Use the DADSM RENAME diagnostic information tables in *z/OS DFSMSdfp Diagnosis* to determine the failure-related conditions.
012 (X'0C')	One of the following conditions occurred: • The DADSM RENAME parameter list is not valid. • The volume list is not valid. • At entry to RENAME, register 0 was not zero and did not point to a valid UCB. The address must be that of a UCB, not a UCB copy. The RENAME status code will not have been set.

Status Codes from RENAME

After the RENAME macro instruction is executed (for RENAME return codes 0, 4, and 8 only), the last byte of each 12-byte entry in the volume list indicates one of the following conditions described in Table 28 on page 142.

Table 28. RENAME Status Codes

Status Code	Meaning
0 (X'00')	The format-1 DSCB for the data set has been renamed in the VTOC on this volume.
1 (X'01')	The VTOC of this volume does not contain the format-1 or format-8 DSCB of the data set to be renamed.
2 (X'02')	One of the following conditions occurred: • The data set could not be renamed because the data set was password protected and the password was not supplied in the two attempts allowed. • An attempt was made to rename a VSAM data space or an integrated catalog facility VSAM data set. • An attempt was made to rename a VTOC index data set. • An SMS-validation failure occurred.
3 (X'03')	A format-1 or format-8 DSCB containing the new data set name already exists in the VTOC of this volume, or an attempt was made to rename a data set to a name starting with SYS1.VTOCIX.
4 (X'04')	One of the following conditions occurred: • A permanent I/O error occurred while trying to rename the data set on this volume. • An invalid format-1 or format-8 DSCB was encountered while processing this volume. • No space is available in the index VIER for the new name, and no additional VIERs are available.
5 (X'05')	It could not be verified that this volume was mounted nor was a unit available for mounting the volume.
6 (X'06')	The operator was unable to mount this volume.
7 (X'07')	The data set was not renamed, because it was currently open for processing.
8 (X'08')	The data set is defined to RACF, but either you are not authorized to the data set or the data set is defined to RACF on multiple volumes.
9 (X'09')	The data set is on a read-only volume and cannot be renamed on this volume.

Chapter 3. Using Catalog Management Macros

This information covers catalog management macro instructions for compatibility purposes only. Catalog management macro instructions can be used to perform the following functions:

- Retrieve information from an integrated catalog facility catalog.
- Catalog, uncatalog, or re-catalog in an integrated catalog facility catalog the following types of data sets:
  - DASD data sets that are not VSAM or SMS-managed
  - Tape data sets.

Application Program Considerations

A catalog management request can be satisfied in an integrated catalog facility catalog. Consider the following restrictions and limitations in relation to your application programs:

- A catalog management request is expressed in a parameter list pointed to by register 1. Generate the parameter list with a CAMLST macro. The CAMLST and its associated fields must not be located in read-only storage.
- Register 15 contains the return code. These return codes are explained in the sections below.

Catalog Search Order

Catalogs are searched for entries using the following methods:

1. If a catalog is specified in a macro, only that catalog is searched.
2. If the entry is identified with a qualified entry name and its first qualifier is the same as the name or alias of a user catalog, the user catalog is searched. When the entry is found, no other catalog is searched.
3. The master catalog is searched.

See *z/OS DFSMS Access Method Services Commands* for more detailed information.

Retrieving Information from a Catalog

To read an entry from a catalog, use the LOCATE and CAMLST macro instructions. You can specify the entry to be read into your output area using the following information:

- The fully- or partially-qualified name of a data set
- The relative block address of the block containing the entry.

If you specify a fully-qualified data set name, a list of volumes on which the data set resides is read into your output area. This volume list always begins with a 2-byte entry indicating the number of volumes in the list.

Restriction: When CAMLST is used to locate a data set that is over 20 volumes in length, only information from the first 20 volumes is returned. If you need to retrieve data from more than 20 volumes, use IGGCSI (Catalog Search Interface). See (link to pertinent section in Man Cat).

For the Catalog interface, a fully-qualified name is one which represents a single data set. A partially-qualified name is one which may contain multiple qualifiers, but does not specify a full data set name.

For example, if LEVEL1.LEVEL2.LEVEL3.LEVEL4 is a data set, then LEVEL1.LEVEL2.LEVEL3.LEVEL4 is a fully-qualified name. The following data set would be considered partially-qualified names:

LEVEL1.LEVEL2.LEVEL3

LEVEL1.LEVEL2
LEVEL1

Restriction: For the catalog interface, you cannot specify an asterisk (*) or an ampersand (&) to specify a partially-qualified data set name.

See *z/OS DFSMSdfp Diagnosis* for descriptions of the contents of the volume control block and the other catalog data areas.

See "Return Codes from LOCATE" on page 147 for a description of the LOCATE return codes.

Retrieving Information by Data Set Name (LOCATE and CAMLST NAME)

Specifying a data set name returns a volume list in your output area. A volume list consists of an entry for each volume on which part of the data set resides. For each volume, the list contains the volume serial number, device type, and file sequence number. Volumes are divided by whether they fall within the minimum unit count, or outside of it. Volumes within and without are in descending binary order by device type, except for reusable KSDS. For a reusable KSDS, the volume serial numbers are returned in the order: first index, first data, remaining index, remaining data.

A volume list begins with a 2-byte field containing the number of volumes in the list. The count field is followed by a variable number of 12-byte entries. Each 12-byte entry consists of a 4-byte device code, a 6-byte volume serial number, and a 2-byte volume sequence number. As many as 20 of these 12-byte entries can be built in your output area. LOCATE can return bytes 252 - 254 of your area containing the relative track address of the DSCB on the first or only volume for that data set. Otherwise, these bytes are zero. Bytes 242 to 251 are reserved, byte 255 contains zeros, and bytes 256 to 264 are also reserved and not intended as a programming interface.

A CAMLST LOCATE on a VSAM cluster returns the volumes for all components. The format of this volume list is as described previously. The output is translated into the format described in the preceding two paragraphs before returning to the caller. The original VSAM or integrated catalog facility return code is saved in register 0.

The macro format for the LOCATE and CAMLST NAME is:

listname_addrx
Points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

NAME
To retrieve information from a catalog by name, code this operand as shown.

dsname_relexp
Specifies the virtual storage location of a fully-qualified data set name. The area that contains the name must be 44 bytes long. The name can be defined by a C-type define constant (DC) instruction.

area_relexp
Specifies the virtual storage location of your 265-byte output area, that you must define. The output area must begin on a doubleword boundary.

Example

In the following example, the catalog entry containing a list of the volumes on which data set A.B resides, is read into virtual storage.

```
          LOCATE    INDAB              READ CATALOG ENTRY FOR DATA SET A.B
*                                      INTO VIRTUAL STORAGE AREA NAMED LOCAREA.
*                                      LOCAREA MAY ALSO CONTAIN A 3-BYTE
*                                      TTR OR A 6-BYTE SERIAL NUMBER
         Check Return Codes

INDAB     CAMLST    NAME,AB,,LOCAREA
AB        DC        CL44'A.B'
LOCAREA   DS        0D
          DS        265C
```

The LOCATE macro instruction points to the CAMLST macro instruction. NAME, the first operand of CAMLST, specifies that the system is to search for a catalog entry using the name of a data set. AB, the second operand, specifies the virtual storage location of the fully-qualified data set name LOCAREA, the fourth operand, specifies a 265-byte area you have reserved in virtual storage.

After these macro instructions execute, the 265-byte area contains a volume list or a volume control block for the data set A.B. If the entry has been located and read successfully, register 15 contains zeros. Otherwise, register 15 contains a return code (see “Return Codes from LOCATE” on page 147).

Retrieving Information by Generation Data Set Name (LOCATE and CAMLST NAME)

Specify the name of a generation data set using the fully-qualified generation index name and the relative generation number of the data set. The value of a relative generation number reflects the position of a data set in a generation data group. The following values can be used to identify a data set in a generation data group:

- Zero—specifies the latest data set (highest generation number) cataloged in a generation data group.
- Negative number—specifies a data set cataloged before the latest data set.

  Rule: If DISP (disposition) is DELETE to make room for other data sets and no generation data group exists, the job will complete indicating a deleted generation name (G0000V00). If a generation data group exists but is not in the range specified for deletion, the step will fail.
- Positive number—specifies a data set not yet cataloged in the generation data group.

Using zero or a negative number as the relative generation number places a volume list (or a volume control block) in your output area and replaces the relative generation number with the absolute generation name.

Using a positive number as the relative generation number creates an absolute generation name and replaces the relative generation number. Because there are no entries in the catalog, zeros are read into the first 256 bytes of your output area.

The format for the LOCATE and CAMLST NAME macros is:

list_addrx
Points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

NAME
To read a block from the catalog by generation data set name, code this operand as shown.

dsname_relexp
Specifies the virtual storage location of the name of the generation index and the relative generation number. The area that contains these must be 44 bytes long.

area_relexp
Specifies the virtual storage location of your 265-byte output area, which you must define. The output area must begin on a doubleword boundary. The output area will contain a volume list that is built from the catalog. If the data set resides on one volume, bytes 252 - 254 can contain the relative track address of the DSCB. This address is relative to the beginning of the volume.

Example

In the following example, the list of volumes containing generation data set A.PAY(-3) is read into virtual storage.

```
           LOCATE    INDGX                READ CATALOG ENTRY
*                                         FOR DATA SET A.PAY(-3)
*                                         INTO YOUR STORAGE
*                                         AREA NAMED LOCAREA
         Check Return Codes

INDGX     CAMLST    NAME,APAY,,LOCAREA
APAY      DC        CL44'A.PAY(-3)'
LOCAREA   DS        0D
          DS        265C
```

The LOCATE macro instruction points to the CAMLST macro instruction. NAME, the first operand of CAMLST, initiates a search for a catalog entry using the name of a data set. APAY, the second operand, specifies the virtual storage location of the name of the generation index and the relative generation number of a data set in the generation data group. LOCAREA, the fourth operand, specifies a 265-byte area you have reserved to receive the catalog information.

After executing this macro instruction, the system replaces the relative generation number that you specified with the data set's absolute generation name. Control is returned to your program at the next executable instruction following the LOCATE macro instruction. If the entry has been located and read successfully, register 15 contains zeros. Otherwise, register 15 contains a return code (see “Return Codes from LOCATE” on page 147). See “Retrieving Information by Data Set Name (LOCATE and CAMLST NAME)” on page 144 for a description of the contents of the output area.

Retrieving Information by Alias (LOCATE and CAMLST NAME)

For each of the preceding functions, you can specify an alias as the name of a data set. Functions proceed as previously described with one exception: the true name replaces the specified alias name.

The format for the LOCATE and CAMLST NAME macros is:

list_addrx
Points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

NAME
To retrieve information from a catalog, code this operand as shown.

dsname_relexp
Specifies the virtual storage location of a fully-qualified data set name, the first or only name of which is the alias. The area containing the name must be 44 bytes long. The name can be defined by a C-type DC instruction.

area_relexp
Specifies the virtual storage location of your 265-byte output area, that you must define. The output area must begin on a doubleword boundary. The first 256 bytes of the output area will contain a volume list that is read from a catalog. If the data set resides on one volume, bytes 252 - 254 can contain the relative track address of the DSCB. This address is relative to the beginning of the volume.

Example

In the following example, the catalog entry containing a list of the volumes on which data set A.B.C resides is read into virtual storage (data set A.B.C, however, is addressed by an alias name, X.B.C).

```
            LOCATE    INDAB                READ CATALOG ENTRY
*                                          FOR DATA SET X.B.C
*                                          INTO VIRTUAL STORAGE
*                                          AREA NAMED LOCAREA.
          Check Return Codes

INDAB       CAMLST    NAME,ABC,,LOCAREA
ABC         DC        CL44'X.B.C'
LOCAREA     DS        0D
            DS        265C
```

The LOCATE macro instruction points to the CAMLST macro instruction. NAME, the first operand of CAMLST, initiates a search of the catalog for an entry using the name of a data set. ABC, the second operand, specifies the virtual storage location of the fully-qualified name of a data set (in this case, data set A.B.C is addressed by its alias X.B.C). LOCAREA, the fourth operand, specifies a 265-byte area you have reserved in virtual storage.

See “Return Codes from LOCATE” on page 147 for a description of the LOCATE return codes.

Reading a Block by Relative Block Address (LOCATE and CAMLST BLOCK)

This format is no longer supported and will result in an error.

Return Codes from LOCATE

Control is returned to your program at the next executable instruction following the LOCATE macro instruction. Register 15 contains one of the following return codes. If register 15 is non-zero, then register 0 contains an ICF catalog return code, described under message IDC3009I in *z/OS MVS System Messages, Vol 6 (GOS-IEA)*.

Table 29. LOCATE Return Codes

Code	Meaning
0 (X'00')	Operation successful.
4 (X'04')	Either the required catalog does not exist or it cannot be opened.
8 (X'08')	Catalog entry not found.
12 (X'0C')	An invalid low-level GDG name was found.
16 (X'10')	A data set exists at other than the lowest index level specified. Register 0 contains the number of the index level where the data set was encountered.
20 (X'14')	An invalid name has been provided

Table 29. LOCATE Return Codes (continued)

Code	Meaning
24 (X'18')	One of the following happened: • A permanent I/O or unrecoverable error was encountered. • An error was found in a parameter list. R1 is set to X'08000000'. • There was a nonzero return code from ESTAE or GETMAIN. R1 is set to X'08000000'.
38 (X'26')	DFSMShsm LOCATE preprocessor has experienced an error.

Note: See *z/OS MVS System Messages, Vol 7 (IEB-IEE)* and *z/OS MVS System Messages, Vol 8 (IEF-IGD)*, message IDC3009I, for documentation of integrated catalog facility catalog and catalog return codes.

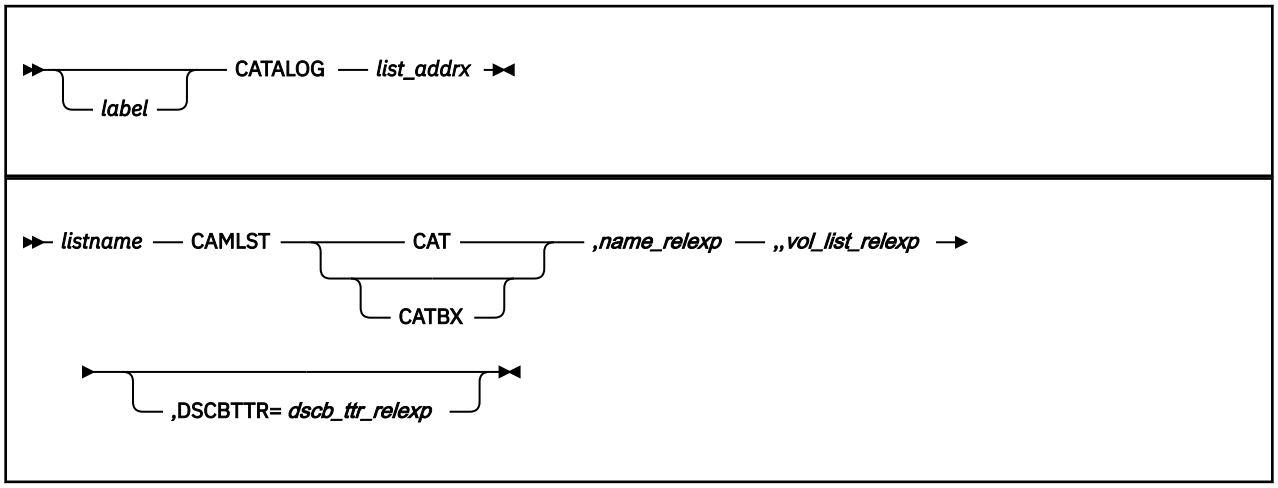
Using Non-VSAM Data Set Catalog Entries

You can catalog, uncatalog, and recatalog non-VSAM data sets using the CATALOG and CAMLST macro instructions. CATALOG macro instructions are used to point to CAMLST macro instructions and to specify cataloging options.

For a description of the search algorithms used for cataloging, uncataloging, and recataloging non-VSAM data sets, see the DEFINE and the DELETE commands in *z/OS DFSMS Access Method Services Commands*.

Cataloging a Non-VSAM Data Set (CATALOG and CAMLST CAT)

The format of the CATALOG and CAMLST macros is:



list_addrx
Points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

CAT or CATBX
Code this operand as shown. Either CAT or CATBX can be coded.

name_relexp
Specifies the virtual storage location of the fully-qualified name of a data set. The name cannot exceed 44 characters. If the name is less than 44 characters, it must be followed by at least one blank. In a DFSMShsm environment, if the data set name is less than 44 characters, it must be padded with blanks until the 44-character length is reached.

vol list_relexp
Specifies the virtual storage location of an area that contains a volume list. The list must begin on a halfword boundary and consist of an entry for each volume on which the data set is stored. The first two bytes of the list indicate the number of entries in the volume list; the number cannot be zero. Each 12-byte volume list entry consists of a 4-byte device code, a 6-byte volume serial number, and a 2-byte data set sequence number. The sequence number is always zero for direct access volumes.

DSCBTTR=*dscb ttr_relexp*
Specifies the virtual storage location of the 3-byte relative track address (TTR) of the data set control block (DSCB). This DSCB is on the first or only volume of the data set. The address is relative to the beginning of the volume.

Programming Considerations for Multiple-Step Jobs

When executing multiple-step jobs, it is preferable to catalog or uncatalog data sets using JCL, instead of using IDCAMS, IEHPROGM, or a user program. Because step allocation and unallocation monitors data sets during job execution and is unaware of functions performed by user programs, conflicting functions can be performed or GDG orientation can be lost.

Unallocation can recatalog existing cataloged data sets at job termination. This action occurs when the data set is opened during the job and the DSCB TTR could not be found in the catalog entry. If you are using the CAMLST macro to uncatalog and then catalog data sets with new volume information, be sure to include the DSCB TTR.

Example

In the following example, the non-VSAM data set named A.B.C is cataloged. The data set is stored on two volumes.

```
           CATALOG ADDABC             CATALOG DATA SET A.B.C.

          Check Return Codes

ADDABC     CAMLST     CAT,DSNAME,,VOLUMES
DSNAME     DC         CL6'A.B.C'      ONE BLANK FOR DELIMITER
VOLUMES    DC         H'2'            DATA SET ON TWO VOLUMES
           DC         X'3010200F'      3390 DISK DEVICE CODE
           DC         CL6'000014'     VOLUME SERIAL NUMBER
           DC         H'0'            DATA SET SEQUENCE NUMBER
           DC         X'3010200F'      3390 DISK DEVICE CODE
           DC         CL6'000015'     VOLUME SERIAL NUMBER
           DC         H'0'            SEQUENCE NUMBER
```

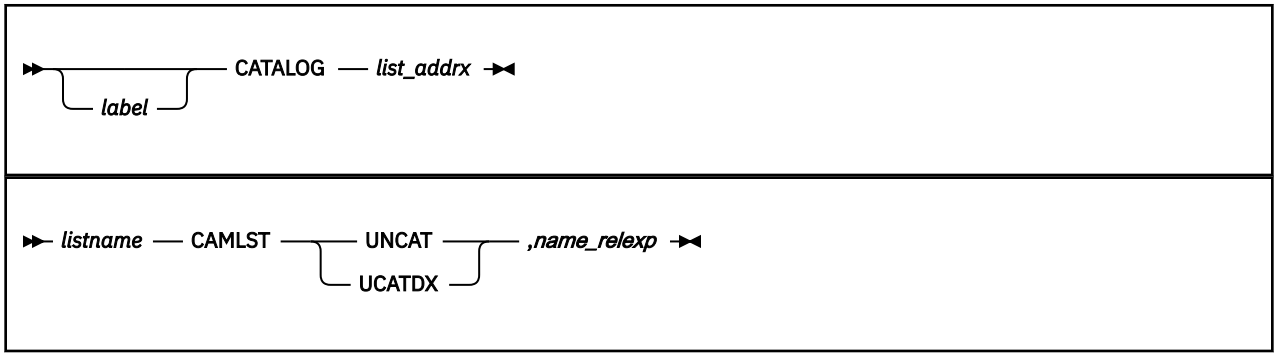
The CATALOG macro instruction points to the CAMLST macro instruction. CAT, the first operand of CAMLST, specifies that a data set is to be cataloged. DSNAME, the second operand, specifies the virtual storage location of the data set name A.B.C. VOLUMES, the fourth operand, specifies the virtual storage location of the volume list.

Control is returned to your program at the instruction following the CATALOG macro instruction. Register 15 contains one of the return codes described under “Return Codes from CATALOG” on page 151.

Uncataloging a Non-VSAM Data Set (CATALOG and CAMLST UNCAT)

Use this macro to remove a data set reference and unneeded indexes.

The format of the CATALOG and CAMLST macros is:



list_addrx
Points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

UNCAT *or* UCATDX
Code this operand as shown. Either UNCAT or UCATDX can be coded but, in either case, unneeded indexes, with the exception of the highest-level index, are removed along with the data set reference.

name_relexp
Specifies the virtual storage location of the fully-qualified name of a data set or index level. The name cannot exceed 44 characters. If the name is less than 44 characters, it must be followed by at least one blank. In a DFSMShsm environment, if the data set name is less than 44 characters, it must be padded with blanks until the 44-character length is reached.

Example

In the following example, the catalog entry for data set A.B.C is removed from a catalog.

```
         CATALOG REMOVE             REMOVE REFERENCES TO DATA SET A.B.C
*                                   FROM CATALOG

        Check Return Codes

REMOVE   CAMLST     UNCAT,DSNAME
DSNAME   DC         CL6'A.B.C'      ONE BLANK FOR DELIMITER
```

The CATALOG macro instruction points to the CAMLST macro instruction. UNCAT, the first operand of CAMLST, specifies that references to a data set are to be removed from the catalog. DSNAME, the second operand, specifies the virtual storage location of the fully-qualified name of the data set whose references are to be removed.

Control is returned to your program at the instruction following the CATALOG macro instruction. Register 15 contains one of the return codes described under “Return Codes from CATALOG” on page 151.

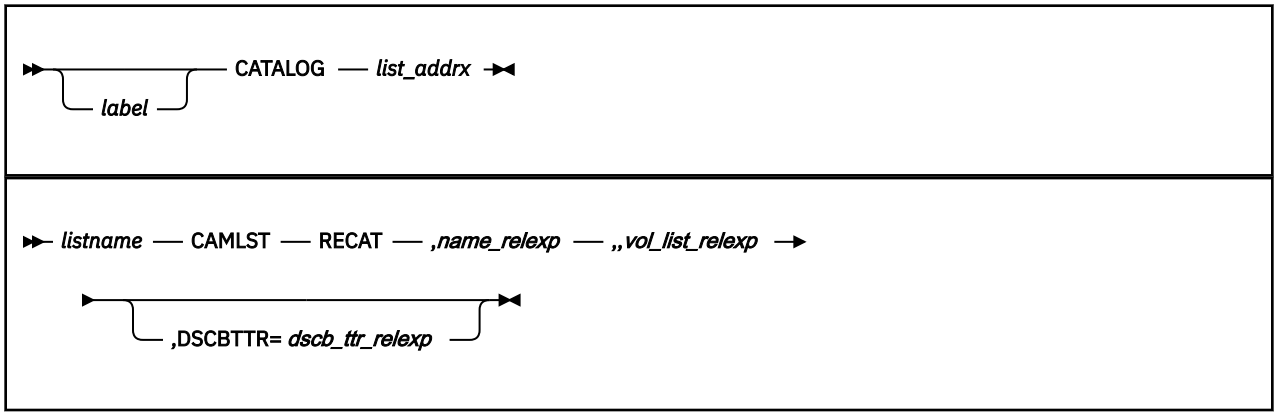
Restriction: The CAMLST UNCAT or UCATDX function is not supported for system-managed data sets. These are ignored. The function is not performed and the return code is 0.

Recataloging a Non-VSAM Data Set (CATALOG and CAMLST RECAT)

You can recatalog a non-VSAM data set using the CATALOG and CAMLST macro instructions. Recataloging is usually necessary if a data set is extended to a new volume.

Build a complete volume list in virtual storage consisting of an entry for each volume on which the data set resides. The first 2 bytes of the list indicate the number of entries in the list; the number must not be zero. Each 12-byte volume pointer consists of a 4-byte device code, a 6-byte volume serial number, and a 2-byte data set sequence number. The sequence number is always zero for direct access volumes.

The format of the CATALOG and CAMLST macros is:



list_addrx
Points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

RECAT
Code this operand as shown.

name_relexp
Specifies the virtual storage location of the fully-qualified name of a data set. The name cannot exceed 44 characters. If the name is less then 44 characters, it must be followed by at least one blank. In a DFSMShsm environment, if the data set name is less than 44 characters, it must be padded with blanks until the 44-character length is reached. A C-type DC instruction can define the name.

vol list_relexp
Specifies the virtual storage location of an area that contains a volume list. The area must begin on a halfword boundary.

DSCBTTR=*dscb ttr_relexp*
Specifies the virtual storage location of the 3-byte relative track address (TTR) of the identifier DSCB. This DSCB is on the first or only volume of the data set. The address is relative to the beginning of the volume.

Example

In the following example, the two-volume data set named A.B.C is recataloged to add a third volume. An entry is added to the volume list, that previously contained only two entries.

```
          CATALOG RECATLG            RECATALOG DATA SET
*                                    A.B.C ADDING A NEW
*                                    VOLUME

         Check Return Codes

RECATLG   CAMLST    RECAT,DSNAME,,VOLUMES
DSNAME    DC        CL6'A.B.C '      FOR DELIMITER ONE BLANK
VOLUMES   DC        H'3'             THREE VOLUMES
          DC        X'3010200E'      3380 DISK DEVICE CODE
          DC        CL6'000014'      VOLUME SERIAL NUMBER
          DC        H'0'             SEQUENCE NUMBER
          DC        X'3010200E'      3380 DISK DEVICE CODE
          DC        CL6'000015'      VOLUME SERIAL NUMBER
          DC        H'0'             SEQUENCE NUMBER
          DC        X'3010200E'      3380 DISK DEVICE CODE
          DC        CL6'000016'      VOLUME SERIAL NUMBER
          DC        H'0'             SEQUENCE NUMBER
```

The CATALOG macro instruction points to the CAMLST macro instruction. RECAT, the first operand of CAMLST, specifies that a data set is to be recataloged. DSNAME, the second operand, specifies the virtual storage location of the fully-qualified name of the data set to be recataloged. VOLUMES, the fourth operand, specifies the virtual storage location of the volume list you have built.

Control is returned to your program at the instruction following the CATALOG macro instruction. If the data set has been successfully recataloged, register 15 contains zeros. Otherwise, register 15 contains one of the return codes described under “Return Codes from CATALOG” on page 151.

Return Codes from CATALOG

Control is returned at the instruction following the CATALOG macro instruction. Register 15 might contain one of the following return codes. If register15 is nonzero, then register 0 contains an ICF catalog return code, described in message IDC3009I in *z/OS MVS System Messages, Vol 6 (GOS-IEA)*.

Table 30. CATALOG Return Codes

Code	Meaning
0 (X'00')	Operation successful.
4 (X'04')	Either the required catalog does not exist or it is not open.

Table 30. CATALOG Return Codes (continued)

Code	Meaning
8 (X'08')	One of the following happened: • The existing catalog structure is inconsistent with the operation requested. • The user is not authorized to perform the operation.
20 (X'14')	There is insufficient space in the catalog data set. If R1 = X'08000000', then an invalid name has been provided.
28 (X'1C')	One of the following happened: • A permanent I/O or unrecoverable error was encountered. • An error was found in a parameter list. R1 is set to X'08000000'. • There was a nonzero return code from ESTAE or GETMAIN. R1 is set to X'08000000'.

Chapter 4. Executing Your Own Channel Programs

This information describes the execute-channel-program (EXCP) and execute-channel-program-virtual-real (EXCPVR) macro instructions and is provided for compatibility with other IBM operating systems. References to EXCP apply equally to EXCPVR unless otherwise stated. IBM recommends using an access method such as VSAM in place of EXCP or EXCPVR.

The EXCP and EXCPVR macro instructions allow you to control the data organization based on device characteristics. The exceptions to this capability are partitioned data sets extended (PDSEs), extended format data sets, spooled and dummy data sets, TSO terminals, and z/OS UNIX files and file systems. They are not supported for user-written applications using EXCP. This information covers EXCP macro instruction application and function and includes descriptions of specific control blocks and macro instructions. Factors that affect the operation of EXCP, such as device variations and program modification, are also discussed.

Before reading this information, you should be familiar with the operational characteristics of the I/O devices required by your channel programs. Operational characteristics are described in IBM publications for each I/O device. You also need to understand the information in the following publications:

- *HLASM Programmer's Guide*, which is available at High Level Assembler and Toolkit Feature in IBM Documentation (www.ibm.com/docs/en/hla-and-tf/1.6), contains information about coding programs in the assembler language.
- *z/Architecture Principles of Operation*, SA22-7832, describes channel command words (CCWs) and channel programs.
- *z/OS DFSMS Using Data Sets* contains the standard procedures for I/O processing under the operating system.
- *z/OS DFSMS Macro Instructions for Data Sets* describes the system macro instructions that can be used in programs coded in the assembler language.

EXCP is primarily for I/O programming situations that cannot be dealt with using standard access methods. When writing your own access method, include EXCP for I/O operations. You must also use EXCP for processing nonstandard magnetic tape labels, including reading and writing labels and positioning volumes.

To issue EXCP, provide a channel program and control blocks in your program area. The I/O process then schedules I/O requests for the specified device, executes commands, handles interruptions, directs error recovery procedures, and posts the results of I/O requests.

Comparing EXCP and EXCPVR

EXCP and EXCPVR are two macros you can use to initiate channel program I/O operations, or as it is often put, execute a channel program. Both provide the same function - a device dependent way to perform I/O operations. However, there are a number of differences:

- In order to issue an EXCPVR request, your program must run in a protection key between zero and seven, run in supervisor state, or be APF authorized. An EXCP request, on the other hand, can be issued by programs running in any key, including user key (key 8) or problem state.
- If you issue an EXCPVR request, your program is responsible for translating its own virtual channel program into a real channel program. This includes page fixing your channel program and I/O buffers either before issuing the EXCPVR request, or by using the page fix appendage and updating your channel program with real addresses, building indirect address lists when needed, and updating the address fields within your channel program with real addresses. This allows your program to improve the efficiency of I/O operations in a paging environment, but does add some complexity to your program.

If you issue an EXCP request, you supply a virtual channel program. In other words, the channel program contains the virtual addresses of storage areas that may be in pageable storage, and the system is responsible for translating your virtual channel program into a real channel program.

Note that EXCP requests issued in an APF-authorized program in a V=R address space (EXCP V=R requests, in other words) are not translated. A V=R address space is one defined with ADDRSPC=REAL in the JCL EXEC statement. Because the address space is V=R, any CCWs created by the user already have correct real data addresses. (Translation would only re-create the user's channel program, so the CCWs are used directly.)

For information on using APF, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

Table 31. Summary of the differences between EXCP, EXCPVR, and EXCP V=R

Function	EXCP Characteristics	EXCPVR Characteristics	EXCP V=R Characteristics
Program state and key	Any protection key, supervisor or problem state.	Protection keys 0-7, supervisor state, or APF authorized.	Protection keys 0-7, supervisor state, or APF authorized.
User responsible for translating their channel program	No	Yes	No translation required.
CCW formats supported	0 and 1	0 and 1	0 and 1
Virtual IDAW supported	Yes	No	No
MIDAWs supported	No	Yes	No
High Performance FICON® for System z (zHPF) channel programs supported	Yes	Yes	No
User can modify channel program during execution	No	Yes, non-zHPF only	Yes
Self modifying channel programs supported	No	Yes, non-zHPF only	Yes

Using EXCP and EXCPVR

This information briefly explains the procedures required when issuing the EXCP and EXCPVR macro instruction. To issue the EXCP or EXCPVR macro instruction directly, perform the following tasks.

1. Allocate the data set or device to be used for the EXCP request. See “Allocating the Data Set or Device” on page 155.
2. Construct and open a data control block (DCB) with the DCB and OPEN macro instructions. Optionally create a data control block extension (DCBE) before issuing the OPEN macro. See “Initializing a Data Control Block (OPEN)” on page 232and “Opening the Data Set” on page 155.
3. Create a channel program containing the commands necessary to perform the I/O operations on the appropriate device. See “Creating the Channel Program” on page 156.
4. Create the control blocks needed to initiate the EXCP request. This includes the input/output block (IOB) and event control block (ECB), and optionally the input/output block extension (IOBE), and input/output error data block (IEDB). See “Creating the EXCP-Related Control Blocks” on page 167.

   For more information on specific control blocks, see “Control Block Fields” on page 177.
5. Issue an EXCP or EXCPVR macro instruction to pass the address of the IOB, and optionally the IOBE, to the routines that initiate and supervise I/O operations.

   After issuing EXCP or EXCPVR, issue a WAIT or EVENTS macro instruction specifying the address of the ECB, to wait for the channel program to complete. See “Executing the Channel Program” on page 168.

6. Once the I/O request has completed, examine the completion status of your EXCP or EXCPVR request and process any error conditions that may have occurred. See "Processing the I/O Completion Status" on page 171.
7. If volume switching is necessary (because of a unit exception or end of DASD extent), issue the EOV macro. See "Handling End of Volume and End-Of-Data-Set Conditions" on page 175.
8. When data set processing is complete, close the data set to restore the DCB. See "Closing the Data Set" on page 176.
9. If your program called dynamic allocation, it can optionally call dynamic unallocation.

Allocating the Data Set or Device

You allocate the data set or device in one of the following ways:

- Using job step allocation, see *z/OS MVS JCL Reference*
- Using dynamic allocation, see one of the following:
  - *z/OS MVS Programming: Authorized Assembler Services Guide*
  - *z/OS TSO/E Command Reference*

When your program calls dynamic allocation using SYC 99 or the DYNALLOC macro, you can use the XTIOT option, which is bit S99TIOEX. It causes creation of an XTIOT entry instead of an entry in the TIOT. The XTIOT resides above the 16 MB line. This option requires your program to be APF authorized, in supervisor state, or in a system key (0–7).

With dynamic allocation, you can use the NOCAPTURE option, which is bit S99ACUCB. If the actual UCB address is above the 16 MB line and you do not code LOC=ANY option on the DCBE macro, then, OPEN and EOV will capture it and EOV and CLOSE will uncapture it. (*Capturing* means to create a 24–bit address for a UCB that has an actual address above the 16 MB line. This NOCAPTURE option also causes creation of an XTIOT instead of a TIOT entry, but it does not require your program to have authorization). If you code the LOC=ANY option on the DCBE, then OPEN will not capture the UCB and its address will be in a four-byte field in the DEB.

With dynamic allocation, you can use the S99DSABA bit, which allows the DSAB control block to reside above the 16 MB line. If you turn S99DSABA on, then you must also turn on S99TIOEX and your program must be authorized.

The purpose of these three dynamic allocation options (XTIOT, NOCAPTURE, and LOC=ANY) is to reduce use of storage below 16MB. BSAM, BPAM, and QSAM also support these three options.

To learn whether the allocation has any of these three options, issue the DEVTYPE macro with the INFO=AMCAP option. On a system before z/OS V1R11, bits S99DSABA and S99TIOEX will be off. On a release V1R12 or later system, these options should only be used when NON_VSAM_XTIOT=YES is specified in the DEVSUPxx parmlib member. This is represented by the DFAXTBAM bit in the DFA as mapped by IHADFA.

Opening the Data Set

You must supply a DCB for each data set or device to open. The OPEN macro instruction finishes initializing one or more DCBs so that their associated data sets can be processed. Issue OPEN for all DCBs used by your channel programs. (A dummy data set cannot be opened for EXCP.) Some of the procedures performed by OPEN are:

- Checking data set access authorization
- Constructing the data extent block (DEB)
- Completing the fields in the DCB and DCBE
- Verifying or creating standard labels
- Positioning tape

- Loading your appendage routines.
- Capturing the UCB if the DCBE option "LOC" was set or defaulted to "BELOW", that is LOC=BELOW or not coded, or the "NON_VSAM_XTIOT" option of the DEVSUPxx member of PARMLIB was set or defaulted to "NO", that is NON_VSAM_XTIOT=NO or not coded. In other words, OPEN does not capture the UCB if LOC and NON_VSAM_XTIOT were specified as follows: LOC=ANY and NON_VSAM_XTIOT=YES.

The parameters and different forms of the OPEN macro instruction are described in “OPEN - Initialize Data Control Block for Processing the JFCB” on page 287 and *z/OS DFSMS Macro Instructions for Data Sets*.

Direct Data Set Considerations

To process a multivolume direct data set (BDAM) with EXCP, use the open routines to build a data extent block for each volume. Your program can do this by reading in the JFCB with a RDJFCB macro instruction and opening each volume of the data set with a separate DCB. See “ Using BSAM or EXCP for Random I/O to a Multivolume Data Set” on page 278 for an example of how to code a routine to do this, and “Reading and Modifying a Job File Control Block (RDJFCB Macro)” on page 273 for further uses of the RDJFCB macro.

VSAM Data Set Considerations

With a DCB used to open a component of a VSAM data set you can perform the following tasks:

- Verify that an application has master password or RACF alter authority for the data set.
- Read from or write to a data set to repair data set or catalog damage if normal VSAM processing cannot. Because of the potential for damaging a valid data set or catalog, exercise extreme caution when writing an application using this interface.

You can specify a DCB when opening a component (data or index) of a VSAM data set if the following conditions are met:

- The application has master password or RACF alter authority for the data set.
- The component must not reside on multiple volumes.
- The component must not be a member of a concatenation.
- The DCB must specify the EXCP access method.
- The data set disposition must be either (OLD,KEEP,KEEP) or (SHR,KEEP,KEEP)
- The DCB must specify either the INPUT or UPDAT option.
- Your program must be either APF authorized or in supervisor state.

When opening a VSAM component on a volume that supports extended attribute DSCBs, the specified DCB must point to a DCBE with the EADSCB=OK keyword. When EADSCB=OK is specified, your program must support extended attribute DSCBs. These are format-8 and format-9 DSCBs, where the extent descriptors may contain 28-bit cylinder numbers.

Creating the Channel Program

The channel program contains the instructions used by the channel subsystem and the device to execute an I/O operation. This section describes what you need to consider when you create a channel program for an EXCP request.

There are two types of channel programs, which are described in the following sections:

- “CCW Channel Program” on page 157
- “zHPF Channel Program” on page 158

CCW Channel Program

The CCW channel program you supply is composed of CCWs on doubleword boundaries. Each channel command word specifies a command to be executed and, for data transfer commands, the source or destination area. CCW operation codes are described in the IBM publications for each I/O device.

You can specify both data chaining and command chaining by setting applicable chaining bits in the channel command word and indicating the type of chaining in the IOB. If an I/O error occurs while your channel program executes, the corresponding chaining bits in the IOB need to be set. Otherwise, error recovery could be impossible. The integrity of your data could be compromised. (See "Input/Output Block (IOB) Fields" on page 189 for additional information.) If you specify both data and command chaining in the same channel command word, data chaining takes precedence.

The location of the CCWs, IDALs, MIDALs and I/O buffers in virtual and central storage depend on whether you are using EXCP or EXCPVR, whether you are using format 0 or 1 CCWs, and whether the device supports 64-bit IDALs or MIDALs. In particular:

- When you use format-0 CCWs, the CCWs and the storage areas pointed to by the CCWs (IDALs, MIDALs, or I/O buffers) must be in 24-bit storage. Otherwise, you can use 31-bit storage.
- When the CCW points to an indirect address list (IDAL), each indirect address list word (IDAW) in the list points to a 31-bit or 64-bit I/O buffer, depending on whether 31-bit or 64-bit IDAWs are used and whether the device supports 64-bit IDAWs. 64-bit IDAWs are only supported for disk and tape devices.
- When the CCW points to a modified indirect address list (MIDAL), each modified indirect address word (MIDAW) in the list points to a 64-bit I/O buffer. MIDAWs are only supported for EXCPVR requests for disk devices and only when running on an IBM System z9® or higher processor.
- For EXCP requests, the system translates your virtual channel program into a real channel program. During the translation process, the CCWs are copied to fixed storage, IDALs are created, and the I/O buffers are page fixed. Therefore, the storage restrictions mentioned above apply only to the virtual storage locations of the CCWs and IDALs; the CCWs and IDALs may reside anywhere in central storage. However, if the I/O buffer resides in 64-bit central storage, the device must support 64-bit IDAWs or else the system fails the EXCP request.
- Because the system does not translate your channel program for an EXCPVR request, the storage restrictions mentioned above apply to both virtual and central storage. For example, if format-0 CCWs are used, the CCWs and the storage areas pointed to by the CCWs (IDALs, MIDALs, or I/O buffers) must be in 24-bit virtual and central storage. However, the IDALS and MIDALS can still point above the 16MB line.

The following table summarizes the channel program storage requirements for different types of CCW channel programs:

Table 32. Storage area locations for CCW channel program components

Request and format	Channel Program Component	Virtual storage location	Central storage location
EXCP format 0	CCW	24-bit	Any
	IDAL	24-bit	Any
	I/O Buffer	• 24-bit if pointed to by CCW • 31 or 64-bit if pointed to by an IDAW	Any

Table 32. Storage area locations for CCW channel program components (continued)			
Request and format	**Channel Program Component**	**Virtual storage location**	**Central storage location**
EXCP format 1	CCW	31-bit	Any
	IDAL	31-bit	Any
	I/O Buffer	• 31-bit if pointed to by CCW • 31 or 64-bit if pointed to by an IDAW or MIDAW	Any
EXCPVR format 0	CCW	24-bit	24-bit
	IDAL	24-bit	24-bit
	MIDAL	24-bit	24-bit
	I/O Buffer	• 24-bit if pointed to by CCW • 31 or 64-bit if pointed to by an IDAW or MIDAW	• 24-bit if pointed to by CCW • 31 or 64-bit if pointed to by an IDAW or MIDAW
EXCPVR format 1	CCW	31-bit	31-bit
	IDAL	31-bit	31-bit
	MIDAL	31-bit	31-bit
	I/O Buffer	• 31-bit if pointed to by CCW • 31 or 64-bit if pointed to by an IDAW or MIDAW	• 31-bit if pointed to by CCW • 31 or 64-bit if pointed to by an IDAW or MIDAW

zHPF Channel Program

zHPF channel programs are supported for EXCP and EXCPVR requests for DASD devices. The following diagram shows the different parts of a zHPF channel program.

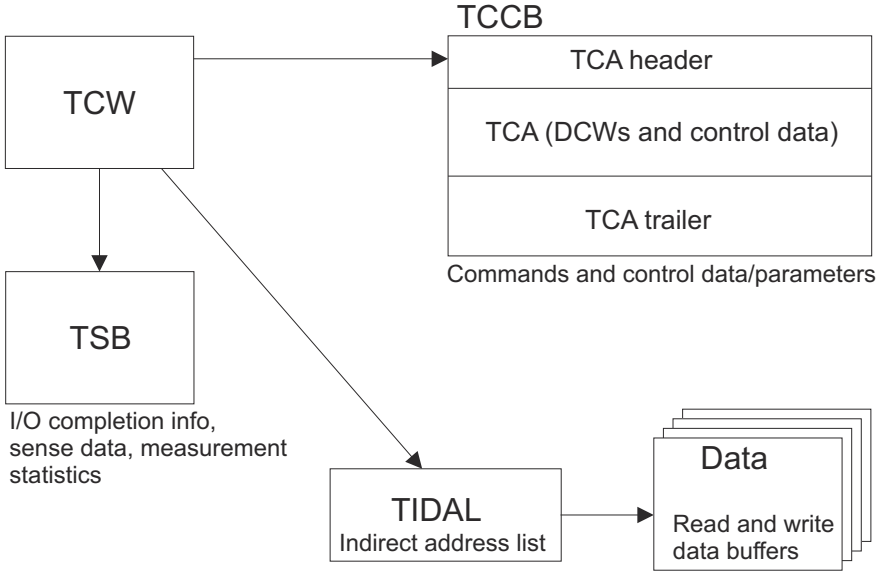


Figure 7. zHPF channel program

The parts of a zHPF channel program include:

- A transport control word (TCW) that contains pointers to all of the other areas of the channel program and the number of bytes to be read or written. The channel uses the TCW to transport the commands and data to the device and locate the status block used to store ending status information; it is not sent to the device. You can use the IOSDTCW macro to map the TCW.

  For an EXCPVR request, the TCW must reside in 24 or 31-bit central and virtual storage.

  For EXCP, the TCW must be in 24 or 31-bit virtual storage or anywhere in central storage.

  For both EXCP or EXCPVR, the TCW must start on a 64-byte boundary. If the TCW does not start on a 64-byte boundary, EXCP fails the request with abend code 800, reason code X'0A'.
- A transport status block (TSB) that contains I/O completion information, sense data, and measurement statistics. The TSB is assigned by z/OS, not the EXCP user. If you provide a TCW with a nonzero TSB address, EXCP fails the request with abend code 800. The system copies all relevant status information from the TSB into the IOBE.
- A transport command control block (TCCB) containing the commands and control data parameters to be passed to the device. The TCCB should start on a doubleword boundary, must reside within a 4 KB page, and may reside in 64-bit virtual and central storage. You can use the IOSDTCCB macro to map the TCCB, including device command words (DCWs). The TCCB consists of three parts:
  - Transport control area header (TCAH) containing information about the transport control area (TCA) and the commands that are contained within.
  - Transport control area (TCA) containing the commands and control parameters. Each command is represented by a DCW that consists of a command code, flags to indicate chaining and other options, a control data count, and a data byte count, if the command is used to transfer data. If the command transfers control data (command parameters) to the device, the control data follows the DCW in the TCA.

    Unlike CCWs, DCWs do not point to their corresponding I/O buffers. The I/O buffers for all DCWs are pointed to by the TCW, and the I/O buffers associated with a particular DCW are based on the amount of data transferred by the previous DCWs.

    The maximum size of the TCA is 240 bytes.
  - Transport control area trailer (TCAT) containing the number of bytes transferred.

  The IBM Z I/O architecture allows the TCCB to be pointed to either directly by the TCW or indirectly via a transport indirect address list (TIDAL). However, TCCB TIDALs are not supported for EXCP or EXCPVR requests. If a TCCB TIDAL is specified, EXCP fails the request with abend code 800.
- One or more I/O buffers. The TCW may point to a single read and/or write buffer.

  For EXCPVR requests, the I/O buffers can be up to 4 KB. If more than 4 KB of data needs to be transferred or the data is non-contiguous or spans a page, a data TIDAL must be created. The I/O buffers and the TIDAWs may reside in 64-bit virtual and central storage. A TIDAL can also be chained to another TIDAL by setting the TIDAL transfer-in-channel (TTIC) bit in the last TIDAW in the list. In this case, the TIDAW address field does not point to an I/O buffer, but instead points to another TIDAL.

  For EXCP requests, the I/O buffers can be greater than 4 KB - see “TIDAW requirements for EXCP requests” on page 164.

The following table summarizes the channel program storage requirements for zHPF channel programs:

Table 33. Storage area locations for zHPF channel program components			
Request and format	**Channel Program Component**	**Virtual storage location**	**Central storage location**
EXCP	TCW	31-bit	Any
	TCCB	Any	Any
	TIDAL	Any	Any
	I/O Buffer	Any	Any
EXCPVR	TCW	31-bit	31-bit
	TCCB	Any	Any
	TIDAL	Any	Any
	I/O Buffer	Any	Any

Comparing CCW and zHPF channel programs

Table 34. Comparing CCW and zHPF channel programs		
Function	**CCW channel programs**	**zHPF channel programs**
Devices supported	All	DASD only
Processors supported	All	EXCP and EXCPVR are only supported on a zEnterprise® 196 (z196) zEnterprise 114 (z114) or higher processor
Command set	All commands for all currently supported devices	Subset of DASD commands
Maximum channel program size	No limit	240 bytes because of the TCA size limit
Maximum data transferred per CCW or TCW	64 KB-1	64 KB, 2 MB, or 4 GB, depending on the processor
Read and write commands in the same channel program	Yes	No
Skip on read	CCW or MIDAW skip bit	TIDAW skip bit
Indirect addressing	IDAL, MIDAL	TIDAL
Data Chaining	Yes	No, but TIDAL is equivalent to data chaining.
Status modifier/search commands	Yes	No
Program controlled interrupt (PCI)	Yes	No
Append to running channel program	Yes	No
Self-modifying channel programs	Yes	No
VIO data sets	Yes	No

EXCP 64-bit Storage Considerations

For EXCP virtual requests, the system page fixes the I/O buffers associated with your channel program. These I/O buffers may reside in 31-bit or 64-bit virtual storage.

If the I/O buffers reside in 64-bit virtual storage (referred to as a memory object), you must follow the following requirement:

- The I/O buffers must not reside within a memory object backed by 1 MB fixed real page frames. For example, you cannot specify PAGEFRAMESIZE=1MEG or PAGEFRAMESIZE=MAX on the IARV64 GETSTOR request that creates the memory object. If you do, EXCP fails when it attempts to page fix the storage.

IDAW Requirements for EXCP Requests

For EXCP requests, the virtual channel program provided by the caller can have one or more CCWs with the indirect address (IDA) flag set and the address portion of these CCWs pointing to a single 31-bit or 64-bit IDAW. This EXCP function is referred to as virtual IDAWs.

The 31-bit IDAW can contain a valid virtual address up to the maximum 31-bit address. Virtual IDAWs are supported on all virtual CCWs except:

- Transfer in channel (TIC) commands
- Read, read backward, and sense commands, with the skip flag set.
- All nondata-transfer type commands: for example, recalibrate, rewind, forward space, fold, block data check, no operation, control commands

A virtual IDAW points to a single, contiguous 31-bit or 64-bit virtual storage area. The virtual storage area can be backed by central storage above or below 2 GB. Either type of virtual IDAW can point to storage that is backed above 2 GB. When EXCP translates the channel program, the virtual IDAW is translated into one or more real IDAWs in 31-bit central storage.

Note: 64-bit IDAWs are supported only for direct access storage devices (DASD) and on all IBM-supplied cartridge tape devices. You should examine bit UCBEIDAW in the UCB to determine whether 64-bit IDAWs are supported by the device.

IDAW Requirements for EXCPVR Requests

For EXCPVR requests, the caller is responsible for creating IDAWs when the data areas associated with the CCWs in your channel program cross certain boundaries. This can be done before issuing the EXCPVR macro or by the SIO appendage after the EXCPVR macro is issued. See “SIO Appendage” on page 203 for more information.

For format-0 channel programs, the CCWs and IDAWs must be below 16 MB in central storage. For format-1 channel programs, the CCWs and IDAWs must be below 2 GB in central storage. Regardless of the channel program format, you can use 31-bit or 64-bit IDAWs to point to storage areas above or below 2 GB in central storage.

Note: 64-bit IDAWs are supported only for direct access storage devices (DASD) and on all IBM-supplied cartridge tape devices. You should examine bit UCBEIDAW in the UCB to determine whether 64-bit IDAWs are supported by the device. If you use 64-bit IDAWs, an IOBE must be specified for the EXCPVR request and flag IOBEEIDA in the IOBE must be set.

If data areas do cross boundaries, provide an additional IDAW in the IDAL for each crossed boundary.

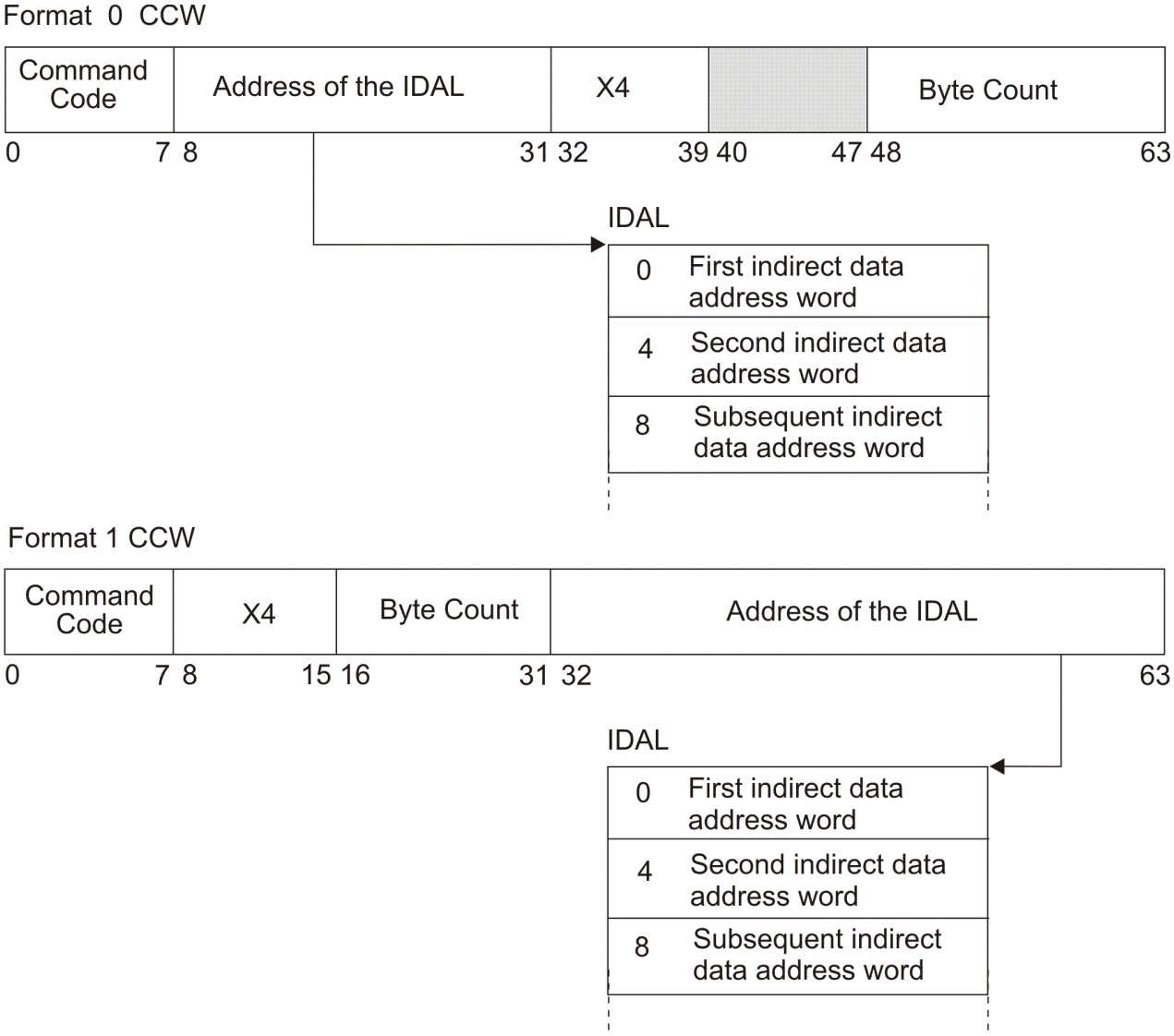
- If you use 31-bit IDAWs , you must use 2 KB boundaries.
- If you use 64-bit IDAWs, you must use 4 KB boundaries.

The channel subsystem uses the IDAL to identify the address where it will continue reading or writing when a boundary is crossed during a read or write operation. The IDAL must contain central storage addresses when it is processed by the channel subsystem.

Before you convert the virtual addresses in the channel program to real addresses, you must first page fix the data areas in central storage. This can be done before issuing the EXCPVR macro or by the page fix appendage after the EXCPVR macro is issued. After the data areas have been page fixed, you can use the LRA instruction to convert the virtual addresses in the channel program to central storage addresses, as long as the central storage addresses are below 2 GB. The LRA instruction returns a 31-bit central storage address regardless of whether you are in 24-bit or 31-bit addressing mode, but fails in those addressing modes if the central storage address is above 2 GB. If the central storage address is above 2 GB, you must either use the LRAG or STRAG instruction to convert the virtual address to a real address or else use the LRA instruction after first switching to 64-bit addressing mode.

If all of the central storage addresses associated with the data buffers are below 2 GB, then you can use 2 KB (31-bit) IDAWs to address the data. Otherwise, you must use 4 KB (64-bit) IDAWs to address the data. For the most efficient use of system resources, code LOC=(ANY,64) or LOC=(BELOW,64) when you obtain storage with the GETMAIN or STORAGE macro. See *z/OS MVS Programming: Assembler Services Reference ABE-HSP*.

The following illustration shows the relationship between the CCW and the IDAWs.



Note the following information about the IDAL:

- If you are using 31-bit IDAWs, then after the first entry, put one entry in the IDAL for each 2 KB page boundary that your data area crosses.
- If you are using 64-bit IDAWs, put one entry in the IDAL for each 4 KB page boundary that your data area crosses and set flag IOBEEIDA in the IOBE..

- If the format 0 CCW has an IDAL address rather than a data address, you must set the indirect address flag (bit 37) on to signal this to the channel. The equivalent format 1 CCW bit is bit 13.
- You can determine the **maximum** number of entries needed in the IDAL from the count in the CCW as follows:

```
Number of IDAWSs = (CCW-byte-count + P + P - 2 ) / P
```

  – Substitute 2048 for P in this formula for 31-bit IDAWs.
  – Substitute 4096 for P in this formula for 64-bit IDAWs.

  For example, the following shows the maximum number of IDAWs for a particular CCW byte-count and a page boundary of 4 KB:

Table 35. Maximum Number of IDAWs for CCW byte-counts

CCW byte-count	Maximum number of IDAWs
1	1
2	2
4095	2
4096	2
4097	2
4098	3

The number of 31-bit IDAWs that are required depends on the number of 2 KB boundaries that are crossed by the data. For example, if your data is 800 bytes long and does not cross a 2 KB boundary, no IDAWs are required. If your data crosses a 4 KB boundary, then two IDAWs are required. If your data is 5000 KB long, at least two IDAWs are required. If your data crosses two 4 KB boundaries, four IDAWs are required.

The first indirect address is the central storage address of the first byte of the data area. The second and subsequent indirect addresses are the central storage addresses of the second and subsequent 2 KB or 4 KB boundaries of the data area. For example, if the data area central storage address is X'707FF' and the byte count is X'1802', the 4-byte IDAL contains the following central storage addresses (assuming the central storage addresses are contiguous):

```
707FF
70800
71000
```

If the data area central storage address is X'707FF' and the byte count is X'800', the IDAL contains the following addresses:

```
707FF
70800
```

MIDAW Requirements

For EXCPVR requests, the channel program provided by the caller can have one or more CCWs with the modified indirect address (MIDA) flag set. When this flag is set, the CCW points to a list of one or more modified indirect address words (MIDAWs). MIDAWs are similar to IDAWs except that the boundary and length requirements are relaxed. For example, each IDAW in a list, except the first, must point to a storage area on either a 2 KB or 4 KB boundary depending on the type of IDAW, and the length includes storage up to the next 2 KB or 4 KB boundary or until the CCW count is exhausted. On the other hand, MIDAWs can point anywhere on a page and contain any length as long as a page boundary is not crossed. MIDAWs can reduce the number of CCWs in your channel program by eliminating the need for data chaining, thereby improving the performance of your channel program.

MIDAWs are supported only for direct access storage devices (DASD) and only when running on an IBM System z9 or higher processor. Examine bit UCBMIDAW in the UCB to determine whether MIDAWs are supported by the device

If you use MIDAWs, you must specify an IOBE for the EXCPVR request and set flag IOBEMIDA in the IOBE.

TIDAW requirements for EXCP requests

The input and output data pointers in the TCW may point to a single, contiguous area of storage, or may point to a virtual TIDAL that consists of one or more virtual TIDAWs, each of which may points to a contiguous area of storage. Unlike the zHPF channel programs created by EXCPVR, these storage areas may be larger than 4K and may span page boundaries.

During channel program translation EXCP creates a real TIDAL when any of the following is true:

- The TCW input or output data pointer points to a virtual storage area that spans pages. For example, the TCW input address points to a 16K byte virtual storage area or points to a 2K byte area that crosses a page boundary.
- The TCW points to a virtual TIDAL. Unlike CCW channel programs, the TIDAL may consist of multiple TIDAWs, each of which can point to a virtual storage area that spans pages. Supporting multiple TIDAWs is necessary for zHPF because there is only a single input and output area pointer (rather than one per CCW), so the storage areas are more likely to be scattered throughout virtual storage.

An EXCP Request with a Single 16 K Storage Area

In Figure 8 on page 164, the TCW points to a single 16 K area of storage. When EXCP translates the channel program, it creates four TIDAWs, one for each page.

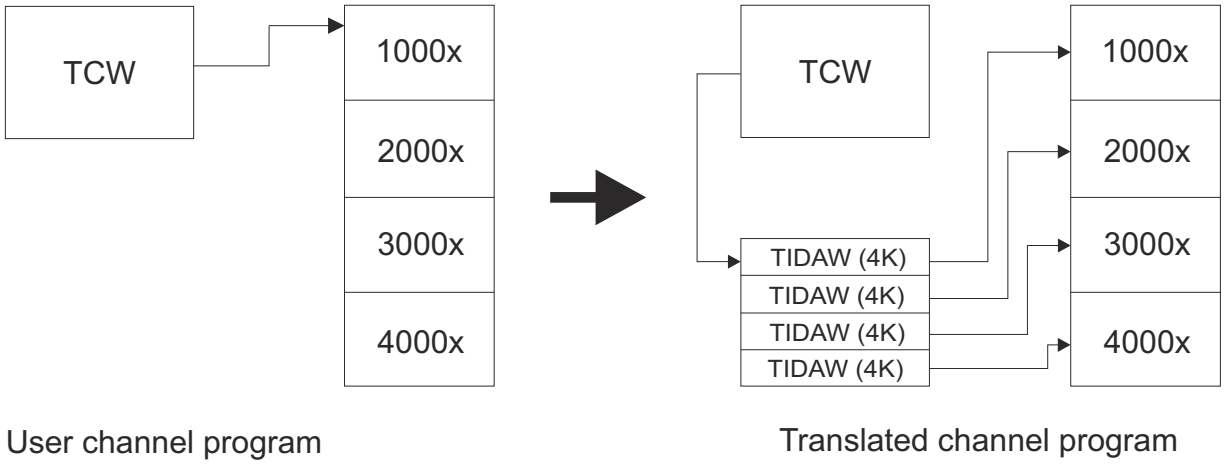


Figure 8. How EXCP translates an EXCP request with a single 16 K storage area

If the TCW points to an area that is less than or equal to 4 K but spanned pages, EXCP will create two TIDAWs.

An EXCP Request with a virtual TIDAL

In Figure 9 on page 165, the TCW points to the virtual address of a TIDAL. The TIDAWs within the TIDAL point either to storage areas that larger than 4K or to storage areas that cross page boundaries. The TIDAL itself also crosses a page boundary. Note that for EXCPVR requests, this would not be allowed because it violates I/O architecture rules. However, for EXCP requests, the virtual TIDAL may have any alignment and may cross page boundaries - EXCP will translate the virtual TIDAL to a real TIDAL and ensure proper alignment and ensure that page boundaries are not crossed.

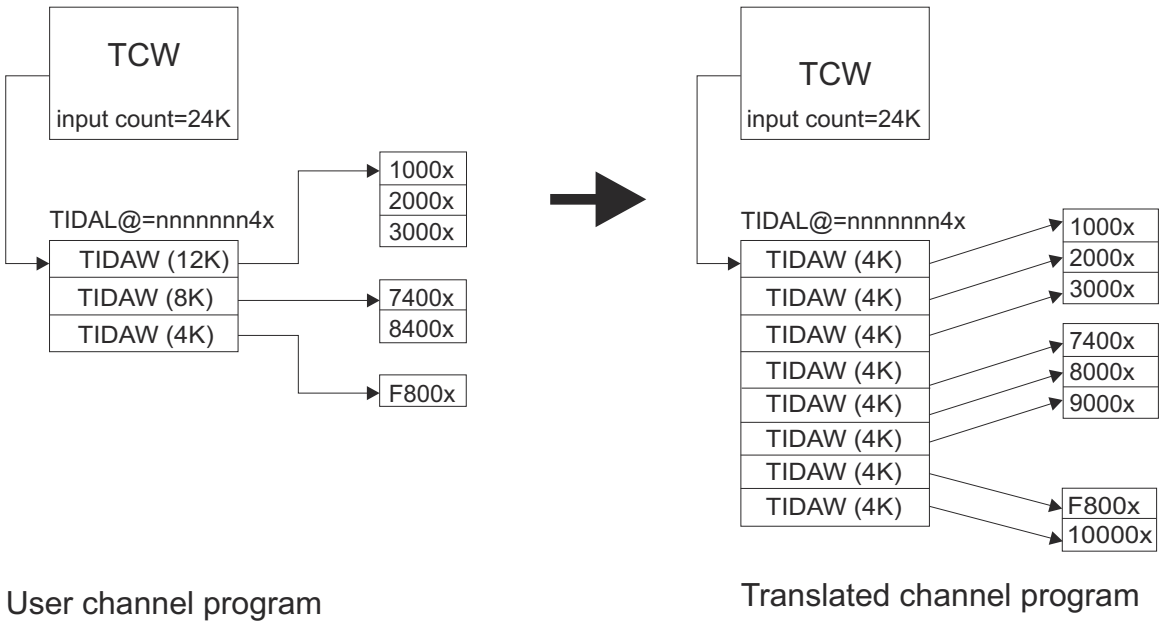


Figure 9. How EXCP translates an EXCP request with a storage areas crossing page boundaries

Determining Whether zHPF is Supported for a Device

Before you create and use a zHPF channel program, you must make sure that zHPF and the zHPF incorrect length facility are supported for both the device and the processor involved. An EXCP or EXCPVR request is only supported on current processors that support the zHPF incorrect length facility. To determine whether the processor supports zHPF, the zHPF incorrect length facility and EXCP/EXCPVR, issue the IOSZHPF macro with the DEVINFO=YES parameter. DEVINFO returns 8 bytes of information about zHPF functions supported for the device, including:

- Processor and z/OS capabilities
  - Functions supported by the channel subsystem and online channels for a device
  - Maximum data transfer size
  - EXCPVR/EXCP virtual supported
  - Incorrect length support
- Device capabilities - You must specify DEVINFO=YES to get device related information. Refer to the device specific architecture to interpret the device related DEVINFO=YES information.
- For those capabilities that have both a processor and device component, you must check bits that are returned in both of the IOSDZHPF and IECDZHPF mapping macros to determine if the capability is supported.

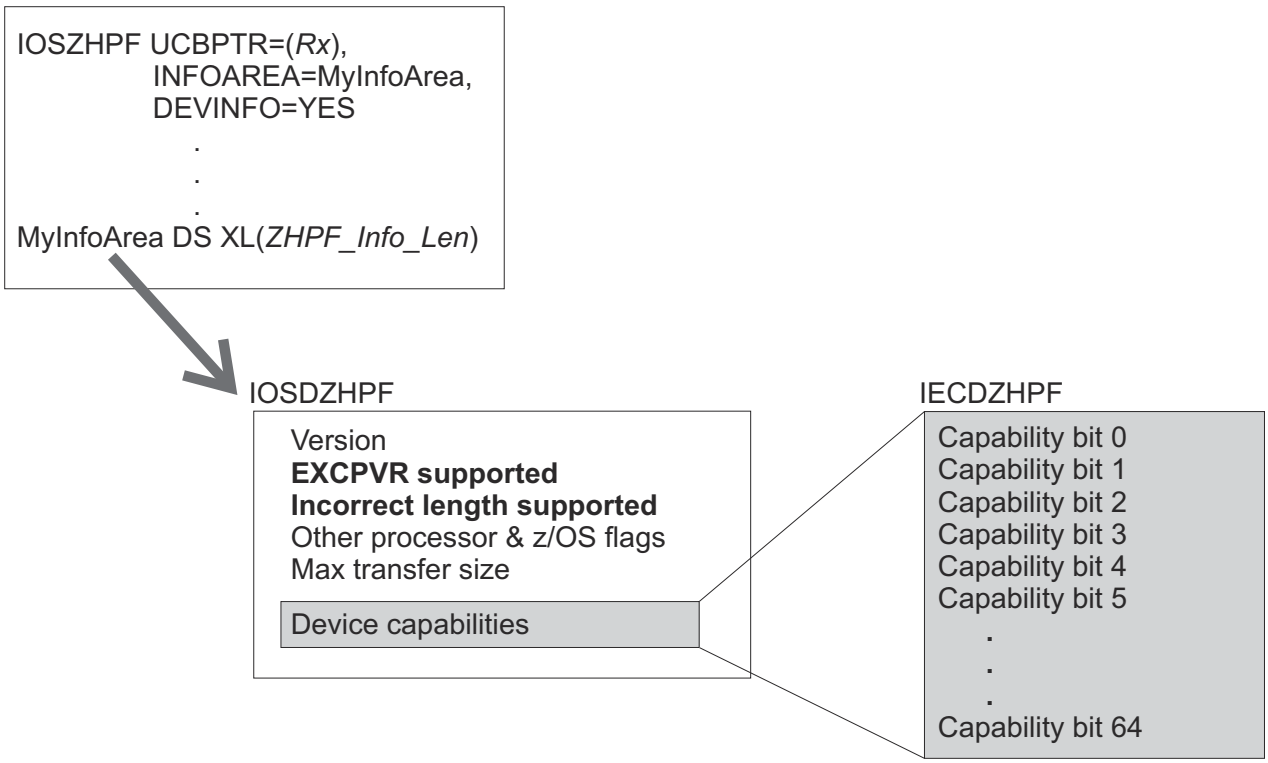


Figure 10. Using IOSPHPF to determine if zHPF is supported by a processor and device

As Figure 10 on page 166 shows, you must check the following in order to check to see whether the processor and device supports zHPF:

- If the processor supports zHPF
- If the processor supports incorrect length
- If the device supports incorrect length

Modifying a Channel Program During Execution

For EXCP requests that run in a V=V address space, the system builds a translated copy of your channel program in real storage. Channel program changes made by your page-fix or start-I/O appendage, which run before the channel program is translated, will affect the real channel program. Later changes to your copy of your channel program with processor instructions or with data read in by an I/O operation will not affect the real translated channel program. Thus, in a V=V address space any attempt to modify an active channel program affects only the virtual image of the channel program, not the real channel program being executed by the channel subsystem. If you wish your changed channel program to be executed, your program can issue another EXCP macro or your channel-end appendage can return at offset 8.

Modifying a channel program during execution is supported for EXCPVR and EXCP requests that run in a V=R address space, since the system does not build a translated copy of your channel program. Modifying a channel program during execution is only supported for CCW channel programs, not zHPF channel programs.

VIO Considerations

VIO data sets are supported for EXCP and EXCPVR requests. When an EXCP or EXCPVR request is issued for a VIO data set, the system simulates all of the common channel commands. Both format-0 and format-1 CCW channel programs are supported for VIO data sets, but note that 64-bit IDAWs, MIDAWs, and zHPF channel programs are not supported for a VIO data set.

For EXCPVR requests, the addresses in the CCWs and IDAWs must be virtual addresses - they must not be converted to real addresses.

Creating the EXCP-Related Control Blocks

Using EXCP requires familiarity with the function and structure of the IOB, ECB, DCB, DEB, and optionally IOBE, DCBE,UCB and IEDB. DCB, IOB, IOBE, IEDB, and ECB fields are illustrated in the "Control Block Fields" on page 177 section. The DEB fields used for EXCP and EXCPVR are illustrated in Appendix A, "Control Blocks," on page 431 (all the DEB fields are illustrated in *z/OS DFSMSdfp Diagnosis*).

The IOB, ECB, DCB, and DEB must be in 24-bit virtual storage. The IOBE, IEDB, and DCBE may reside in 31-bit virtual storage, even if your program runs in 24-bit mode.

All EXCP control blocks that you provide must be located in storage that your program's protection key allows the program to modify. Descriptions of these control blocks follow.

Input/Output Block (IOB)

The input/output block (IOB) is used for communication between the problem program and the system. It provides the addresses of other control blocks, and maintains information about the channel program, such as the type of chaining and the progress of I/O operations. You must define the IOB and specify its address as the only parameter of the EXCP or EXCPVR macro instruction. See "Input/Output Block (IOB) Fields" on page 189.

Input/Output Block Common Extension (IOBE)

The input/output block common extension (IOBE) specifies the type of channel program and its format, options that control the execution of the channel program and the level of ERP processing, and provides the anchor to the IEDB. The IOBE is an extension to the IOB and, like the IOB, provides for communication between the user of EXCP and the system. This control block is required for format-1 CCW channel programs and zHPF channel programs, but is optional for format-0 CCW channel programs. See "Input/Output Block Common Extension (IOBE) Fields" on page 193.

Event Control Block (ECB)

The event control block (ECB) provides you with a completion code that describes whether the channel program was completed with or without error. A WAIT or EVENTS macro instruction, which can be used to synchronize I/O operations with the problem program, must identify the ECB. You must define the ECB and specify its address in the IOB. See "Event Control Block (ECB) Fields" on page 197.

Input/Output Error Data Block (IEDB)

The system uses the input/output error data block IEDB to provide extended error information. This control block is optional. See "Input/Output Error Data Block (IEDB) Fields" on page 187.

Data Control Block (DCB)

The data control block (DCB) provides the system with information about the characteristics and processing requirements of a data set to be read or written by the channel program. A DCB must be generated by a DCB macro instruction that includes parameters for EXCP. If you are not using appendages, a short DCB is constructed. Such a DCB does not support reduced error recovery. You specify the address of the DCB in the IOB. See "Data Control Block (DCB) Fields" on page 177.

Data Control Block Extension (DCBE)

The data control block extension (DCBE) provides further processing options. The DCBE options currently supported by EXCP are the following:

- BLKSIZE
- BLOCKTOKENSIZE
- CAPACITYMODE

- EADSCB
- EODAD
- LOC
- SYNC

See “Data Control Block Extension (DCBE) Fields” on page 186.

Data Extent Block (DEB)

The data extent block (DEB) contains one or more extent entries for the associated data set and other control information. An extent defines all or part of the physical boundaries on an I/O device occupied by, or reserved for, a particular data set. Each extent entry contains the address of a unit control block (UCB) that provides information about the type and location of an I/O device. More than one extent entry can contain the same UCB address. For all I/O devices supported by the operating system, the DEB is produced during execution of the OPEN macro instruction for the DCB. The system places the address of the DEB into the DCB. See “Data Extent Block (DEB) Fields” on page 198.

Executing the Channel Program

This information explains how to pass the channel program to the system for execution and how the system uses your channel program and control blocks after you issue EXCP or EXCPVR request.

Using the EXCP macro instruction

The EXCP macro instruction initiates channel program I/O operations. Whenever you want to execute one of your channel programs, issue EXCP.

The format of the EXCP macro is:

***iob_addr*—RX-type address, (2-12), or (1)**

The address of the IOB of the channel program to be executed.

If your program is also supplying an IOBE, then set register 0 to the address of the IOBE, and set flag IOBCEF on in the IOB to indicate that the IOB extension is present. An IOBE is required for format-1 CCW channel programs.

Using the EXCPVR macro instruction

The EXCPVR macro instruction provides you with the same functions as the EXCP macro instruction and allows your program to improve the efficiency of the I/O operations in a paging environment by translating its own virtual channel programs to real channel programs. To issue EXCPVR, your program must be executing in protection key from 0 - 7, executing in supervisor state, or be APF authorized.

The program issuing the EXCPVR must remain in authorized state until completion of the channel programs. For a description of how to authorize a program, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

The format of the EXCPVR macro is:

***iob_addr*—RX-type address, (2-12), or (1)**
the address of the input/output block of the channel program to be executed.

If your program is also supplying an IOBE, then set register 0 to the address of the IOBE, and set flag IOBCEF on in the IOB to indicate that the IOB extension is present. An IOBE is required for format-1 CCW and zHPF channel programs.

To use EXCPVR, follow the procedures needed to execute an EXCP request and also follow the procedures listed below. If you have already page fixed your channel program and data areas prior to issuing the EXCPVR macro, or you are in a V=R address space, steps 1 and 2 are not necessary.

1. Code PGFX=YES in the DCB associated with the EXCPVR requests and provide a page-fix (PGFX) appendage by specifying SIOA=symbol in the DCB.
2. Fix the data area containing your channel program, the data areas referred to by your channel program, the PCI appendage (if your program can generate program-controlled interrupts), and any area referred to by the PCI appendage including the DEB, IOB, and so on. To fix these areas, build a list in your PGFX appendage containing the addresses of these virtual areas. Any area that you know already is in fixed storage during the I/O can be omitted from the page fix list.
3. If you have not already built indirect address lists or if your channel program still contains virtual addresses prior to issuing the EXCPVR macro, you must do the following two items in your start I/O (SIO) appendage. The SIO appendage is described in "Start-I/O Appendage" on page 202.
   - Determine whether the data areas in virtual storage specified in the address fields of your channel program cross page boundaries. If they do, build an indirect data address list and update your channel program with the address of the indirect address list. For CCW channel programs, build an IDAL or MIDAL.

     If you build an IDAL, put the address of the IDAL in the affected CCW and turn on the IDA bit in the CCW.

     If you build a MIDAL, put the address of the MIDAL in the affected CCW and turn on the MIDA bit in the CCW.

     For zHPF channel programs, build a TIDAL, put the address of the TIDAL in the input or output address field in the TCW, and turn on the input or output TIDAL bit in the TCW.
   - Translate the addresses in your channel programs from virtual to central storage addresses.

Initiating the Channel Program

Issuing EXCP or EXCPVR macro requests execution of the channel program specified in the IOB. The system validates the request by checking fields of the control blocks associated with this request. If the system detects invalid information in a control block, it initiates abnormal termination procedures. The system gets the address of the:

- DCB from the IOB
- DEB from the DCB
- UCB from the DEB.

If this is an EXCPVR request and you have provided a page-fix appendage, the system passes control to it to allow you to either page fix your channel program and data areas, or to provide EXCP with a page fix list so that EXCP will do the page fixing. For a description of the page-fix appendage and its linkage to the system, see "Page Fix and EXCPVR Start I/O Appendage" on page 202.

If you have provided a start I/O (SIO) appendage, the system passes control to it. The system does not examine the channel program until the return from the SIO appendage.The return address from the SIO appendage determines whether the system executes or skips the I/O operation. For a description of the SIO appendage and its linkage to the system, see "Start-I/O Appendage" on page 202.

Translating the Channel Program

For EXCP requests that do not run in a V=R address space, the system performs the following tasks:

- Copies your virtual channel program and translates the copy into one that uses only central storage addresses
- Fixes in real storage the pages used as I/O areas for the data transfer operations specified in your channel program.

DASD Channel Program Prefix CCW Commands

For direct access devices, specify the seek address in the IOB. The system constructs a CCW chain that begins with a Define Extent or Prefix command and passes control to the real version of your channel program. The system uses the contents of DEBXDEF, as described in Appendix A, "Control Blocks," on page 431 Appendix A, on page 443, when building the Define Extent or Prefix command. You can issue a define extent command at the beginning of the channel program. The system copies the data area and replaces the beginning and ending extent addresses and the file mask byte. If file mask byte in your Define Extent is set to inhibit writes, the system will use this value when it builds the Define Extent or Prefix command. The file mask is set to prohibit seek-cylinder CCWs, or, if space is not allocated in full cylinders, seek-head commands. If the data set is open for INPUT, write CCWs are also prohibited. Your channel program can contain Locate Record CCWs.

DASD Rotational Positioning Sensing

On newer storage subsystems, such as IBM D58000, the sector value has no effect. On other subsystems, your channel program can be more efficient for device and channel usage if it supplies sector numbers. Your program can read sector numbers with the read sector command or calculate them with the sector conversion routine. That routine is described in "Obtaining the Sector Number of a Block on an RPS Device" on page 211.

Command Retry Considerations

Command retry is a function of many IBM 3990 and newer storage controllers. When the channel subsystem receives a retry request, it repeats the execution of the CCW, without requiring any additional input/output interrupts. For example, a storage control might initiate a retry procedure to recover from a transient error.

A command retry during the execution of a channel program can cause the following conditions to be detected by the initiating program:

- Modifying CCWs: For EXCPVR and EXCP V=R requests, a CCW used in a channel program must not be modified before the CCW operation has been successfully completed. Without the command retry function, a command is fetched only once from storage by a channel. This allowed a program to determine through condition codes or program controlled interruptions (PCI) that a CCW had been fetched and accepted by the channel. The CCW could be modified before execution. With the command retry function, this procedure cannot be repeated because the channel fetches the CCW from storage again on a command retry sequence. In the case of data chaining, the channel retries commands starting with the first CCW in the data chain.
- Program Controlled Interrupts (PCI): A CCW containing a PCI flag can cause multiple program-controlled interrupts. This will happen if the PCI-flagged CCW was retried during a command retry procedure and a PCI could be generated each time the CCW is executed.
- Residual Count: If a channel program is prematurely terminated during the retry of a command, the residual count in the channel status word (CSW) will not necessarily indicate how much storage was used. For example, if the storage control detects a wrong-length record error condition, an erroneous residual count is stored in the CSW until the command retry is successful. When the retry is successful, the residual in the CSW reflects the correct length of the data transfer.
- Command Address: When data chaining with command retry, the CSW might not indicate how many CCWs have been executed at the time of a PCI. For example:

CCW#	Channel Program
1	Read, data chain

CCW#	Channel Program
2	Read, data chain
3	Read, data chain, PCI
4	Read, command chain

In this example, assume that the storage control signals command retry on Read #3 and the processor accepts the PCI after the channel resets the command address to Read #1 because of command retry. The CSW stored for the PCI will contain the command address of Read #1 when the channel has actually progressed to Read #3.

- Testing Buffer Contents on Data Read: Any program that tests a buffer to determine when a CCW has been executed and continues to execute based on this data can get incorrect results if an error is detected and the CCW is retried.

Magnetic Tape Considerations

For a magnetic tape device, the system constructs a CCW chain to set the mode specified in the DEB and pass control to the real version of your channel program. (You cannot set the mode yourself.) For cartridge tape devices, the mode byte also prohibits supervisor channel command words such as the mode set command. If your program opens a tape for input or read backwards and you do not have RACF authority to write on the volume, the system normally prevents writes. With a reel tape the system does this by requiring the operator to remove the write ring. With a cartridge tape, the system does this using the Mode Set command. See *z/OS DFSMS Using Magnetic Tapes* for more information on tape handling.

Lost Data Condition on IBM 3800

With the IBM 3800 Printing Subsystem, a cancel key or a system-restart-required paper jam causes both a lost data indicator to be set in DCBIFLGS and a lost page count and channel page identifier to be stored in the UCB extension. Reset the lost data indicator bit (DCBIFLDT) and the first two bits in the DCBIFLGS field to zero before reissuing requests to the printer. For additional information see *IBM 3800 Printing Subsystem Programmer's Guide* and *IBM 3800 Printing Subsystem Models 3 and 8 Programmer's Guide.*

Processing the I/O Completion Status

For direct access and tape devices, the system considers the channel program completed when it has received both a channel-end and device-end. The channel-end and device end can be presented by the device separately or simultaneously. But for other devices, the system considers the channel program completed when it receives a channel-end condition in the subchannel status word (SCSW). Unless a channel-end (CHE) or abnormal-end (ABE) appendage directs otherwise, the system places a completion code in the IOB and IEDB and then in the ECB (after appendages have been called). The completion code refers to errors associated with channel end. If device and channel end occur simultaneously, errors associated with device end (that is, unit exception or unit check) are also accounted for.

If device end follows channel end and an error is associated with device end, the completion code in the ECB will not indicate the error. However, the status of the device and channel is saved by the system for the device. The next I/O request directed to the I/O device from any address space is marked as intercepted. The error is assumed to be permanent, and the completion code in the ECB for the intercepted request indicates interception. The DCBIFLGS field of the DCB will also indicate a permanent error. Note that, if a write-tape-mark or erase-long-gap CCWis the last or only CCW in your channel program, the I/O process does not attempt recovery procedures for device end errors. In these circumstances, command chaining a NOP CCW to your write-tape-mark or erase-long-gap CCW ensures initiation of device-end error recovery procedures.

To be prepared for device-end errors, you should be familiar with device characteristics that can cause such errors. After one of your channel programs has terminated, do not release buffer space until determining that your next request for the device has not been intercepted. You can reissue an intercepted request.

Interruption Handling and Error Recovery Procedures

An I/O interruption allows the processor to respond to signals from an I/O device that indicate either termination of a phase of I/O operations or external action on the device. A complete explanation of I/O interruptions is contained in the publication *z/Architecture Principles of Operation.* For descriptions of interruption by specific devices, see the IBM publications for each device.

If error conditions are associated with an interruption, the system schedules the appropriate device-dependent error routine. The operating system might then start another request that is not related to the channel program in error. If the error recovery procedures (ERPs) fail to correct the error, the system places an error code in the IOB, IEDB, and then in the ECB, after appendages have been called.

A channel program might depend upon the successful completion of a previous channel program (as when one channel program retrieves data to be used in building another). The previous channel program is called a related request and must be identified to the system. For a description of this procedure, see IOBFLAGS in “Input/Output Block (IOB) Fields” on page 189 and “Purging and restoring I/O requests (PURGE and RESTORE macros)” on page 288.

If a permanent error occurs in the channel program of a related request, the system removes the request queue elements for all dependent channel programs and returns them to the caller without executing the request. For all requests dependent on the channel program in error, the system places completion codes in the ECBs. The system also places ones in the first two bit positions of the DCBIFLGS field of the DCB. Any new requests for a DCB with error flags are posted complete without execution. To reissue requests that depend on the channel program in error, reset the first two bits of the DCBIFLGS field of the DCB to zero. Then reissue EXCP for each desired channel program.

Reexecuting Channel Programs by Error Recovery Procedures

Under some circumstances the ERP might reexecute a channel program from the beginning. For example, DASD channel programs are almost always retried from the beginning. You will want to build channel programs that the ERP can restart after any CCW or DCW has failed.

For CCW channel programs, if a CCW modifies a data area used by an earlier CCW, the channel program might not reexecute properly.

The following are some situations where a channel program might not give correct results when reexecuted by the ERP (see “Modifying a Channel Program During Execution” on page 166):

- The channel program modifies itself.
- The application program or the PCI appendage modifies the channel program or a data area before receiving notification that the channel program has completed. Generally you can attempt to add CCWs to the end of the channel program.

Example

The following is an example of a DASD channel program that will not always execute correctly. Using this channel program is inadvisable.

```
         CCW   X'47',LRArea,X'60',L'LRArea   Locate Record
         CCW   X'86',Data,X'60',L'Data       Read Data
         CCW   X'92',Search,X'60',L'Search   Read Count
         CCW   X'22',Sector,X'20',1          Read Sector for this count
LRArea   DC    0XL16              Area for Locate Record command
         DC    X'06010002'        Operation, auxiliary, block count (2)
Seek     DC    X'xxxxxxxx'        CCHH for seek
Search   DC    X'xxxxxxxxxx'      Search argument
Sector   DC    X'FF0000'          Sector and transfer length factor
Data     DC    ----
```

The application program would have to store appropriate values at labels "Seek" and "Search" before issuing EXCP. (On an IBM 3990 storage subsystem, it will execute more efficiently if an appropriate value is stored at "Sector" also. On newer storage subsystems, such as the IBM 2107, the sector value has no effect.)

If all or part of the third CCW executes (Read Count), then reexecution of the channel program by the ERP will give different results.

Requesting Extended Error Information

You can request that EXCP and EXCPVR processing return extended error information. This extended error information consists of sense information, a completion code, and other information you can use to help more accurately diagnose I/O errors later encountered. Extended error information is available for all devices.

To request extended error information, you must perform the following steps:

- Follow the EXCP/EXCPVR initialization procedures described in "Input/Output Block (IOB) Fields" on page 189, including building and initializing the DCB, IOB and ECB.
- Define an input/output block common extension (IOBE) and an input/output error data block (IEDB) using the IOSDIOBE and IOSDIEDB mapping macros, respectively. Each must be defined on word boundaries. You must supply both an IOBE and an IEDB to receive extended error information. Initialize the fields as described in "Input/Output Block Common Extension (IOBE) Fields" on page 193 and "Input/Output Error Data Block (IEDB) Fields" on page 187.
- Set register 0 to the address of the IOBE.
- Set register 1 to the address of the IOB.
- Set flag IOBCEF on in the IOB to indicate that the IOB extension is present.
- Issue the EXCP or EXCPVR macro (see "Using the EXCP macro instruction" on page 168 and "Using the EXCPVR macro instruction" on page 168 respectively).

There are two versions of the IEDB. The version number within the IEDB defines how long the IEDB is - the version 1 IEDB is 48 bytes long, and the version 2 IEDB is 96 bytes long.

- The version 1 IEDB contains the following information:
  - The completion code from I/O processing that gives the status. The completion codes and their meanings are shown in Figure 23 on page 197.

    The completion code is contained in the IEDBCOD, which is updated with the results of the I/O requests. The system sets the IEDBCOD field prior to calling the abnormal-end, normal-end, PCI, and end-of-extent appendages. The system also sets this field when the ECB is posted, whether or not you have set any appendages. The system can set the IEDBCOD multiple times as events occur.
  - The sense information, set only after a unit check. Once set, the sense information remains until overlaid by the next unit check. Since the system never clears this area, you might want to clear it before issuing EXCP. See the appropriate device publication for an explanation of the sense information.
- The version 2 IEDB is recommended for zHPF channel programs, and contains all of the information in the version 1 IEDB **plus** the failing storage address when a channel control check or channel data check occurs as a result as a storage or storage key error error, or if a program check or protection check occurs for a zHPF channel program due to a bad storage address.

Requesting Different Levels of ERP Processing

You can request the system to limit error recovery procedure (ERP) processing to selected functions, typically when EXCP or EXCPVR processing has encountered an I/O error. The processing selections available depend upon the device type.

For all devices, except magnetic tape subsystems that use cartridges, the following processing levels are available:

- **No ERP processing**. ERP processing does not attempt error recovery or issue messages. However, ERP processing can perform recovery for non-error unit checks for logging, forced logging mode and buffered log overflow. To request no ERP processing, set at least one of the DCBIFIOE bits on in the DCBIFLGS field in the DCB.

- **Full ERP processing.** ERP processing performs such functions as logging the errors, logging data collected by a control unit for a device, retrying errors, issuing error messages and processing requests from the device. You do not have to request full ERP processing; it is the system default.

For magnetic tape subsystems that use cartridges, such as the 3490 or 3590-1, the following processing levels are available:

- **Basic ERP processing.** ERP processing logs the errors, logs data collected by a control unit for a device and processes requests from a device. In this case, the system does not issue messages or retry errors. To request basic ERP processing, set one of the DCBIFIOE bits on in the DCBIFLGS field in the DCB.
- **Intermediate ERP processing.** ERP processing performs the functions provided by basic processing and also issues any permanent error messages. To request intermediate ERP processing, define an IOBE and set IOBEPMSG on in IOBEERPM. In addition, set at least one of the DCBIFIOE bits on in the DCBIFLGS field in the DCB.
- **Full ERP processing.** ERP processing performs such functions as logging the errors, logging data collected by a control unit for a device, retrying errors, issuing error messages and processing requests from the device. You do not have to request full ERP processing; it is the system default. Bits in DCBIFIOE must be 0. The DCB macro assembles them as zeros.

VIO considerations

When you issue EXCP or EXCPVR against a VIO data set, the system simulates all the common channel commands. When working with VIO data sets, if you supply an IEDB the system verifies its validity, but will not set any fields in it in the current level of the operating system. VIO is not supported for zHPF channel programs.

Invalid ending status

Normally when a channel program ends, the ending CCW or TCW address is stored in field IOBCMD31 or IOBCMDA depending on the type of channel program. However, for certain types of errors, the ending status of the channel program is unpredictable and the ending address will be set to zero. Therefore, your program should never assume that a valid ending address is always provided in the IOB.

For CCW channel programs, the ending CCW address will be set to zero when any of the following conditions occur:

- An interface control check, channel control check, or channel data check has occurred and the ending address in the subchannel status word (SCSW) is not valid
- A program check, protection check, or chaining check has occurred
- The ending CCW address was in a part of the channel program that was modified by device dependent system code. If device dependent system code needs to modify your channel program, it does so in system related storage that is not available to your program.

For zHPF channel programs:

- The ending TCW address will be set to zero when an interface control check, channel control check, or channel data check has occurred and the ending address in the SCSW is not valid.
- The ending DCW offset in the IOBE is more useful for determining where the channel program ended than the TCW, since the DCW is analogous to the CCW in zHPF channel programs. The ending DCW offset will not be stored if the device did not send ending status containing a valid DCW offset or the channel was unable to store the ending status. In this case, the flag indicating that the DCW offset is valid will be off.

Device No Longer Supports zHPF or Required zHPF functions

As mentioned in “Determining Whether zHPF is Supported for a Device” on page 165, you must make sure that a device supports zHPF as well as all the zHPF functions your channel program requires before issuing an EXCP or EXCPVR request for a zHPF channel program. In addition, even if the processor and

device support zHPF and all the zHPF functions required, these might be disabled when you actually submit your channel programs:

- zHPF itself might be disabled on a processor and device that supports it when you submit your channel program for any of the following reasons:
  - The operator issued a command to disable zHPF for the system
  - The zHPF feature is no longer enabled for the control unit associated with the device
  - Internal errors have occurred that caused zHPF to be disabled for the device
  - The device used in the EXCP or EXCPVR request has been swapped to a device that does not support zHPF.

  When the device no longer supports zHPF, the IOB completion code (IOBECBCC) is set to X'41' and the reason code (IOBERCOD) is set to X'0E'. You can retry the EXCP or EXCPVR request by creating a CCW channel program and either reissuing the EXCP or EXCPVR request, or returning at offset +8 in your abnormal end appendage to execute a new channel program.
- Some required zHPF functions might be disabled on a supporting processor and device if you swap to a device that does not support all required zHPF functions. In that case, you must evaluate the zHPF functions that are supported on that device to determine whether you still can use a zHPF channel program or whether you must substitute a CCW channel program instead.

Handling End of Volume and End-Of-Data-Set Conditions

The EOV macro instruction identifies end-of-volume and end-of-data-set conditions. For an end-of-volume condition, EOV causes switching volumes and verifying or creating standard labels. For an end-of-data-set condition (except when another data set is concatenated), EOV causes your end-of-data set routine to be entered. Before processing trailer labels on a tape input data set, decrement the DCBBLKCT field. Your program issues EOV for any of the following reasons:

- Switching magnetic tape or direct access volumes is necessary
- Performing a secondary allocation on the same or another volume for a direct access data set opened for output
- Switching to the next concatenated data set is necessary.

To determine how the system disposes of a tape volume when a program issues an EOV macro, see the description of the DISP parameter of the OPEN macro in *z/OS DFSMS Macro Instructions for Data Sets*.

For magnetic tape, issue EOV when either a tapemark is read or a write command received a unit exception condition. You can also issue EOV to go to the next volume or data set even though neither a tapemark was read nor end-of-tape reached. Bit settings in the 1-byte DCBOFLGS field ofthe DCB determine the action taken when EOV is executed. Before issuing EOV for magnetic tape or DASD make sure that appropriate bits are set in DCBOFLGS. Bit positions 2, 3, 6, and 7 of DCBOFLGS are used only by the system; you are concerned with bit positions 0, 1, 4, and 5. The use of these DCBOFLGS bit positions is as follows:

Table 36. DCBOFLGS Usage

Position	Bit name	Meaning
0	DCBOFLWR	set to 1 indicates that a write command was executed and that a tapemark or DASD file mark is to be written. OPEN sets this bit to 1 if the OPEN option is OUTPUT, OUTIN, OUTINX, or EXTEND. OPEN sets it to 0 for any other OPEN option. For DASD, a 1 also indicates that ECBFDAD and DCBTRBAL contain valid information. See "Device-Dependent Parameters" on page 182 for more information.
1	DCBOFLRB	indicates that a backward read was the last I/O operation
4	DCBOFPPC	indicates that concatenated data sets are to be treated as unlike. For further information, refer to *z/OS DFSMS Using Data Sets*.
5	DCBOFTM	indicates that a tapemark has been read (tape only).

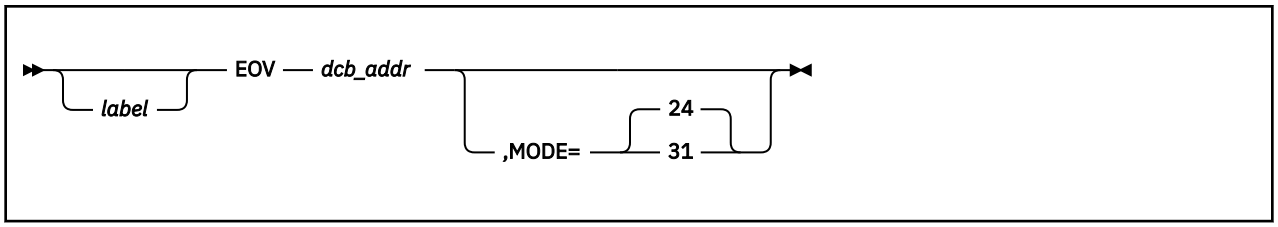
If bits 0 and 5 of DCBOFLGS are both off when EOV is executed, EOV spaces the tape past a tapemark, and standard labels, if present, are verified on both the old and new volumes. The direction of spacing depends on bit 1. If bit 1 is off, the tape is spaced forward; if bit 1 is on, the tape is backspaced.

For tape, if bit 0 is on, but bit 5 is off, EOV writes a tapemark at the current position, which is assumed to be following the last data record of the data set on the current volume. EOV also writes labels on both the old and new tapes if they are labeled. See *z/OS DFSMS Using Magnetic Tapes* for information on label processing. For DASD, if bit 0 is on, EOV attempts to write a file mark. See the DCBFDAD description in "Device-Dependent Parameters" on page 182 for more information.

When you issue EOV, the system might rebuild the DEB at another location. After each EOV, obtain the rebuilt DEB address from the DCB. If the data set was allocated without the nocapture option of dynamic allocation, then EOV might have uncaptured the UCB for the previous volume. If so, that captured UCB address might become invalid. Note that the 24-bit address of the new UCB might be the same as the previous UCB.

After issuing EOV for sequentially organized output data sets on direct access volumes, you can determine whether additional space was obtained on the same or a different volume. Do this by examining the DEB and the UCB. If the volume serial number in the UCB has not changed, additional space was obtained on the same volume. Otherwise, space was obtained on a different volume.

The format of the EOV macro is:



***dcb_addr*—RX-type address, (2-12), or (1)**
The address of the DCB that is opened for the data set. If this parameter is specified as (1), register 1 must contain this address.

MODE=24 or 31
Indicates how the registers are set. With either value, your program can be in 24-bit or 31-bit mode. The modes are:

24
If you do not specify the MODE operand, this mode is assumed. The expansion of the EOV macro stores the DCB address into register 1. The high-order byte of register 1 is ignored during EOV processing. The DCB must be below 16MB, but the calling program can be above the line.

31
The EOV macro expansion code puts the DCB address into register 15 and sets register 1 to zeros. The DCB address must be below 16 MB, because providing a DCB above 16 MB causes an ABEND50D. The high-order byte of the address specified in the EOV macro must be zero.

Closing the Data Set

The CLOSE macro instruction restores one or more DCBs so that processing of their associated data sets can be terminated. Issue CLOSE for all DCBs that were used by your channel programs. Some of the procedures performed when CLOSE is executed are:

- Releasing data extent block
- Removing information transferred to DCB fields when OPEN was executed
- Verifying or creating standard labels
- Volume disposition
- Releasing programmer-written appendage routines

- Uncapture UCB if the actual UCB is above the line and the allocation was done without the nocapture option and LOC=ANY in the DCBE macro was not specified.

When CLOSE is issued for a data set on a magnetic tape volume, the system processes labels according to bit settings in the DCBOFLGS field of the DCB. Before issuing CLOSE for magnetic tape, set the appropriate bits in DCBOFLGS. The significant DCBOFLGS bit positions are listed in the EOV macro instruction description.

When CLOSE is issued for a data set open for output on a direct access device, the system will try to write a file mark if fields are as described in "Device-Dependent Parameters" on page 182.

The parameters and different forms of the CLOSE macro instruction are described in *z/OS DFSMS Macro Instructions for Data Sets*.

Control Block Fields

Control block field information covered here includes fields of the DCB, DCBE, input/output block, IOBE, IEDB, event control block, and data extent block.

To use the DCBD macro to map the DCB, see "Mapping the DCB" on page 185.

Data Control Block (DCB) Fields

The EXCP form of the DCB macro instruction produces a DCB that can be used with the EXCP macro instruction. Code a DCB macro instruction for each data set to be processed by your channel programs. (Notation conventions and format illustrations of the DCB macro instruction are given in *z/OS DFSMS Macro Instructions for Data Sets*.) DCB parameters that apply to EXCP depend on the following elements of the DCB that are generated:

- Foundation block: This portion is required and is always 12 bytes in length. If the DCBMACRF field indicates that a DCB portion before this is missing or short, the system ignores values in those fields.
- EXCP interface: This portion is optional. If you specify any parameter in this category, 20 bytes are generated.
- Foundation block extension and common interface: This portion is optional and is always 20 bytes in length. If this portion is generated, the device-dependent portion is also generated.
- Device dependent: This portion is optional and is generated only if the foundation block extension and common interface portion is generated. Its size ranges from 4 to 20 bytes, depending on specifications in the DEVD parameter. If you do not specify the DEVD parameter and the foundation extension and common interface portions are generated, 16 bytes for this portion are generated.

Some of the procedures performed by the system when the DCB is opened and closed (such as writing file marks for output data sets on direct access volumes) require information from optional DCB fields. Make sure that the DCB is large enough to provide information for the procedures you want the system to handle.

Figure 11 on page 178 shows the relative position of each portion of an opened DCB. The fields, corresponding to the DCB macro instruction parameters, are also identified, except for DDNAME. (DDNAME is not included in a DCB that has been opened.) The fields in parentheses represent system information that is not associated with parameters of the DCB macro instruction.

Sources of information for DCB fields other than the DCB macro instruction are data definition (DD) statements or dynamic allocation parameters, data set labels, and DCB modification routines. You can use any of these sources to specify DCB parameters. However, if a particular portion of the DCB is not generated by the DCB macro instruction, the system will not accept information intended for that portion from any alternative source.

You can provide symbolic names for the fields in one or more EXCP DCBs by coding a DCBD macro to generate a dummy control section (DSECT). For further information, see "Mapping the DCB" on page 185.

The EXCP DCB does not have a field to contain the current or maximum block size but the DCBE does have such a field.

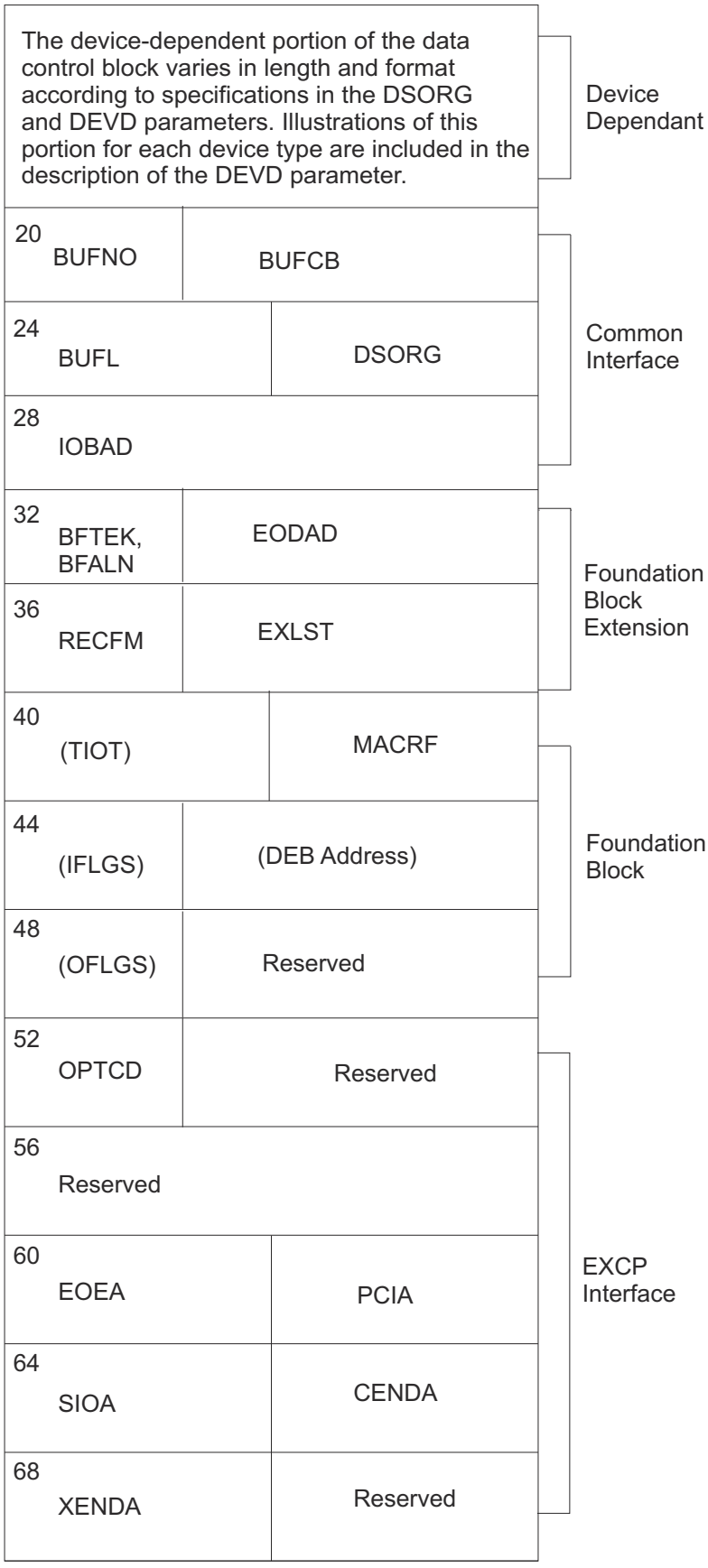


Figure 11. Data control block format for EXCP (after OPEN)

The fields in parentheses represent system information that is not associated with parameters of the DCB macro instruction.

DCB Fields that do not have Macro Parameters

DCBOFLGS

See "Handling End of Volume and End-Of-Data-Set Conditions" on page 175.

DCBIFLGS

See "Processing the I/O Completion Status" on page 171 and "Interruption Handling and Error Recovery Procedures" on page 172.

These are the bits that EXCP uses in DCBIFLGS after the DCB is OPEN:

Table 37. Bits that EXCP uses in DCBIFLGS after the DCB is OPEN

Bits that EXCP uses in DCBIFLGS after the DCB is OPEN		
11..	DCBIFPEC	An IOB without the unrelated bit on got a permanent I/O error. EXCP posts subsequent related requests with X'48' in the ECB until the user clears DCBIFPEC. Corresponds to DCBIBEC in DCBIFLG before OPEN.
..11	DCBIFPCT	Channel 9 or 12 code detected on printer. Corresponds to DCBIFC9 and DCBIFC12 in DCBMACR before OPEN.
.... 11..	DCBIFIOE	Request less than full system ERP (error recovery procedure) processing. Your program can set these bits directly, or you can code the IMSK parameter (see "EXCP Interface Parameters" on page 180). See "Requesting Different Levels of ERP Processing" on page 173.

DCBTIOT

If the data set allocation was dynamic and had the XTIOT (S99TIOEX) or nocapture (S99ACUCB) option, then this field contains zeroes. You can use DEBXTNP, DEBXDSAB and DSABTIOT to get the TIOT or XTIOT entry address. If the allocation was not dynamic or did not have the XTIOT or nocapture option, this field is unsigned (16-bit) offset in the TIOT to the entry.

An entry in the TIOT or XTIOT is mapped by the IEFTIOT1 macro.

Foundation Block Parameters

DDNAME=*symbol*
: The name of the data definition (DD) statement that describes the data set to be processed. This parameter is required and must be supplied before your program issues the OPEN macro. You can code a dummy value and change it before issuing OPEN.

MACRF=E
: The EXCP or EXCPVR macro instruction is to be used in processing the data set. This operand must be coded and must be supplied before your program issues the OPEN macro.

REPOS=Y or N
: Magnetic tape volumes: This parameter indicates to the system whether the user is keeping an accurate block count. (Maintain the block count in the DCBBLKCT field.) The system ignores this parameter when the program is executing and the device is not magnetic tape.

  The EOV and CLOSE functions can record the block count in the IBM standard or ISO/ANSI standard trailer labels.

  On input, the EOV and CLOSE functions can compare the DCB block count with the block count in the standard trailer labels to detect missing or duplicate blocks. This is supported on reel and cartridge tapes.

On input or output, the EOV and CLOSE functions can compare the DCB block count with the block count calculated from tape cartridge subsystems to detect missing or duplicate blocks. The system does this for any label type and for unlabeled tapes. Refer to *z/OS DFSMS Using Magnetic Tapes*.

For magnetic tape devices with reels (not cartridges), coding REPOS=Y allows the dynamic device reconfiguration (DDR) function to move the volume to another tape drive if the drive has a failure. With a cartridge, DDR can do this without your program maintaining block count.

For magnetic tape devices with reels (not cartridges), restart requires the block count in order to restore the tape to its position at the time of the checkpoint. With a cartridge, restart can do this without requiring that you maintain a block count.

The system performs the following tasks:

- Y-The count is accurate.
- N-The block count is inaccurate.

If the operand is omitted, N is assumed.

EXCP Interface Parameters

If you do not code any of the parameters described here, the DCB macro does not create this portion of the block and OPEN will ignore any value that your program sets. You can code values to affect the macro expansion and change any of these fields before the end of the DCB OPEN exit routine. For more information on EXCP appendages, see "Making Appendages Available to the System" on page 200.

EOEA=*symbol*
Last two characters of an EOE appendage name that you have entered into SYS1.LPALIB or SYS1.SVCLIB.

PCIA=*symbol*
Last two characters of a PCI appendage name that you have entered into SYS1.LPALIB or SYS1.SVCLIB.

SIOA=*symbol*
Last two characters of a SIO appendage name that you have entered into SYS1.LPALIB or SYS1.SVCLIB.

PGFIX=YES
The SIO appendage includes a page-fix entry point. Refer to "Using the EXCPVR macro instruction" on page 168 and "Page Fix and EXCPVR Start I/O Appendage" on page 202.

CENDA=*symbol*
Last two characters of a CHE appendage name that you have entered into SYS1.LPALIB or SYS1.SVCLIB.

XENDA=*symbol*
Last two characters of an ABE appendage name that you have entered into SYS1.LPALIB or SYS1.SVCLIB.

OPTCD=Z
Indicates that, for devices that have a magnetic tape reel (input only), a reduced error recovery procedure (four reads only) will occur when a data check is encountered. Only specify this operand when the tape is known to contain errors and the application does not require that all records be processed. Its proper use would include error frequency analysis in the SYNAD routine. Specification of this parameter will also cause generation of a foundation block extension. This parameter does not apply to magnetic tape subsystems that use cartridges. Your program can change this parameter value at any time.

IMSK=*value*
The IMSK parameter lets you specify whether or not you want to ignore system error routines:

- Code X'FFFFFFFF' to specify that you want the system to use the error routines.
- If you code any value other than X'FFFFFFFF', the system will not use the system's error routines. This causes one of the DCBIFOIOE bits to be set on. See "DCBIFLGS" on page 179.

Foundation Block Extension and Common Interface Parameters

EXLST=*address*
The address of an exit list that you have written for exception conditions. The format and purpose of the exit list are provided in *z/OS DFSMS Using Data Sets*.

EODAD=*address*
The address of your end-of-data-set routine for input data sets. If this routine is not available when it is required, the task is abnormally terminated.

It is required when you issue an EOV macro for the last volume containing data and bit 0 of DCBOFLGS indicates that the last operation was a read. (See "Handling End of Volume and End-Of-Data-Set Conditions" on page 175 for more information concerning EOV.) An exception to this is when another data set is concatenated after the current data set.

The EODAD address used here is a 24-bit address in the DCB. If you use the DCB extension (DCBE), and its 31-bit EODAD field is non-zero, the system uses that address instead.

DSORG=PS or PO or DA
The data set organization (one of the following codes). Each code indicates that the format of the device-dependent portion of the DCB is to be similar to that generated for a particular access method:

Code	DCB Format:
PS	QSAM or BSAM
PO	BPAM
DA	BDAM

When writing in a partitioned or sequential DASD data set, you can cause EOV or CLOSE to write a file mark. Refer to "Device-Dependent Parameters" on page 182 for further information.

Do not issue the STOW macro against an EXCP DCB; it can corrupt the directory contents.

IOBAD=*address*
The address of an input/output block. You can use this field for any purpose.

RECFM=*code*
The record format of the data set. (Record format codes are given in *z/OS DFSMS Macro Instructions for Data Sets*.) When writing a data set to be read later, RECFM, LRECL, and BLKSIZE should be specified to identify the data set attributes. LRECL and BLKSIZE can only be specified in a DD statement, in an SVC 99 call, or in the JFCB, because these fields do not exist in a DCB used by EXCP.

IBM recommends that when you are creating a data set you issue a RDJFCB macro and set JFCLRECL and JFCBLKSI before issuing OPEN TYPE=J. This will allow LRECL and BLKSIZE to be in the data set label. An alternate way to set or learn the block size is to use the BLKSIZE field in the DCBE.

The RECFM parameter is not used by the EXCP routines. It provides information stored for subsequent use by access methods that read or update the data set.

The following parameters are optional. The system does not manage the buffers described.

BFALN=F or D
The word boundary alignment of each buffer, either word or doubleword.

BUFL=*length*
The length in bytes of each buffer; the maximum length is 32760.

BUFCB=*address*
The address of a buffer pool control block, that is, the 8-byte field preceding the buffers in a buffer pool. If the low order bit is on, the address is invalid and if you issue the FREEPOOL macro, it has no effect. The FREEPOOL macro sets the low order bit on. The DCB macro sets the low order bit on if the common interface portion is valid.

BUFNO=*number*

The number of buffers assigned to the associated data set; the maximum number is 255. For an EXCP DCB, OPEN ignores this parameter and the BFALN, BUFL, and BUFCB parameters. They have an effect only if your program uses those fields or if your program issues a GETPOOL or GETBUF macro.

Buffer number and block size affect the data transfer rate and the operating system overhead per block. Using more buffers reduces (per block transferred) the system overhead. If you allocate more buffers than your program can process effectively, the virtual pages containing those buffers might be paged out, adding to the system overhead for the job. A large number of buffers also causes a large amount of real storage to be allocated to the job while the data is being transferred.

A job in a low-performance group can get swapped out more frequently than a higher-priority job. The number of buffers allocated for the job effect the number of pages that have to be swapped out.

Programs that access data sets with a small block size (for example, 80) can easily make effective use of 30 buffers, which fit in, at most, two 4096-byte pages. The use of 30 buffers in this case is more efficient than 5, resulting in only 1 channel program rather than 6 channel programs to transfer 30 blocks.

Using data sets with large blocking factors such as half-track blocking on DASD can be effective if only three or four buffers are specified, rather than five or more. The slightly lower DASD performance and small increase in system instruction costs should be more than offset by a reduction in paging or swapping in a constrained environment.

The DCB OPEN installation exit can use installation criteria for a default buffer number for EXCP DCBs (for a description of the OPEN installation exit, see *z/OS DFSMS Installation Exits*). Your program can use the BUFNO value that the OPEN installation exit might set.

Device-Dependent Parameters

DCBE=

The DCBE= parameter is a device-independent parameter but is included here because if DCBE= is specified the entire device-dependent section of the DCB is generated. You specify DCBE= when the DCB extension (DCBE) is required. The first word of the DCB (at offset +0) points to the DCBE when 2 bits at offset 32 (X'20') are on as follows:

Table 38. DCB bits to signify presence of DCBE

Name	Bit
DCBH1	X'80'
DCBH0	X'04'

Coding DCBE= in the DCB macro sets these 2 bits on. For more DCBE information, see "Data Control Block Extension (DCBE) Fields" on page 186. See *z/OS DFSMS Macro Instructions for Data Sets* for more information on the DCBE= parameter of the DCB macro.

DEVD=*code*

The device in which the data set might reside. The codes are listed in order of descending space requirements for the DCB:

Table 39. DCB DEVD options

Code	Device
DA	Direct access
TA	Magnetic tape
PR	Printer
PC	Card punch
RD	Card reader

If you do not want to select a specific device until job setup time, specify the device type requiring the largest area; that is, DEVD=DA.

The following diagrams illustrate the device-dependent portion of the DCB for each combination of device type that is specified in the DEVD parameter and data set organization that is specified in the DSORG parameter. Fields that correspond to device-dependent parameters in addition to DEVD are indicated by the parameter name.

When processing concatenated data sets, the system changes the value in DCBDEVT as appropriate as it reaches each new data set. These values are described in *z/OS DFSMS Macro Instructions for Data Sets*.

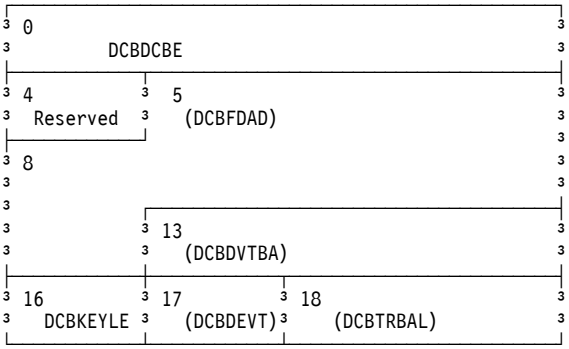


Figure 12. Device-dependent portion of the DCB with DEVD=DA and DSORG=PS (or DSORG=PO)

The fields in parentheses represent information that is not associated with parameters of the DCB macro instruction. EOV sets all of these fields. OPEN sets DCBDVTBA and DCBDEVT, but not DCBFDAD. Your program can modify DCBFDAD, DCBKEYLE, or DCBTRBAL as described in the text.

When writing on DASD, maintain certain fields of the device-dependent portion of the DCB. The system uses the information in the following instances:

- To write a file mark for output data sets
- When releasing unused space at the end of the allocated area

  You can request partial space release using the management class, RLSE on the DD SPACE parameter, TSO/E ALLOCATE command RELEASE parameter or PARTREL macro.
- When adding to the data set with DISP=MOD or the EXTEND or OUTINX options of OPEN
- When DFSMSdss copies the data set.

Maintain the following fields of the device dependent portion of the DCB when writing on DASD:

- The track balance (DCBTRBAL) field that contains a 2-byte unsigned binary number that indicates the remaining number of bytes on the current track. It is recommended that your program not directly calculate this number but use the TRKCALC macro (see "Performing Track Calculations (TRKCALC macro)" on page 292). Both the calculation values and the algorithm differ among device types.
- The full disk address (DCBFDAD) field that indicates the location of the current record. The address is in the form MBBCCHHR. For the actual form of MBBCCHHR, see Table 44 on page 208.

If space is available for output data sets, the system uses the contents of the full disk address (DCBFDAD) field, plus one block, to write a file mark when the DCB is closed or EOV is issued. If the track balance field (DCBTRBAL) is less than eight, the file mark is written on the next sequential track.

Note that the track containing your last data block, as identified by DCBFDAD, might not be the best place for a file mark. Consider the following:

- If the file mark is near the end of the track and a future user extends the data set using DISP=MOD on the DD statement or using the OPEN EXTEND or OUTINX option, then the first new block might be on the next track. This leaves a file mark inside the data.

- For compatibility with BSAM and QSAM when you are writing fixed-standard records, you should cause the file mark to be written wherever the next block would have been written, as if all blocks were full size. The file mark should not be "squeezed" on to the current track.

If the system is to write a file mark, you must maintain the contents of these two fields and set on bit 0 of DCBOFLGS. For further information on DCBOFLGS, see "Handling End of Volume and End-Of-Data-Set Conditions" on page 175. Use the OPEN macro instruction to initialize DCBDVTBA and DCBDEVT. You can use DCBDVTBA or DCBDVTBL with the DEVTAB parameter of the TRKCALC macro (see "Performing Track Calculations (TRKCALC macro)" on page 292 for the TRKCALC description).

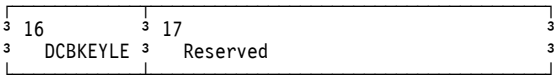
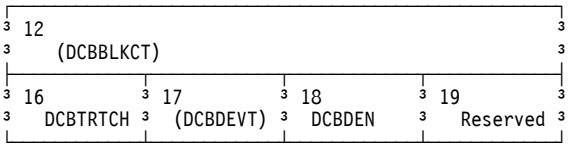


Figure 13. Device-dependent portion of the DCB with DEVD=DA and DSORG=DA



The fields in parentheses represent information not associated with parameters of the DCB macro instruction. They are set by OPEN and EOV. Your program can modify DCBBLKCT as described for the REPOS parameter above.

Figure 14. Device-dependent portion of the DCB with DEVD=TA and DSORG=PS

The system uses the contents of the block count (DCBBLKCT) field to write the block count in trailer labels when the DCB is closed or when the EOV macro instruction is issued. For tape cartridges, the system also compares this count with a count calculated from hardware information. OPEN and EOV set this DCB field to zero except when reading standard labeled tape backward. In that case OPEN or EOV set DCBBLKCT to the block count in the trailer label.

The I/O process increments this field by the contents of the IOBINCAM field of the IOB upon completion of each I/O request.

To indicate to the system that your program is maintaining DCBBLKCT, code foundation block parameter REPOS=Y. See "Foundation Block Parameters" on page 179.

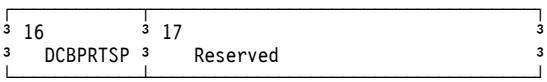


Figure 15. Device-dependent portion of the DCB with DEVD=PR and DSORG=PS

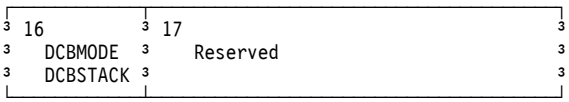


Figure 16. Device-dependent portion of the DCB with DEVD=PC or RD and DSORG=PS

The following DCB operands pertain to specific devices and can be specified only when the DEVD parameter is specified.

KEYLEN=*length*
For direct access devices, the length in bytes of the key of a physical record, with a maximum value of 255. When a block is read or written, the number of bytes transmitted is the key length plus the record length. This parameter does not directly affect EXCP processing, but is stored in the data set label.

DEN=*value*
For magnetic tape reels, the tape recording density in bits per inch.

2
800 (NRZI)

3
1600 (PE)

4
6250 (GCR)

NRZI—Non return-to-zero change to ones recording PE—phase encoded recording GCR—group coded recording

If this parameter is omitted, the highest density available on the device is assumed. Refer to *z/OS DFSMS Macro Instructions for Data Sets* for further information on DEN.

TRTCH=*value*
For magnetic tape subsystems with Improved Data Recording Capability, the tape recording techniques consist of the following values:

Value	Tape recording technique
COMP	Record data in compacted format.
NOCOMP	Record data in standard uncompacted format.

For 7-track magnetic tape, the tape recording technique:

Value	Tape recording technique
COMP	Record data in compacted format.
C	Data conversion feature is available.
E	Even parity is used. (If omitted, odd parity is assumed.)
T	BCDIC to EBCDIC translation is required.

MODE=*value*
For a card reader or punch, the mode of operation. Either C (column binary mode) or E (EBCDIC code) can be specified. This field and parameter do not directly affect EXCP processing but your program can use the field. This is useful to allow you to specify the value on the DD statement.

STACK=*value*
For a card punch or card reader, the stacker bin to receive cards, either 1 or 2. This field and parameter do not directly affect EXCP processing but your program can use the field. This is useful to allow you to specify the value on the DD statement.

PRTSP=*value*
For a printer, the line spacing is 0 — 3. This field and parameter do not directly affect EXCP processing but your program can use the field. This is useful to allow you to specify the value on the DD statement.

Mapping the DCB

In addition to the operands described in *z/OS DFSMS Macro Instructions for Data Sets* for the DSORG parameter of the DCBD macro, you can specify the following operand:

DSORG=XA or XE
Specify the section of the DCB to be mapped.

XA
Specifies a DCB with the foundation block and EXCP interface.

XE
Specifies a DCB with the common interface, foundation block extension, and foundation block.

Code DSORG=(XA,XE) to map all four sections of the DCB.

Data Control Block Extension (DCBE) Fields

The data control block extension (DCBE) provides further processing options. EXCP supports these options:

- The BLKSIZE parameter when you issue the OPEN and EOV macros.
- The BLOCKTOKENSIZE parameter to signify that your program is able to handle a DASD data set that has the large format attribute. The data set is not necessarily large. If you code BLOCKTOKENSIZE=LARGE, then it means that your program can handle a data set that exceeds 65535 tracks or might grow above that size. Look for these differences:
  - If the data set is a large format data set, the DSCB the DS1LARGE bit will be on and the two bytes in DS1LSTAR that contain a relative track number are logically extended with the DS1TTTHI byte. See the format 1 DSCB description in "Format-1 and Format-8 DSCBs" on page 4.
  - The two bytes in each DEBNMTRK field are logically extended by the DEBNMTRKHI byte. See Appendix A, "Control Blocks," on page 431.
  - If your program calls either of the track conversion routines that CVTPRLTV or CVTPCNVT point to, then your program should use the +12 entry points instead of the +0 entry points.

  If you do not code BLOCKTOKENSIZE=LARGE on the DCBE macro, then:
  - If the OPEN macro option is not INPUT and the data set is large format, then OPEN will issue a 213-14 ABEND.
  - If the OPEN macro option is INPUT and the data set has more than 65535 tracks on the volume, then OPEN will issue a 213-16 ABEND.
- The CAPACITYMODE parameter to write more data on an IBM 3590 Magnetic Tape Subsystem that emulates an IBM 3490 when you issue the OPEN macro.
- The EADSCB=OK parameter to specify that your application program supports one of the following, as appropriate:
  - VTOC that describes a volume supporting extended attribute DSCBs. ("Reading and Modifying a Job File Control Block (RDJFCB Macro)" on page 273 describes how to open a VTOC.) An extended address volume may have extended attribute DSCBs. They are format-8 and format-9 DSCBs. If you do not code this option, the OPEN function will issue ABEND 113-48 and message IEC142I. Code this option when your application program supports format-8 and format-9 DSCBs.
  - A data set that has extended attribute DSCBs, Track addresses in DSCBs pointed to by a format-8 DSCB may contain cylinder addresses above 65,520. If you do not code this option, then OPEN issues a 113-44 ABEND and message IEC142I. Code this option when your application program supports MACRF=E (EXCP) and the data set has format 8 and 9 DSCBs.
- The EODAD parameter when you issue an EOV macro. If the DCBEEODAD field has a non-zero value when the system needs to use it, this value takes precendence over an EODAD specification in the DCB. See "Foundation Block Extension and Common Interface Parameters" on page 181.
- The LOC parameter when you issue an OPEN macro. To successfully open a data set that has been allocated with the XTIOT, UCB NOCAPTURE, or DSAB above the line dynamic allocation options, you must specify both:
  - DCBE option LOC=ANY. The LOC=ANY option is represented by the DCBELOCANY bit in the DCBE.
  - DEVSUPxx parmlib parameter NON_VSAM_XTIOT=YES. The DEVSUPxx parameter NON_VSAM_XTIOT is represented by DFAXTBAM bit in the DFA as mapped by IHADFA.
- The CONCURRENTRW=YES,TRKLOCK parameter means that your application program can tolerate serialization on a track basis. Another system can modify one or more tracks while your program is reading another track in the data set extent. Your program cannot read multiple blocks spread across multiple tracks with consistency. For BSAM, QSAM, and EXCP only specify TRKLOCK if your program can tolerate this level of serialization. For EXCP channel programs that do not cross a track boundary, the blocks are consistent. This applies to any type of data set access using EXCP.
- The SYNC parameter to control buffered tape marks on an IBM 3590 when you issue the OPEN, EOV, or CLOSE macros.

Note that the DCBE can be above the 16 MB line even though your user program is running in 24-bit mode. See *z/OS DFSMS Macro Instructions for Data Sets* for more information concerning the DCBE.

Set and Retrieve Data Set Block Size

The DCBE has a field to contain the current or maximum block size for the data set.

In order to use this field, your program must code a value for the BLKSIZE keyword on the DCBE macro or turn on the DCBEULBI bit. The keyword value can be numeric or a non-relocatable symbolic expression. The value can be 0.

If you code a zero value or turn on DCBEULBI, then OPEN stores into DCBEBLKSI the block size value from the data set label if opening for input or output with DISP=MOD, which can be disk or tape. If your program issues the OPEN macro with the OUTPUT or OUTIN option, then OPEN tries to calculate an optimal value for BLKSIZE as it does for BSAM and QSAM. The system-determined block size function is described in *z/OS DFSMS Using Data Sets*. The basic principle is that your program supplies the LRECL value and RECFM value (not U) and OPEN calculates an optimal BLKSIZE value for the device and stores it into DCBEBLKSI.

Attention: OPEN might calculate a BLKSIZE value in the DCBE that exceeds the maximum that is supported by the regular access methods or other programs that read your program's data sets. For example, on magnetic tape the value probably will exceed 32760. If this is a problem, you can do the following:

Supply a DCB OPEN exit routine as described in *z/OS DFSMS Using Data Sets*. If your exit routine finds that the DCBEBLKSI field is still 0, it means that the data set label and the DD statement do not have a value for BLKSIZE. If your program is opening for output, it can leave the DCBE alone and let OPEN calculate an optimal BLKSIZE value or it can calculate one. This is your program's last chance to set it before the system stores the BLKSIZE value in various system control blocks and the data set label. Your DCB OPEN exit routine can issue the DEVTYPE macro with the INFO=AMCAP parameter to learn the optimal and maximum BLKSIZE values for the device. If it is too large, your program can calculate a smaller valid value. Store a value into DCBEBLKSI before OPEN can set it. The INFO=AMCAP parameter of DEVTYPE is described on “DEVTYPE—Info Form” on page 259.

OPEN builds the DEB after calling the DCB OPEN exit routine. If your program calculates a maximum block size, it can take into consideration a value for BLKSZLIM coded on the DD statement. To find this value, issue the RDJFCB macro with an X'13 ' code. See “Reading and Modifying a Job File Control Block (RDJFCB Macro)” on page 273.

Why should your program do this?

- To decrease your program's dependence on the device type
- The system stores this value in the data set label for use by z/OS and on other systems
- The system stores this value in various SMF records to improve monitoring of system resources
- DFSMSrmm™ retains this information for its tape reports.

Unlike the regular access methods, EXCP processing does not do complete checking of the BLKSIZE value in an EXCP DCBE.

Input/Output Error Data Block (IEDB) Fields

The system uses the IEDB to provide extended error information, and for zHPF the failing storage address. The IEDB has two versions:

- The version 1 IEDB is 48 bytes.
- The version 2 IEDB is 96 bytes. This is the recommended version for zHPF channel programs.

You provide an IEDB by setting its address in an IOBE. Set reserved fields to X'00'. The system moves the sense bytes to IEDBSNS. If fewer than 32 sense bytes are available, there might be residual data. Refer to “Requesting Extended Error Information” on page 173 for more information on using the IEDB.

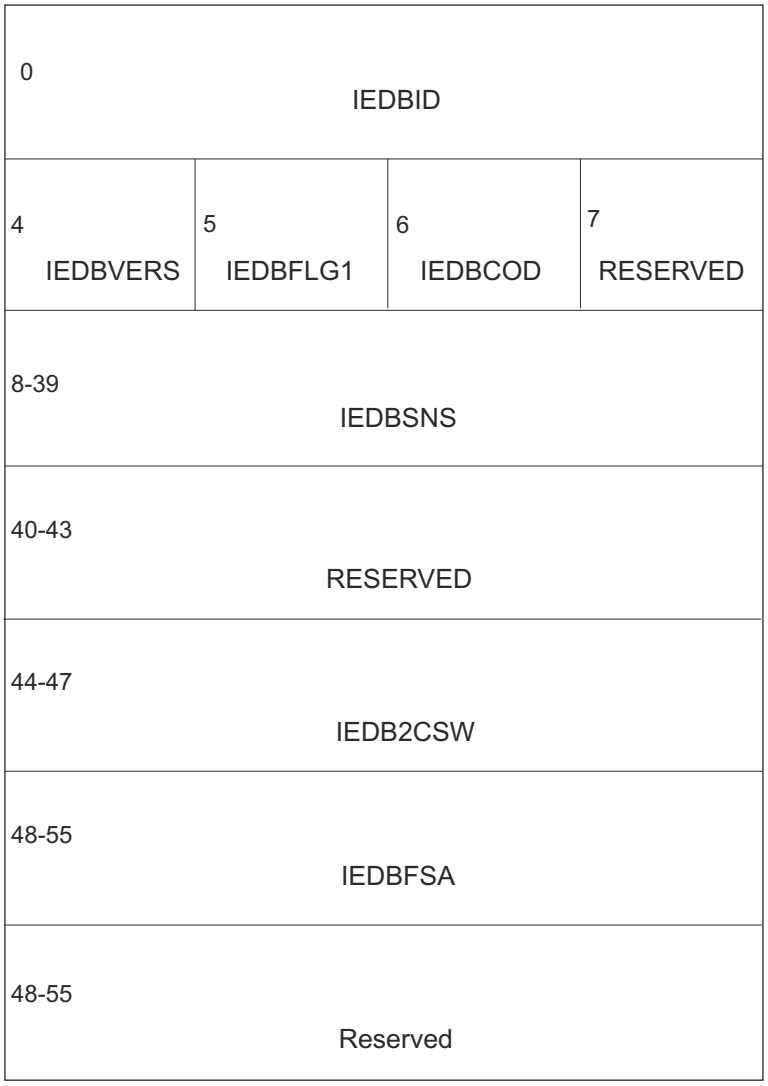


Figure 17. Format of an IEDB, mapped by the IOSDIEDB macro

Table 40. IEDB structure mapping

Offset	Length or Bit Pattern	Name	Description
0 (X'0')	4	IEDBID	Eye catcher. Must be "IEDB"
4 (X'4')	1	IEDBVERS	Version. • The version 1 IEDB is 48 bytes. • The version 2 IEDB is 96 bytes. This is the recommended version for zHPF channel programs. For version 1, the macro defines IEDBVRSC as the version constant. For version 2, the macro defines IEDBVRS2 as the version constant.
5 (X'5')	1	IEDBFLG1	Flags field.
	1...	IEDBBDSN	The sense data is invalid and begins with X'10FE'.
	.1..	IEDBFSAV	The failing storage address (IEDBFSA) is valid.
	..xx xxxx		Reserved.

Table 40. IEDB structure mapping (continued)

Offset	Length or Bit Pattern	Name	Description
6 (X'6')	1	IEDBCOD	Original I/O completion code prior to EXCP or EXCPVR changing it. This is the same format as the ECB completion code byte.
7 (X'7')	1		Reserved.
8 (X'8')	32	IEDBSNS	Sense bytes.
	1	IEDBSNS00	Sense byte 0.
9 (X'9')	1	IEDBSNS01	Sense byte 1.
(Fields IEDBSNS02 to IEDBSNS31 define sense bytes 2 to 31.)			
40(X'28')	4		Reserved.
44(X'2C')	4	IEDB2CSW	Virtual CCW address pointing after the last CCW executed by the control unit. The system sets this only if all of the following are true: (1) EXCP or the user set IOBEP on to allow prefetching of CCWs and data, (2) the user set IOB2CSWS on (two channel status words), (3) the control unit was executing ahead of the channel, (4) the control unit detected an error and (5) the control unit reported the failing CCW. The system never clears this field.
48(X'30')	8	IEDBFSA	The failing storage address for channel control checks and channel data checks. For zHPF channel programs, a failing storage address may also be provided for program checks and protection checks. This field is valid only if IEDBFSAV is on.
56(X'38')	40		Reserved.

Input/Output Block (IOB) Fields

The input/output block (IOB) is not automatically constructed by a macro instruction; it must be defined as a series of constants and be on a word boundary. For unit-record and tape devices, the IOB is 32 bytes long. For direct access, teleprocessing, and graphic devices, 8 additional bytes must be provided. Use the system mapping macro IEZIOB, which expands into a DSECT, to help in constructing an IOB. IEZIOB fields that are not described here are not part of the programming interface.

In Figure 18 on page 190 the shaded areas indicate fields in which you must specify information. The other fields are used by the system and must be defined as all zeros. You cannot place information into these fields, but you can examine them.

You do not have to set the following IOB fields to any particular value before issuing EXCP because the system itself sets them:

- IOBSENS0
- IOBSENS1
- IOBECBCC
- IOBCSW
- IOBSIOCC
- IOBCMD31

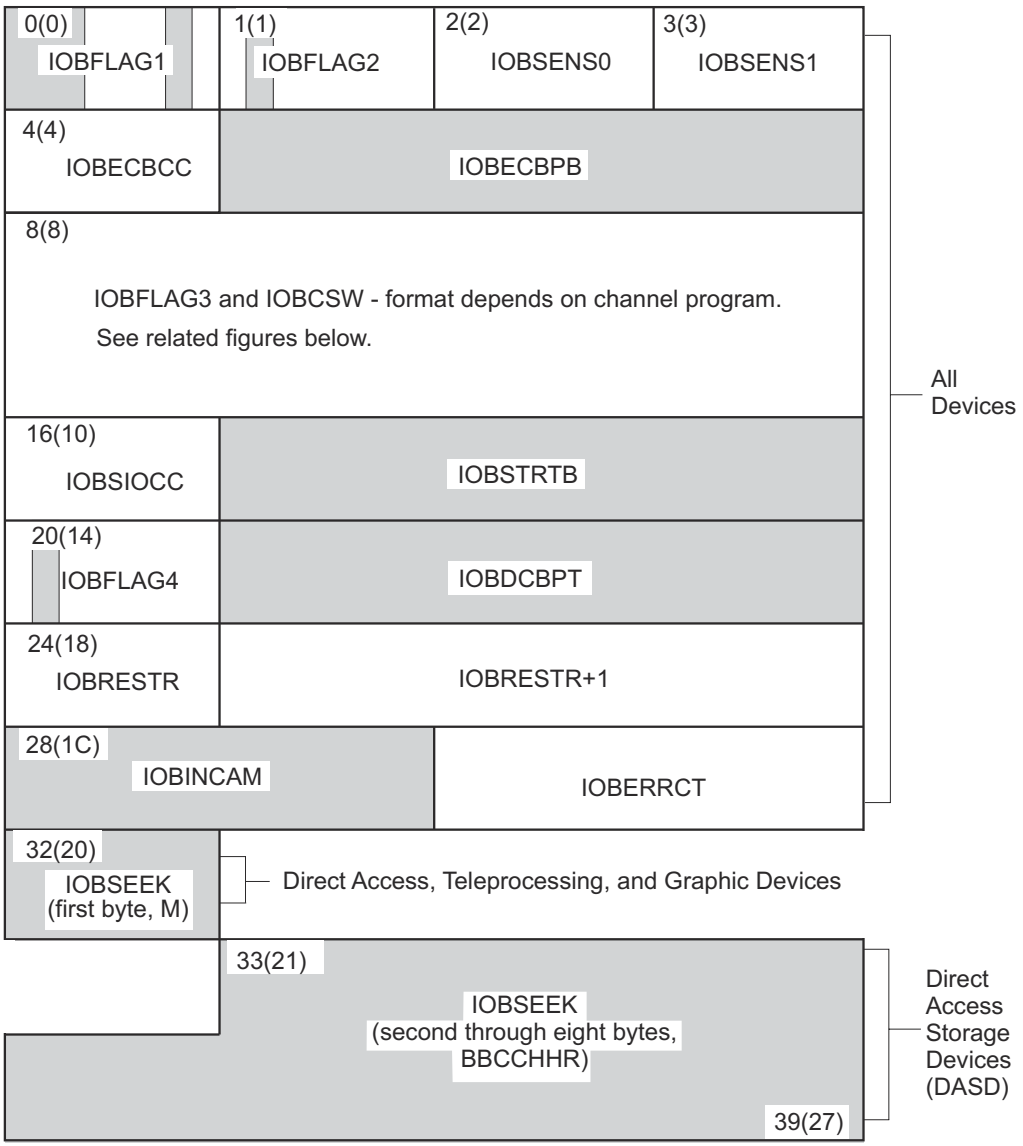


Figure 18. Input/output block (IOB) format

IOBFLAG1 (1 byte)

Set bit positions 0, 1, 6, and 7. One-bits in positions 0 and 1 (IOBDATCH and IOBCMDCH) indicate data chaining and command chaining, respectively. (If you specify both data chaining and command chaining, the system does not use error recovery routines except for the direct access and tape devices.) If an I/O error occurs while your channel program executes, a failure to set the chaining bits in the IOB that correspond to those in the CCW might make successful error recovery impossible. The integrity of your data could be compromised.

A one-bit in position 6 (IOBUNREL) indicates that the channel program is not a related request; that is, the channel program is not related to any other channel program. See bits 2 and 3 of IOBFLAG2 below.

If you intend to issue an EXCP or XDAP macro with a BSAM, QSAM, or BPAM DCB, you should turn on bit 7 (IOBSPSVC) to prevent access-method appendages from processing the I/O request.

IOBFLAG2 (1 byte)

If you set bit 6 in the IOBFLAG1 field to zero, bits 2 and 3 (IOBRRT3 and IOBRRT2) in this field must then be set to one of the following:

- 00, if any channel program or appendage associated with a related request might modify this IOB or channel program.

- 01, if the conditions requiring a 00 setting do not apply, but the CHE or ABE appendage might retry this channel program if it completes normally or with the unit-exception or wrong-length-record bits on in the CSW.
- 10 in all other cases.

The combinations of bits 2 and 3 represent related requests,known as type 1 (00), type 2 (01), and type 3 (10). The type you use determines how much the system can overlap the processing of related requests. Type 3 allows the greatest overlap, normally making it possible to quickly reuse a device after a channel-end interruption. (Related requests that were executed on a pre-MVS system are executed as type-1 requests if not modified.)

IOBSENS0 and IOBSENS1 (2 bytes)
are set by the system when a unit check occurs. These are the first two sense bytes. Occasionally, the system is unable to obtain any sense bytes because of unit checks when sense commands are issued. In this case, the system simulates sense bytes by moving X'10FE' to IOBSENS0 and IOBSENS1.

The first six of these 16 bits have these device-independent meanings:

```
1... ....         Command reject
.1.. ....         Intervention required
..1. ....      Bus out check
...1 ....         Equipment check
.... 1...         Data check
.... .1..         Overrun
```

The last ten of these 16 bits have device-dependent meanings. See appropriate hardware documentation.

If you wish to retrieve more than two sense bytes, supply an IOBE and IEDB as described in “Interruption Handling and Error Recovery Procedures” on page 172.

IOBECBCC (1 byte)
The first byte of the completion code for the channel program. The system places this code in the high-order byte of the event control block when the channel program is posted complete. The completion codes and their meanings are listed under “Event Control Block (ECB) Fields” on page 197.

IOBECBPB (3 bytes)
The address of the 4-byte event control block (ECB) you have provided. The ECB always must be below the 16 MB line. If your program runs in 24-bit, it can use the symbol IOBECBPT, which is four bytes beginning at IOBECBCC but you might want to preserve IOBECBCC.

IOBFLAG3 (1 byte) and IOBCSW (7 bytes)
The system stores status information in these eight bytes. See “IOBFLAG3 and IOBCSW Format for Different Channel Program Types” on page 192.

IOBSIOCC (1 byte)
If the channel program uses format 0 CCWs, bits 2 and 3 contain the start subchannel (SSCH) condition code for the instruction the system issues to start the channel program.

If this is a format 1 CCW channel program or is a zHPF channel program, then field IOBSIOCC is redefined as field IOBSTART, which contains the four byte starting address of the channel program to be executed. (The IOBE field IOBESIOC is used instead of IOBSIOCC.)

IOBSTRTB (3 bytes)
If the channel program uses format 0 CCWs, the three byte starting address of the channel program to be executed. If you supply an IOBE with the IOBEFMT1 bit on, see the above note for IOBSIOCC.

IOBFLAG4 (1 byte)
Set bit 3 (IOBCEF) to indicate whether you are supplying an IOB common extension (IOBE). If this bit is 1, then register 0 contains the IOBE address when you issue EXCP or EXCPVR. Refer to “Requesting Extended Error Information” on page 173 and “Requesting Different Levels of ERP Processing” on page 173.

You must set IOBCEF on if you want to use format 1 CCWs, 64 bit IDAWS, MIDAWS, or a zHPF channel program.

IOBDCBPB (3 bytes)
The address of the DCB of the data set to be read or written by the channel program. The DCB always must be below the line. You can use IOBDCBPT to address the whole word but it is best to preserve the bits in the high order byte.

Reserved (1 byte)
Used by the system.

IOBRESTR+1 (3 bytes)
If a related channel program is permanently in error, this field is used to chain together IOBs that represent dependent channel programs. To learn more about the conditions under which the chain is built, see "Purging and restoring I/O requests (PURGE and RESTORE macros)" on page 288.

IOBINCAM (2 bytes)
For magnetic tape, the amount by which the system increments the block count (DCBBLKCT) field in the device-dependent portion of the DCB. You can alter these bytes at any time. For forward operations, these bytes should contain a binary positive integer (usually +1); for backward operations, they should contain a binary negative integer. When these bytes are not used, all zeros must be specified. See Figure 14 on page 184.

IOBERRCT (2 bytes)
Used by the system.

IOBSEEK (first byte, M)
For direct access devices, the extent entry in the data extent block that is associated with the channel program (0 indicates the first entry; 1 indicates the second, and so forth). For teleprocessing and graphic devices, it contains the UCB index.

IOBSEEK (last 7 bytes, BBCCHHR)
For direct access devices, the seek address for your channel program.

IOBFLAG3 and IOBCSW Format for Different Channel Program Types

The fields IOBFLAG3 and IOBCSW have different meanings depending on the type of channel program and its format.

For format 0 CCW channel programs, these fields have the following format:

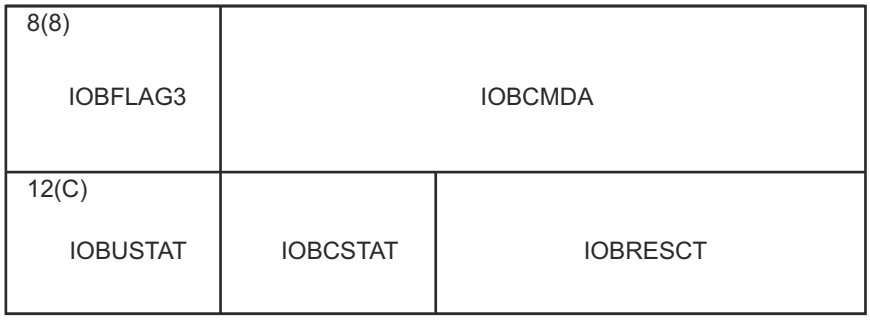


Figure 19. IOBFLAG3 and IOBCSW fields for format 0 channel program

For format 1 CCW channel programs, these fields have the following format:

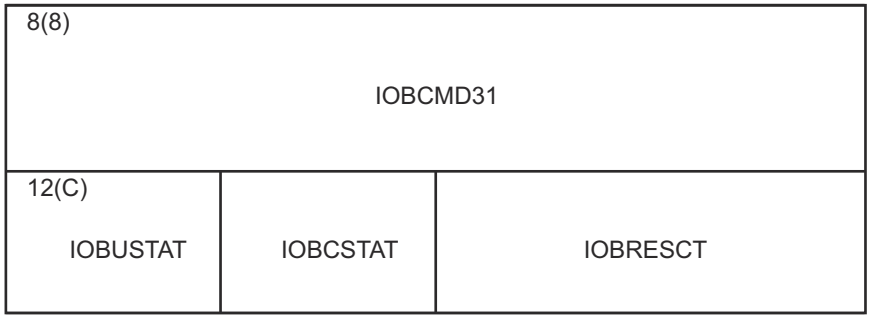


Figure 20. IOBFLAG3 and IOBCSW fields for format 1 channel program

For zHPF channel programs, these fields have the following format:

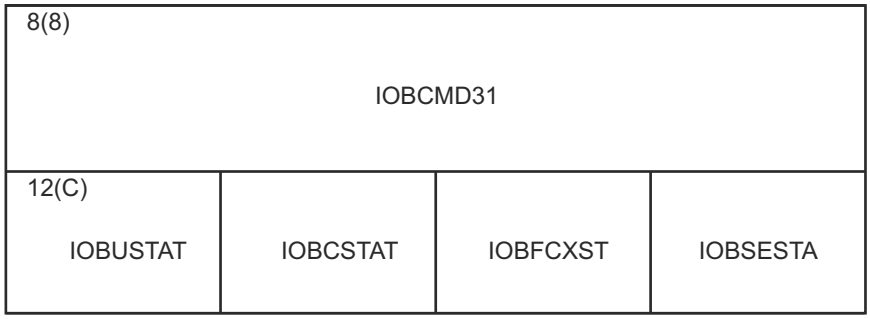


Figure 21. IOBFLAG3 and IOBCSW fields for zHPF channel program

IOBFLAG3
This field is used by the system for format-0 CCW channel programs only.

IOBCMDA
The 24-bit ending CCW virtual address or zero for format-0 CCW channel programs. The ending CCW address points after the last executed CCW in your channel program. The ending address may be zero if either the system determined that the address was invalid or the last executed CCW was a CCW that was added by the system.

IOBCMD31
The 31-bit ending virtual address or zero for format-1 CCW channel programs and zHPF channel programs. For format-1 channel programs, the ending address points after the last executed CCW in your channel program. For zHPF channel programs, the ending address points to the TCW. The ending address may be zero if either the system determined that the address was invalid, or for CCW channel programs, the last executed CCW was a CCW that was added by the system.

IOBUSTAT
The device status (previously called unit status).

IOBCSTAT
The subchannel status (previously called channel status). If the ending address is zero or the subchannel status byte in the IOB (IOBCSTAT) shows any of the following errors: program check, protection check, channel data check, channel control check, interface control check, or chaining check, and your appendage determines that the ERP has not yet run, then let the ERP try to recover and do not modify IOBSTART. If the ERP has completed and one or more of these six bits is on or the address is zero, then the status of the channel program is not known.

IOBRESCT
The residual count for format-0 and format-1 CCW channel programs, which is the number of bytes not transferred in the last CCW. For zHPF channel programs, the residual count is four bytes and contained in the IOBE.

IOBFCXST
Used by the system.

IOBSESTA
The subchannel extended status for zHPF channel programs. The subchannel extended status further qualifies subchannel related errors defined in IOBCSTAT. Macro IHASESQ defines the different subchannel extended status values.

Input/Output Block Common Extension (IOBE) Fields

You can construct an IOB common extension (IOBE) block to receive extended error information or to control the level of error recovery procedure (ERP) processing. To provide an IOBE, set IOBCEF in IOBFLAG4 on and set the IOBE address in register 0 when you issue EXCP or EXCPVR. (The IOBE is mapped by the IOSDIOBE macro).

Set reserved bytes to X'00'. The IOBE can reside above or below 16MB virtual.

The constant IOBELNTH is set to the length of the IOBE and should be used when obtaining and clearing storage. In addition, the constant IOBEEND represents the end of the IOBE.

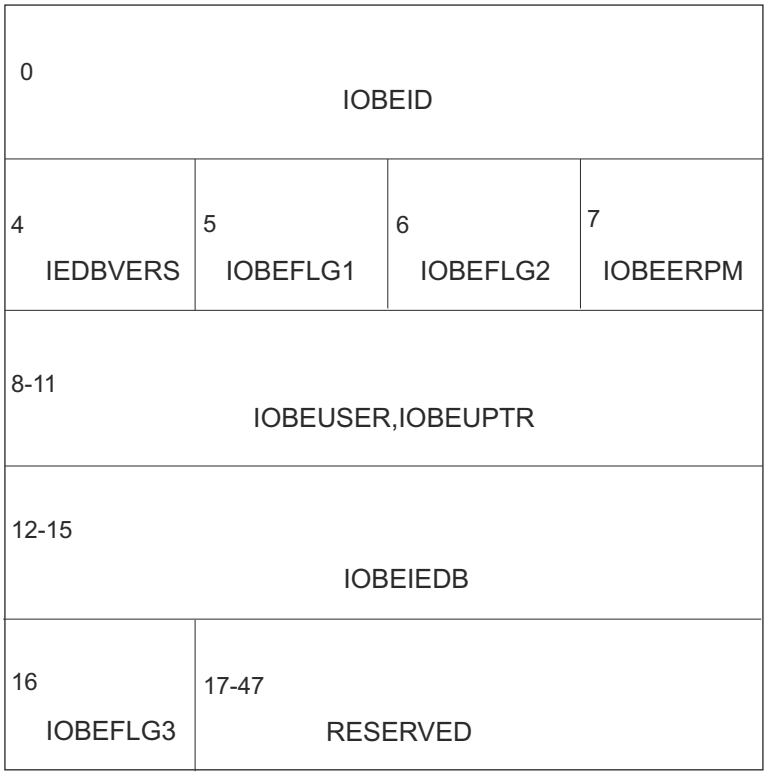


Figure 22. Format of an IOBE, mapped by the IOSDIOBE macro

Table 41. IOBE structure mapping

Offset	Length or bit pattern	Name	Description
0 (X'0')	4	IOBEID	Eye catcher ("IOBE")
4 (X'4')	1	IOBEVERS	Version number (X'01')
5 (X'5')	1	IOBEFLG1	Reserved.

Table 41. IOBE structure mapping (continued)

Offset	Length or bit pattern	Name	Description
6 (X'6')	1	IOBEFLG2	Flag field 2. Set by issuer of EXCP or EXCPVR.
	1...	IOBEMIDA	The channel program might have one or more CCWs that has the MIDA (modified indirect data addressing) bit on and point to a MIDAL (modified indirect addressing list). Supported only for EXCPVR requests. This bit must set to zero if you are using a zHPF channel program.
	.1..	IOBEP	The user allows unlimited prefetching of CCWs. (Unlimited prefetching of the IDAWs, MIDAWs, and data associated with the currrent or prefetched CCW is always allowed.) If zero, no prefetching is allowed, except in the case of data chaining on output, where prefetching of one CCW describing a data area is allowed. Prefetching of CCWs applies only for FICON channels. It has no effect on Enterprise Systems Connection (ESCON) or parallel channels. This bit applies to EXCPVR; EXCP always uses unlimited prefetching. This bit must set to zero if you are using a zHPF channel program.
	..1.	IOBECPNM	The user guarantees that the channel program will not be modified during execution other than to add CCWs at the end. This bit is supported only for EXCPVR requests. EXCP requests that do not run in an authorized V=R address space cannot be modified during execution. This bit must set to zero if you are using a zHPF channel program.
	...1	IOBEEIDA	Any IDAWs are eight bytes each and point to 4096-byte boundaries with the possible exception of the first IDAW in each list. Currently supported only on direct access storage device (DASD) and on IBM-supplied cartridge tape devices. Bit UCBEIDAW in the UCB tells you whether the device supports 64-bit IDAWs. Supported for EXCP and EXCPVR and for format-0 and format-1 channel programs. Not currently supported in VIO. This bit must set to zero if you are using a zHPF channel program.
	.... 1...	IOBEPCIS	PCI synchronization. Channel must synchronize after the next CCW following a PCI (at CCW+8). Supported for EXCPVR only if IOBEP is on. Not supported in VIO. EXCP requests never require PCI synchronization. This bit must set to zero if you are using a zHPF channel program.
	.... .1..	IOBNORWS	No read/write synchronization. The channel should not synchronize on read/write transitions. User guarantees that the reads and writes are from different areas. Always supported for EXCP. Supported for EXCPVR only if IOBEP is on. This bit must set to zero if you are using a zHPF channel program.
	.... ..1.	IOB2CSWS	Two channel status words. If an error occurs where the control unit is executing ahead of the channel, the system returns two ending CCW addresses. The second ending CCW address is in the IEDB. Always allowed for EXCP. Supported for EXCPVR only if IOBEP is on. If this bit is off when the control unit has an error when executing ahead of the channel, the system simulates an invalid ending CCW address. This bit must set to zero if you are using a zHPF channel program.
	1	IOBEFMT1	Channel program is format-1 CCWs and IOBSTART contains a 31-bit address. This bit also causes the system to use IOBESIOC instead of IOBSIOCC and for the ending CCW address to be in IOBCMD31 instead of the three bytes in IOBCMDA. Supported for both EXCP and EXCPVR requests. This bit must set to zero if you are using a zHPF channel program.
7 (X'7')	1	IOBEERPM	Mask indicating the functions the ERP is allowed to perform.
	1.....	IOBEPMSG	User allows basic error recovery procedures (ERP) recovery and permanent error messages that do not require interaction with the system or an operator. See "Requesting Different Levels of ERP Processing" on page 173.
	.xxx xxxx		Reserved.

Table 41. IOBE structure mapping (continued)

Offset	Length or bit pattern	Name	Description
8 (X'8')	4	IOBEUSER	Address field for user use.
	4	IOBEUPTR	Character field for user use.
12 (X'C')	4	IOBEIEDB	Zero or address of IEDB if you want extended error information. For more information see, "Requesting Extended Error Information" on page 173.
16 (X'10')	1	IOBEFLG3	Flag byte 3.
	1...	IOBENSER	Set by user to allow the device to bypass the channel program extent collision checking. Extent range enforcement remains active. Meaningful only if the DASD supports it; has no effect otherwise. Can improve performance on IBM 2105 Enterprise Storage Server™ and newer DASD control units when using multisystem access or multiple parallel access volumes (PAV).
	.1..	IOBENVAL	Set by user to allow the device to bypass the validation checking of the parameters on define extent and locate record commands. Extent enforcement remains active; meaningful only if the DASD device supports it, has no effect otherwise. Can improve performance on IBM 2015 Enterprise Storage Server® and newer DASD control unitswhen using multisystem access or multiple parallel access volumes (PAV).
	..1.	IOBEDSMC	Set by user to disable streaming mode control. This bit must set to zero if you are using a zHPF channel program.
	...1	IOBEIOT	If the user sets this to 0, IOBETIME applies to the request only while the request is active. If this is 1, IOBETIME applies while the request is queued or active.
	.... 1...	IOBEDCWOffsetValid	For zHPF channel programs, the DCW offset in IOBEDCWOffset is valid.
	.... .1..	IOBEResCountValid	For zHPF channel programs, the residual count in IOBEResCount is valid.
	xx		Reserved.
17 (X'11')	1	IOBESIOC	Same format as IOBSIOCC. Valid only if IOBEFMT1 is on.
18 (X'12')	1	IOBETIME	Time limit. If this value is non-zero, it is the number of seconds the user allows the request to take. Has an effect if the DEBACCS field shows the DCB is open for input. There will be no message or error recording due to reaching the limit. Bit IOBEIOT specifies what time is measured.
19(X'13')	1	IOBEFLG4	Flag byte 4.
	1...	IOBEZHPF	The zHPF channel program. This field is set by the user for EXCPVR and EXCP virtual requests. This bit must be set to zero for CCW channel programs.
	.1..	IOBECacheMiss	Specifies one or more I/O device cache misses occurred. This bit must be set to zero if you are using a zHPF channel program.
.xxx xxxx	Reserved	Reserved	Reserved
20(X'14')	4	IOBEResCount	For zHPF channel programs, this field contains the residual count. This field is valid only if IOBEResCountValid is on.
24(X'18')	2	IOBEDCWOffset	For zHPF channel programs, this field contains the offset within the TCA of the DCW that was partially or completely executed. If the channel program could not be completed, this offset identifies the DCW for which processing could not be completed. This field is valid only if IOBEDCWOffsetValid is on.
26(X'1A')	1	IOBEDDPC_RC	For zHPF channel programs, this field contains the device detected program check reason code. This field is valid only when the subchannel status indicates a program check and the subchannel extended status indicates a device dependent program check. The reason codes are documented in *z/Architecture® Principles of Operation*.
27(X'1B')	1	IOBEDDPC_RCQ	For zHPF channel programs, this field contains the first byte of the device detected program check reason code qualifier. This field is valid only when the subchannel status indicates a program check and the subchannel extended status indicates a device dependent program check. The reason code qualifiers are documented in *z/Architecture Principles of Operation*.

Table 41. IOBE structure mapping (continued)

Offset	Length or bit pattern	Name	Description
28(X'1C')	1	IOBERCOD	The I/O completion reason code. This is the second byte of the completion code for the channel program. The system also places this code in the second byte of the event control block when the channel program is posted complete. The reason codes applicable to EXCP or EXCPVR and their meanings are listed in "Event Control Block (ECB) Fields" on page 197.
29(X'1D')	19		Reserved.

Event Control Block (ECB) Fields

Define an event control block (ECB) as a 4-byte area on a word boundary. When the channel program has been completed, the system places a completion code containing status information into the ECB (Figure 23 on page 197). Before examining this information, test for the setting of the complete bit. If the complete bit is not on, and your problem program cannot perform other useful operations, issue a WAIT or EVENTS macro instruction that specifies the ECB. Do not construct a program loop to test for the complete bit.

The ECB is mapped by the IHAECB macro.

You do not have to set any particular values in the ECB before issuing EXCP or EXCPVR because the system clears it.

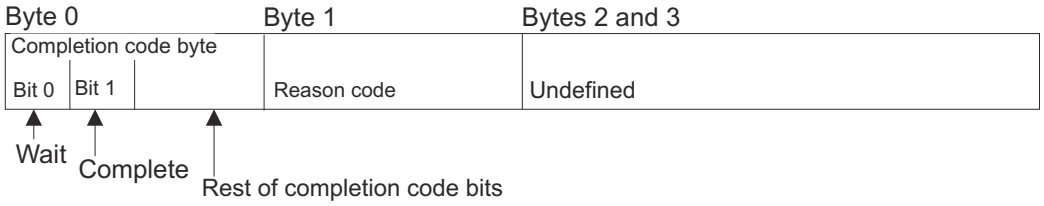


Figure 23. Event control block after posting of completion code

Wait bit
One in this position indicates that the WAIT or EVENTS macro instruction has been issued, but the channel program has not been completed.

Complete bit
A one in this position indicates that the channel program has been completed. If it has not been completed, a zero bit is in this position.

Completion code
This code, which includes the wait and complete bits, might be one of the following hexadecimal expressions. If an appendage posts the ECB, it might use other 4-byte codes in which the first two bits are 01. Refer to "Start-I/O Appendage" on page 202 and "Channel-End Appendage" on page 206.

The system sets these codes in IOBECBCC and IEDBCOD before posting the ECB.

Code
Meaning

41
Permanent I/O error

42
Extent error (DASD only). Either IOBSEEK lies outside the extents described by the DEB or the channel program attempted execution outside the current extent.

44
An error occurred after the previous I/O request to the device was posted complete. The appendages and (if allowed to execute) the ERP have determined that this is a permanent error.

The I/O request is terminated with the permanent error. The CSW contents and sense data in the IOB do not apply to the attempted operation. They apply to the previous operation attempted for any data set on the device. You can reissue the EXCP macro instruction to restart the channel program.

45
A program check or machine check occurred in IOS while processing the I/O request.

48
The channel program was purged.

4B
An error occurred during tape repositioning that was requested by the tape ERP.

4F
Error recovery routines were entered following a direct access error but are unable to read the home address or record 0.

51
Simulated error status. The appendages and, if allowed to execute, the ERP have determined that this is a permanent error. The I/O request is terminated with the permanent error. This code indicates that the device is in a permanent error state, boxed, or not connected. The code can indicate also that a missing interrupt was detected and that the I/O operation was terminated as a result of recovery operations by the missing interrupt handler.

This completion code appears only in the IEDB, if one was provided. It is translated to a 41 completion code prior to updating the ECB and IOBECBCC.

74
Simulated error status. This code is set as a result of attempting to start an I/O operation to a device that is in a permanent error state, boxed, or not connected. This code is also set if a missing interrupt was detected and the I/O operation was terminated as a result of recovery operations by the missing interrupt handler. The system has set this code temporarily in IOBECBCC and IEDB to invoke the ERP, if allowed to execute, and the appendages to determine if the error is permanent or correctable. If the error is determined to be permanent the system will change the value to 51. The system does not set the 74 code in the ECB.

7F
Normal I/O completion, but does not appear in the ECB.

Reason code byte
Reason code for the completion code.

Code
Meaning

0E
A zHPF channel program was specified but the device does not support zHPF. This reason code appears with completion code 41.

10
A zHPF channel program was specified, but the I/O request was terminated because a capability needed by the I/O request is not supported by the processor, device, or software. This reason code is presented if an EXCPVR request is issued when the processor and device do not support the zHPF incorrect length facility. This reason code appears with completion code 41.

Data Extent Block (DEB) Fields

The data extent block is constructed by the system when an OPEN macro instruction is issued for the DCB. You cannot modify the fields of the DEB, but you can examine them. The DEB is mapped by the IEZDEB macro. The DEB fields used for EXCP and EXCPVR are illustrated in Appendix A, “Control Blocks,” on page 431. You can find a complete view of DEB fields “Data Extent Block (DEB) Fields” on page 431.

EXCP and EXCPVR Appendages

An appendage is a routine that provides additional control over I/O operations. Using appendages, you can examine the status of I/O operations and determine the actions to be taken for various conditions. An appendage can receive control as shown in Table 42 on page 199.

Table 42. EXCP Appendages

Appendage	Description	When Called
ABE	Abnormal-end	Abnormal conditions
CHE	Channel-end	Channel-end, unit exception, wrong-length record
EOE	End-of-extent	DASD track address in I/O block outside allocated extent limits
PCI	Program-controlled interruption	When one or more PCI bits are on in a channel program
PGFX	Page-fix	Prior to SIO for EXCPVR requests
SIO	Start-I/O	Just prior to translating channel program

Appendages get control in supervisor state, protection key 0, receiving the pointers from the system described in the following table. The appendages receive control in 24-bit addressing mode and must return in the same mode. They get control enabled for I/O interrupts in the primary address space under an SRB. It does not operate in cross memory mode.

Register
Content

0
Points to the user's IOBE if one was provided as input to EXCP or EXCPVR. Otherwise 0 is passed to the appendage routine.

1
Points to the request queue element.

2
Points to the input/output block.

3
Points to the data extent block.

4
Points to the data control block.

6
Points to the seek address (MBBCCHHR) if control is given to an end-of-extent appendage.

The track address of the block reference (CCHH) may contain 28-bit cylinder numbers for devices with more than 65,520 cylinders. Showing nibbles it is in the form of CCCCcccH, where ccc represent bits 0-11 of the 28-bit cylinder number and CCCC represents bits 12-27 the 28-bit cylinder number. Use the TRKADDR macro to manipulate 16-bit and 28-bit cylinder numbers correctly.

7
Points to the unit control block (UCB) and always contains a clean 31–bit UCB address. If the DEB flag "DEB31UCB" is off, then the UCB address is captured below the 16 MB line. Whereas if DEB31UCB is on, then the address might point above the 16 MB line even though the appendages always are entered in 24–bit mode. The UCB address is captured by OPEN or EOV until EOV or CLOSE uncaptures it. If the DCBE option "LOC" was not set or defaulted to "BELOW", that is LOC=BELOW or not coded, or the "NON_VSAM_XTIOT" option of the DEVSUPxx member of PARMLIB was set or defaulted to "NO", that is NON_VSAM_XTIOT=NO or not coded. In other words, OPEN does not capture the UCB if LOC and NON_VSAM_XTIOT were specified as follows: LOC=ANY and NON_VSAM_XTIOT=YES.

13
Points to a 16-word area you can use to save input registers or data.

14
Points to the location in the system where control is to be returned following execution of an appendage. When returning control to the system, you can use displacements from the return address in register 14. Allowable displacements are summarized and described later for each appendage in Table 43 on page 200.

15
Points to the entry point of the appendage. When the PGFX appendage is entered, points to the SIO entry point.

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

- Register 9, if used, must be set to binary zeros before control is returned to the system. All other registers, except those indicated in the descriptions of the appendage, must be saved and restored if you use them. Table 43 on page 200 summarizes register conventions. Note that the need to save and restore registers applies to all eight byes in each register.
- No SVC instructions or instructions that change the status of the system (for example, WTO, LPSW, or similar privileged instructions) can be issued.
- Loops testing for the completion of I/O operations cannot be used.

The information here describes appendage types, with explanations of when they are entered, how they return control to the system, and which registers they can use without saving and restoring their contents. If you do not supply a particular appendage, or supply no appendage, the system acts as though that appendage had returned at offset 0 from register 14.

Table 43. Entry Points, Returns, and Available Work Registers for Appendages

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

Note: The register conventions for passing parameters from appendages to the system are described in the individual appendage descriptions.

Making Appendages Available to the System

Prior to execution, appendages must be members of either the SYS1.LPALIB or SYS1.SVCLIB data set. To put appendages into SYS1.LPALIB or SYS1.SVCLIB, link-edit them into these data sets after the system has been built. Each appendage must have an 8-character member name, the first six characters being IGG019 and the last two being anything in the range from WA to Z9. If your program runs in a V=R address space and uses a PCI appendage, the appendage and any routine that the PCI appendage refers

to must be placed in either SYS1.SVCLIB or the fixed link pack area (LPA). For information on providing a list of programs to be fixed in storage, see *z/OS MVS Initialization and Tuning Guide*.

The Authorized Appendage List (IEAAPP00)

If an unauthorized program opens a DCB to be used with an EXCP macro instruction, the names of any appendages associated with the DCB must be listed in the IEAAPP00 member of SYS1.PARMLIB. (An unauthorized program is one that runs in a protection key greater than 7 and has not been marked as authorized by the Authorized Program Facility.) Once you have added your appendages to SYS1.LPALIB or SYS1.SVCLIB after the system was built, you can add IEAAPP00 to SYS1.PARMLIB and put the names of the appendages in it with the IEBUPDTE utility or with another program that updates partitioned data sets. See the description of the IEAAPP00 parmlib member in *z/OS MVS Initialization and Tuning Reference*.

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

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

Note the following about the IEBUPDTE input:

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

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

```
//             JOB. . .
//             EXEC     PGM=IEBUPDTE,PARM='NEW'
//SYSPRINT     DD       SYSOUT=A
//SYSUT2       DD       DSN=SYS1.PARMLIB,DISP=SHR
//SYSIN        DD       *
./ ADD NAME=IEAPP00
PCIAPP Y1,
EOEAPP WA,
SIOAPP X1,
CHEAPP Z3,Z4,
ABEAPP Z2,Z4
./ ENDUP
/*
```

Note the following about the IEBUPDTE input:

- The command to IEBUPDTE is ADD but a replace occurs because PARM='NEW' is specified.
- All the appendage names that are to remain in IEAAPP00 are repeated.

- IGG019Z4 is both a CHE and an ABE appendage.

Start-I/O Appendage

Unless an ERP is in control, the system passes control to the SIO appendage just before the system translates and starts your channel program. It is called even if the channel program is not later translated. Your SIO appendages can build the channel program. The system does not test IOBSTART until after the SIO appendage returns. IOBSTART contains the virtual address of the start of the channel program.

The start I/O appendage may build a zHPF channel program or a CCW channel program. The caller does not have to set the IOBEZHPF bit prior to issuing the EXCPVR request. If the start I/O appendage builds a zHPF channel program, it should set the IOBEZHPF bit if it is not already set, and reset the IOBEFMT1 bit. Otherwise, it should reset the IOBEZHPF bit.

If the device is not enabled for zHPF or does not have the necessary capabilities, the start I/O appendage should either build a CCW channel program, or post the request in error and return to EXCP indicating that the I/O operation should be skipped (return +4).

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

- Reg. 14 + 0 — Normal return. The system should translate the channel program, if required, and initiate the I/O.
- Reg. 14 + 4 — Skip the I/O operation. The channel program is not initiated. The channel program is not posted complete. You can post the channel program complete by using the POST macro, as follows.
  - Set the completion code in register 10. This register is used to post the ECB.
  - Set register 11 to the ECB address in the IOB.
  - Issue the POST macro as shown in the example below:

    ```
    POST (11),(10),LINKAGE=BRANCH
    ```

    For more information on the POST macro, see *z/OS MVS Programming: Authorized Assembler Services Guide* and *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU*

Page Fix and EXCPVR Start I/O Appendage

This appendage is a combination of two independent appendages. The complete appendage is a reenterable subroutine with two entry points, one for the SIO appendage and one for the PGFX appendage.

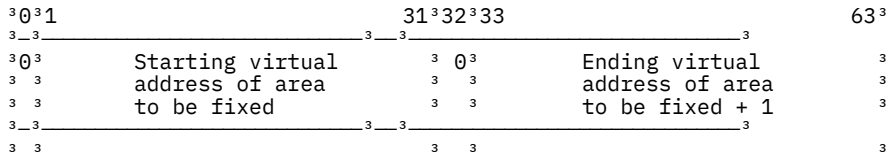
The SIO entry point is located at offset +0 in the SIO subroutine; from this entry point you might have an instruction to branch to any other location in the appendage. The entry point of the PGFX appendage is at offset +4 in the SIO subroutine. The address of offset +0, the SIO appendage entry address, is set in register 15 as the entry point of the PGFX appendage, allowing you to use the same entry linkage code for both entry points.

Note that you cannot fix pages that were allocated with CONTROL=UNAUTH on the IARV64 macro. Unauthorized programs cannot override this setting, but they can allocate 64-bit storage with the IARST64 macro.

PGFX Appendage

This appendage creates a list of the addresses of the areas that must be fixed to prevent paging exceptions during the execution of the current input/output (I/O) request. While this appendage can be entered more than once for one I/O request, each time it is entered it must create the same list of areas to be fixed. The appendage can use the 16-word save area pointed to by register 13. Registers 10, 11, and 13 can be used as work registers.

Each page-fix entry placed in the list by the appendage must have the following doubleword format:



On return from the PGFX appendage to the system (via the return address provided in register 14), register 10 must point to the first page-fix entry and register 11 must contain the number of page-fix entries in the work area. The system then fixes the pages corresponding to the areas listed by the PGFX appendage. The pages remain fixed until the associated EXCPVR request terminates.

If either the channel end appendage or the abnormal end appendage returns via the return address in register 14 plus 8, the PGFX appendage is not normally reentered. Instead, the SIO appendage is entered, and the page-fix list built by the PGFX appendage is still active. When a PURGE macro has been issued (for instance, when a storage swap has occurred), the PGFX appendage is entered after either the channel end appendage or the abnormal end appendage returns via the return address in register 14 plus 8. When I/O is restored, the PGFX appendage is entered. The page-fix list must be in page-fixed storage.

SIO Appendage

If you are using EXCPVR to execute a channel program and your channel program does not already contain real addresses, translate the virtual addresses in the operands of your channel program to central storage addresses. This should be done in the SIO appendage. If indirect data addressing is required, use the SIO appendage to build the indirect address lists (IDALs, MIDALs, or TIDALs) and turn on the appropriate indirect address list flag for the channel program.

You can use EXCPVR for VIO data sets with the following considerations for the SIO appendage:

- The use of IDAWs and an IDAL is not required.
- MIDALs and zHPF channel programs are not supported for VIO.
- Addresses in the CSWs and IDAWs must be virtual. They must not be converted to central storage addresses.

The UCBVRDEV bit in the UCBJBNR byte can be checked to determine if the data set is being processed with VIO.

When building an IDAL for CCW channel programs consider the following:

- Bit IOBEEIDA, described in "Input/Output Block Common Extension (IOBE) Fields" on page 193, specifies whether you are using 31-bit or 64-bit IDAWs. With 31-bit IDAWs, use 2 KB boundaries when determining whether a storage boundary is crossed for a CCW. With 64-bit IDAWs, use 4 KB boundaries.
- The LRA instruction returns a 31-bit central storage address regardless of whether you are in 24-bit or 31-bit addressing mode, but fails in those addressing modes if the central storage address is above 2 GB. If the central storage address is above 2 GB, you must either use the LRAG or STRAG instruction to convert the virtual address to a real address or use the LRA instruction (after first switching to 64-bit addressing mode). If your program switches to 64-bit addressing mode, you must ensure that you save and restore the high 32 bits of any register that you modify. If you fail to do this, unpredictable results can occur for other 64-bit programs in your address space.

See "IDAW Requirements for EXCP Requests" on page 161 for more information on creating IDAWs for EXCPVR requests.

Program-Controlled Interruption Appendage

This appendage is entered if the channel finds one or more program-controlled-interruption (PCI) bits on in a channel program. It can be entered as many times as the channel finds PCI bits on, or more often. Before the appendage is entered, the contents of the subchannel status word are placed in the channel status word field of the input/output block.

Note that PCI and PCI appendages are not supported for zHPF channel programs.

A PCI appendage is reentered if an ERP is retrying a channel program in which a PCI bit is on. The IOB error flag is set when the ERP is in control (IOBFLAG1 = X'20'). (For special PCI conditions encountered with command retry, see “Command Retry Considerations” on page 170.)

To post the channel program from a PCI appendage to an EXCP request (EXCP V=V), use the procedure described in “SIO Appendage” on page 203.

If the step is running ADDRSPC=REAL (V=R) and an authorized program issued the EXCP request or if an EXCPVR request was issued, the PCI appendage uses central storage addresses. Use the following procedure to post the channel program from the PCI appendage. For more information on the POST macro, see *z/OS MVS Programming: Authorized Assembler Services Guide* and *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU*.

The POST macro is coded as follows:

```
POST  ecbaddr,compcode,ASCB=addr,ERRET=addr,ECBKEY=key,
      LINKAGE=BRANCH,MEMREL=NO
```

The ERRET routine address must point to a BR 14 instruction. This instruction must be in storage addressable from any address space (for example, CVTBRET) and addressable by 24 bits.

Note: If you specify the ASCB parameter with MEMREL=NO, only registers 9 and 14 are restored when returning from the POST macro.

The following procedure posts the channel program from the PCI appendage.

1. Save necessary registers, because only registers 9 and 14 are restored on the return from the POST macro.
2. Set the ECB key in register 0.
3. Set the 4-byte completion code in register 10.
4. Set the ECB address in register 11.
5. Set the error routine address in register 12, by setting it to address of CVTBRET and turn on the high-order bit (X'80') of the high-order byte.
6. Set the ASCB address in register 13. If you do not have the ASCB address, you can use the following procedure to obtain the ASCB address:
   a. Issue the EPAR instruction to obtain the ASID. For information on the EPAR instruction, see *z/Architecture Principles of Operation.*
   b. Issue the LOCASCB macro to obtain the ASCB address. The LOCASCB macro is documented in *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU*.
7. Issue the POST macro, as shown in the following example:

   ```
   POST (11),(10),ASCB=(13),ERRET=(12),ECBKEY=(0),LINKAGE=BRANCH
   ```

8. On return, reestablish necessary registers.

To return control to the system for normal operation, use the return address in register 14.

End-of-Extent Appendage

If an end-of-cylinder or file-protect condition occurs, the system updates the seek address to the next higher cylinder or track address and retries the request. If the new seek address is within the current extent for the data set, the request is executed; if the new seek address is not within the current extent for the data set, the EOE appendage is entered. To try the request in the next extent, move the new seek address to the location pointed to by register 6.

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

The end-of-extent (EOE) appendage is entered when the seek address specified in the input/output block is outside the allocated extent limits indicated in the data extent block.

The simplest way to compare two track or block addresses to see their relative positions on the volume is to use the CLC instruction; however that technique works only if both addresses (CCHH or CCHHR) contain cylinder numbers of less than 65536. To handle any addresses that are more or less than cylinder 65536, use the TRKADDR macro with the COMPARE option. See “Compare two track addresses (TRKADDR COMPARE)” on page 301.

You can use the following optional return addresses:

- Contents of register 14 to return control to the system, causes the ABE appendage to be entered. The system places an end-of-extent error code (X'42') in the ECB code field of the input/output block for subsequent posting in the ECB.
- Contents of register 14 plus 4: The channel program is posted complete with X'42'. The ABE appendage is not reentered for this request.
- Contents of register 14 plus 8: The request is tried again.

Registers 10 through 13 in an EOE appendage can be used without saving and restoring their contents.

Abnormal-End Appendage

To determine the method the system uses to handle an abnormal condition use the abnormal-end (ABE) appendage. The following information explains how to use the ABE appendage.

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

- If IOBECBCC is set to X'41', this appendage was entered because of a unit exception or wrong-length record indication or both. The system previously called the channel-end appendage, if present. For further information on these conditions, see “Channel-End Appendage” on page 206.
- If the IOBECBCC is set to X'42', this appendage was entered because of an out-of-extent error. The system previously called the end-of-extent appendage, if present.
- If this appendage is entered with IOBECBCC set to X'4B', the tape error recovery procedure (ERP) either encountered an unexpected load point, or found zeros in the command address field of the CSW.
- If the IOBECBCC is set to X'7E', the appendage was first entered because of an intercept condition. If it is then determined that the error condition is permanent, the appendage will be reentered with the IOBECBCC set to X'44'. The intercept condition signals that an error was detected at device end after channel end on the previous request.
- If the IOBECBCC was set to X'48', the appendage was entered because of an EXCP being issued to an already purged DCB. This applies only to related requests.
- If the appendage is entered with IOBECBCC set to X'7F', it might be because of a unit check, program check, protection check, channel data check, channel control check, interface control check, or chaining check. If the IOBECBCC is X'7F', it is the first detection of an error in the associated channel program. If the IOBIOERR flag (bit 5 of the IOBFLAG1) is on, the IOBECBCC field will contain X'41', X'42', X'48', X'4B', or X'4F', indicating a permanent I/O error.
- If the ending address is zero or the subchannel status byte in the IOB (IOBCSTAT) shows any of the following errors: program check, protection check, channel data check, channel control check, interface control check, or chaining check, and your abnormal end appendage determines that the ERP has not yet run, then do not modify IOBSTART. This lets the ERP try to recover. If the ERP has completed and one or more of these six bits is on or the address is zero, then the status of the channel program is not known.

To determine if an error is permanent, check the IOBECBCC field of the IOB for a X'4x' completion code.

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

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

- Contents of register 14 plus 4: The channel program is not posted complete, but its request element is made available. You can post the channel program by using the calling sequence described under the SIO appendage.
- Contents of register 14 plus 8: The channel program is not posted complete, and its request element is placed back on the request queue to be retried. Reinitialize the IOBFLAG1, IOBFLAG2, and IOBFLAG3 fields of the input/output block and set the IOBERRCT field to zero. As an added precaution, clear the IOBSENS0, IOBSENS1, and IOBCSW fields.

  The appendage can request that a different type of channel program be started. For example:

  – The original request was for a zHPF channel program but the device is no longer enabled for zHPF or does not support the zHPF capabilities required by the new channel program.
  – The original request was for a non-zHPF channel program but the new I/O request allows a zHPF channel program to be used.

  The channel program type may be changed from non-zHPF to zHPF only if the caller passed an IOBE to EXCP. If the channel program type is changed, the appendage must set or reset the IOBEFMT1 and IOBEZHPF bits correctly to reflect the type of channel program. For example, if the original channel program was a format-1 CCW channel program and the new channel program is a zHPF channel program, then the IOBEFMT1 bit must be reset and the IOBEZHPF bit must be set.

  The system will call appendages, beginning with the SIO appendage, as if this were a new EXCP or EXCPVR request. Note that the EXCPVR page fix appendage will not be called again.
- Contents of register 14 plus 12: The channel program is not posted complete, and its request element is not made available. (Use this return only if the appendage has passed the request queue element to the exit effector for use in scheduling an asynchronous routine.)

Registers 10 through 13 in an ABE appendage can be used without saving and restoring their contents.

Channel-End Appendage

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

By using the return address in register 14 to return control to the system, the channel program is posted complete, and its request element is made available. In the case of unit exception or wrong-length record, the system sets on the IOBIOERR flag (X'04') in IOBFLAG1 before calling the channel-end appendage. If the appendage returns with that bit still on, the system calls the ERP. If the bit still is on after the ERP, the channel program is posted with X'41'. The CSW status can be obtained from the IOBCSW field.

If the appendage takes care of the wrong-length record or unit exception or both, it can turn off the IOBIOERR (X'04') flag in IOBFLAG1 and return normally. The event is posted complete (completion code X'7F' under normal conditions, taken from the high-order byte of the IOBECBCC field). If the appendage returns normally without resetting the IOBIOERR flag to zero, the request is routed to the associated device ERP. If the ERP did not correct the error, the ABE appendage is entered with the completion code in IOBECBCC set to X'41'. (See the first bullet in “Abnormal-End Appendage” on page 205.)

You can use the following optional return addresses:

- Contents of register 14 plus 4: The channel program is not posted complete, but its request element is made available. You can post the channel program by using the calling sequence that is described under the SIO appendage. This is especially useful to post an ECB other than the ECB in the input/output block.
- Contents of register 14 plus 8: The channel program is not posted complete, and its request element is placed back on the request queue so that a channel program at the same or different address can be executed. For correct execution of the channel program, reinitialize the IOBFLAG1, IOBFLAG2, and IOBFLAG3 fields of the input/output block and set the error counts field to zero. As an added precaution, clear the IOBSENS0, IOBSENS1, and IOBCSW fields. You can change IOBSTART before

returning. If the device is DASD, you can change the IOBSEEK field but the first byte must refer to an extent with the same UCB address. The system calls appendages, beginning with the SIO appendage, as if this were a new EXCP or EXCPVR request. Note that the EXCPVR page fix appendage is not called again.

The appendage can request that a different type of channel program be started. For example:

- The original request was for a zHPF channel program but the device is no longer enabled for zHPF or does not support the zHPF capabilities required by the new channel program.
- The original request was for a non-zHPF channel program but the new I/O request allows a zHPF channel program to be used.

The channel program type may be changed from non-zHPF to zHPF only if the caller passed an IOBE to EXCP. If the channel program type is changed, the appendage must set or reset the IOBEFMT1 and IOBEZHPF bits correctly to reflect the type of channel program. For example, if the original channel program was a format-1 CCW channel program and the new channel program is a zHPF channel program, then the IOBEFMT1 bit must be reset and the IOBEZHPF bit must be set.

- Contents of register 14 plus 12: The channel program is not posted complete, and its request element is not made available. (Use this return only if the appendage passed the request queue element to the exit effector for use in scheduling an asynchronous routine. For information on asynchronous exit routines, see *z/OS MVS Programming: Authorized Assembler Services Guide*.)

Registers 10 through 13 in a CHE appendage can be used without saving and restoring their contents.

Converting a Relative Track Address to an Actual Track Address

Convert a relative track address to the actual address by using a resident system conversion routine that can be called in 24 or 31 bit mode.

The conversion routine has two entry points. One, at the address in the CVTPCNVT field of the communication vector table (CVT), is intended for data sets with up to 65535 tracks. The other, at offset +12 from the address in CVTPCNVT, is intended for any data set. Call the conversion routine with one of the following instructions:

```
BALR 14,15 Call +0 entry point for track conversion
BASR 14,15 Call +0 entry point for track conversion
BAL 14,12(,15) Call +12 entry point for track conversion
BAS 14,12(,15) Call +12 entry point for track conversion
```

IBM does not provide a macro for this conversion routine's invocations.

The address of the CVT is in storage location 16 (field FLCCVT of the PSA data area, mapped by macro IHAPSA).

The conversion routine does all its work in general registers. Load registers 0, 1, 2, 14, and 15 with input to the routine. Register usage is as follows:

Table 44. Registers and Their Use for Converting Relative to Actual

Register	Use
0	Must be loaded with a 4-byte value of the relative track number. If entered at CVTPCNVT+0, this value must be in the form TTR*n*, where: **TT** The track number relative to the beginning of the data set **R** The block identification on that track ***n*** If the DEB is for a partitioned concatenation, (the DSORG field in the DCB indicates PO), then supply the concatenation number for the data set. A value of 0 indicates the first data set, 1 indicates the second data set and so forth. If your partitioned concatenation includes PDSEs or z/OS UNIX directories, each of them is represented by a dummy extent in the DEB. EXCP and EXCPVR are not valid for those extents. If the data set is not concatenated or the concatenation is not partitioned, set to 0. If entered at CVTPCNVT+12, this value must be in the form TTTR, where: **TTT** The track number relative to the beginning of the data set **R** The block identification on that track This entry point is intended for any type of data set.
1	Must be loaded with the address of the data extent block (DEB) of the data set. Each DEB resides below 16 MB. When you call the +0 entry point, the called routine clears the high order byte of this register. When you call the +12 entry point, this byte must contain X'00'. Note that if the DEB31UCB bit is zero, the UCB address field is three bytes in DEBUCBA. If the DEB31UCB bit is one, the UCB address field is four bytes in DEBUCBAD.
2	Must be loaded with the address of an 8-byte area that is to receive the actual address of the block to be processed. The converted address is of the form MBBCCHHR, where: **M** Indicates which extent entry in the data extent block is associated with the direct access program. (0 indicates the first extent, 1 indicates the second, and so forth) **BB** Two bytes of zeros **CC** Low order 16 bits of the cylinder number. The cylinder number is 28 bits on all currently supported DASD. **HH** The actual track number in the low order four bits and the high order twelve bits of the cylinder number in the high order twelve bits. **R** The block number.
3–8	Not used by the conversion routine.
9–12	Used by the conversion routine and not restored.
13	Used by the conversion routine and not restored. If you call the +12 entry point, the high order three bytes must contain zero and the low order byte must be as set as follows: • If the DEB is for a partitioned concatenation, (the DSORG field in the DCB indicates PO), then supply the concatenation number for the data set. A value of 0 indicates the first data set, 1 indicates the second data set and so forth. • If the data set is not concatenated or the concatenation is not partitioned, set to 0.

Table 44. Registers and Their Use for Converting Relative to Actual (continued)

Register	Use
14	Must be loaded with the address to which control is to be returned after execution of the conversion routine.
15	Used by the conversion routine as a base register and must be loaded with the address where the conversion routine is to receive control (from field CVTPCNVT of the CVT).

Return Codes from the Relative to Actual Conversion Routine

When control is returned to your program, register 15 contains one of the following return codes:

Table 45. Relative to Actual Conversion Routine Return Codes

Return Code	Description
0 (X'00')	Successful conversion.
4 (X'04')	The relative track address converts to an actual track address outside the extents defined in the DEB.
8 (X'08')	Internal access method control blocks are invalid. Call IBM service if the control blocks should be valid.
12(X'0C')	Internal access method control blocks are invalid. Call IBM service if the control blocks should be valid.
16 (X'10')	Passed concatenation number is too big for the data set.
20 (X'14')	The combination of the DEBNMTRK and DEBNMTRKHI fields in the last extent shows that the input track number points into this extent, but the calculated CCHH does not lie within this extent.

Converting an Actual Track Address to a Relative Track Address

Convert an actual track address to a relative track address by using a resident system conversion routine that can be called in 24 or 31 bit mode.

The conversion routine has two entry points. One, at the address in the CVTPRLTV field of the communication vector table (CVT), is intended for data sets with up to 65535 tracks. The other, at offset +12 from the address in CVTPRLTV, is intended for any data set. Call the conversion routine with one of the following instructions:

```
BALR 14,15 Call +0 entry point for track conversion
BASR 14,15 Call +0 entry point for track conversion
BAL 14,12(,15) Call +12 entry point for track conversion
BAS 14,12(,15) Call +12 entry point for track conversion
```

IBM does not provide a macro for this conversion routine's invocations.

The conversion routine will return either TTR0 (if entered at CVTPRLTV+0) or TTTR (if entered at CVTPRLTV+12), where TTR0 and TTTR are as described in Table 44 on page 208. If the actual track address is in a PDS in a partitioned concatenation, the returned relative track address (TTR0 or TTTR) is relative to the beginning of that PDS. It is not relative to the beginning of all data sets in the concatenation. The conversion routine does not return an indication of which data set the track is in. This means that the low order byte of the returned TTR0 is zero and not as described in that table.

The address of the CVT is in storage location 16 (field FLCCVT of the PSA data area, mapped by macro IHAPSA).

The conversion routine does all its work in general registers. Load registers 1, 2, 14, and 15 with input to the routine. Register usage is as follows:

Table 46. Registers and their use for converting actual to relative

Register	Use
0	Loaded with the resulting TTR0 or TTTR to be passed back to the caller. These two formats are as described for register 0 in Table 44 on page 208.
1	Must be loaded with the address of the data extent block of the data set. Each DEB resides below 16 MB. When you call the +0 entry point, the called routine clears the high order byte of this register. When you call the +12 entry point, this byte must contain X'00'.
2	Must be loaded with the address of an 8-byte area containing the actual address to be converted to a TTR0 or TTTR. The actual address is of the form MBBCCHHR.
3–8	Not used by the conversion routine.
9–13	Used by the conversion routine and not restored.
14	Must be loaded with the address to which control is to be returned after execution of the conversion routine.
15	Used by the conversion routine as a base register and must be loaded with the content of field CVTPRLTV of the CVT.

Return Codes from the Conversion Routine

When control is returned to your program, register 15 contains one of the following return codes:

Table 47. Actual to Relative Conversion Routine Return Codes

Return Code	Description
0 (X'00')	Successful conversion.
4 (X'04')	CCHH is outside of extent M. Returned relative track number is invalid. If you called the +12 entry point, then it has set the output R byte to X'FE'.
8 (X'08')	Internal access method control blocks are invalid. Call IBM service if the control blocks should be valid.
12(X'0C')	Internal access method control blocks are invalid. Call IBM service if the control blocks should be valid.
16 (X'10')	Passed extent number M is too big for the data set. If you called the +12 entry point, then it has set the output R byte to X'FE'.
20 (X'14')	The calculated relative track number is too large to be returned. If you called the +0 entry, then the track number exceeds X'FFFF', so the routine returned X'FFFFFE00'. If you called the +12 entry point, then the track number exceeds X'FFFFFF', so the routine returned X'FFFFFFFE'. The likely causes of this are that the DEB is not valid or (for entry +0) the data set is a large format data set, but the track location is more than X'FFFF' tracks into the data set.

Note: For return codes 4, 16 and 20 with the +12 entry point, this routine returns a different value for the R byte (X'FE') than was passed to it.

Using the IECTRKAD Callable Service or the TRKADDR Macro

The IECTRKAD callable service or the TRKADDR macro can be used to perform the following operations on both 16-bit and 28-bit cylinder addresses:

- Calculate the relative track number on the volume
- Compare two track addresses
- Extract the 28-bit cylinder number
- Extract the 4-bit track number
- Increment the track address by one track and increment the cylinder number if necessary.

- Normalize cylinder number to permit comparing one *cchh* against another
- Convert a relative track number to a 28-bit cylinder address
- Set the cylinder number in a 28-bit track address
- Convert a normalized track address into an absolute 28-bit track address.

See "Call for converting and comparing 28-bit cylinder addresses (IECTRKAD)" on page 336 and "Perform calculations and conversions with track addresses (TRKADDR macro)" on page 300 for more information.

Obtaining the Sector Number of a Block on an RPS Device

For programs that can be used with both RPS and non-RPS devices, test the UCBRPS bit (bit 3 at offset 17 of the UCB) to determine whether the device has rotational position sensing. If the UCBRPS bit is off, do not issue a channel program with a Set Sector command to the device. The address of the sector conversion routine's entry point is in the CVT0SCR1 field of the CVT. The IBM DS8000® and later storage subsystems accept the Set Sector command, but the command has no effect.

Your program can call the conversion routine by issuing a BASR 14,15 or BAS 14,16(,15) instruction. If you are passing the track balance to the routine, invoke the routine using a BAS 14,8(15) or BAS 14,20(,15). If you are computing the sector value on modulo devices with variable length records, pass the track balance to the sector convert routine.

The sector convert routine can be called in 24-bit or 31-bit mode at any of its entry points. When calling in 24-bit mode all addresses must point below the 16 MB line.

Note:

1. The sector convert routine does not support PDSEs or extended format data sets. Using the sector convert routine against a PDSE or extended format data set returns results that are inconsistent with the physical record format.
2. The sector convert routine does not support data sets that are not DASD or do not have a UCB. Examples include tape, dummy data sets (DD DUMMY), spooled data sets (*, DATA or SYSOUT), TSO terminals and z/OS UNIX files. You will receive unpredictable results if you use the conversion routine with a data set or file that is not on DASD or does not have a UCB.

For RPS devices, the conversion routine does all its work in general registers. Load registers 0, 2, 14, and 15 with input to the routine. Register usage is as follows:

Table 48. Registers and Their Use for A Sector Convert Routine

Register	Use
0	For fixed, standard blocks or fixed, unblocked records not in a partitioned data set: Register 0 must contain a 4-byte value in the form XXKR, where: **XX** A 2-byte field containing the physical block size **K** A 1-byte field containing the key length **R** A 1-byte field containing the number of the record for which a sector value is desired. To indicate fixed-length records, turn off (set to 0) the high-order bit of register 0. Passing the track balance: Register 0 must contain a 4-byte value of the track balance of the record preceding the required record. For all other cases: Register 0 must contain a 4-byte value in the form BBIR, where: **BB** The total number of key and data bytes on the track up to, but not including, the target record. To indicate variable-length records, turn on (set to 1) the high-order bit of register 0. **I** A 1-byte key indicator (1 for keyed records, 0 for records without keys) **R** A 1-byte field containing the number of the record for which a sector value is desired.
1	Not used by the sector-convert routine.
2	When called at offset 0 or 8, must contain a 4-byte field where: • The first byte is the UCB device type code for the device (obtainable from UCB+19). • The remaining 3 bytes are the address of a 1-byte area that is to receive the sector value. If called at offset 16 or 20, must contain the address of a 1-byte area that is to receive the sector value.
3-8	Not used.
9-10	Used by the convert routine and not saved or restored.
11	If called at offset 0 or 8, not used. If called at offset 16 or 20, contains 1-byte UCBTBYT4 code in the low-order byte. The remaining three bytes must contain zero.
12,13	Not used.
14	Must be loaded with the address in which control is to be returned after execution of the sector conversion routine.
15	Used by the conversion routine as a base register and must be loaded with the address of the entry point to the conversion routine (from field CVT0SCR1 of the CVT).

Encrypting and Decrypting with the IGGENC Macro

When you bypass the access methods, you can use the IGGENC macro to encrypt and decrypt data such that the result is compatible with how the access methods encrypt records. This section describes how to use IGGENC with basic format and large format data sets that are SMS-managed. Data set encryption is described further in *z/OS DFSMS Using Data Sets*.

Encryption and decryption require the functions of Integrated Cryptographic Service Facility (ICSF). One of the ICSF functions is to manage access to your encryption key. You can use a cryptographic service such as ICSF to generate a random encryption key but you cannot see your encryption key. You provide a name for the key and you use that name. The name is called a "key label". The key label is not a secret but its key is a secret even from the owner of the key.

Your program can test the DFASEQENCRYPT bit in the DFA to determine whether the IGGENC function is installed and functional. See the IHADFA mapping macro. If the basic cryptographic facility is not functional, IGGENC will give a reason code with the two low order bytes as X'611' or X'081x'. See Table 58 on page 224.

These are the functions that you can perform with the IGGENC macro:

- Connect to the encryption function.
- Encrypt or decrypt one or more blocks of data. These require that your program either call the connect function first or open an encrypted basic or large format data set.
- Disconnect from the encryption function.

IGGENC description

Programming environment

The requirements for the caller are described in Table 49 on page 214.

Table 49. Requirements for the caller

Environmental	Requirement
Authorization	Problem state or supervisor state and any PSW key. Calls after the connect call must be in the same protection key or be in key 0.
Dispatchable unit mode	The connect and disconnect functions require task mode. The encrypt and decrypt functions can be called in task or SRB mode.
Cross memory mode	PASN=HASN=SASN
AMODE	24-bit, 31-bit or 64-bit. If the invocation is in 64-bit, you must precede the invocation with an invocation of the SYSSTATE macro with AMODE64=YES.
ASC mode	Primary
Interrupt Status	Enabled for I/O and external interrupts.
Locks	Holding the local lock of the home address space is optional. When the caller holds that lock during encrypt or decrypt calls, ICSF captures statistics but does not write them out until a call without the local lock or when the disconnect function is called. The system calls EXCP appendages while holding the local lock.

Programming Requirements

Encryption and RACF Security

In order to encrypt or decrypt with the key label, your program must have at least RACF read authority to the key label. There are two exceptions to this requirement for authorization to the key label:

1. If the system is bypassing security for your job step because the JSCBPASS bit is on. It will be on if your job step program is in the system's program properties table (PPT) and the entry has the NOPASS attribute. The program properties table is defined by the SCHEDxx member of SYS1.PARMLIB.
2. If your program has APF authorization, supervisor state or system key and either of these is true:
   - Your program passed a DCB to the encrypt or decrypt function and the DCB points to a DCBE with BYPASS_AUTH=YES coded or
   - Your program called the IGGENC macro for the connect function with an options block that has the IGGENCBypassAuth bit on.

If the ICSF installation options specifies CHECKAUTH(YES), the caller must also have authority to the CSFSERV class with profile CSFKRR2.

Note: CHECKAUTH(NO) is the default.

Testing for Data Set Encryption

Testing Before or After Opening the Data Set

To test whether a data set is encrypted, your program can read the DSCB with the OBTAIN or CVAFDIR macro as described in this book. If the DS1ENCRP bit is on, the data set is encrypted.

- If the DS1PDSE bit also is on, then the data set is a PDSE and cannot be used with EXCP. (If the DS1PDSEX bit also is on, it is an HFS data set and the data set cannot be encrypted.)
- If the DS1STRP bit also is on, then the data set is extended format (possibly also compressed format) and cannot be used with EXCP.

- If neither of the above two bits is on and the DS1DSORG field has the DS1DSGPS bit on or the DCBDSORG field is empty, then it means that the data set is a basic format or large format data set. If the DS1LARGE bit is on, it is a large format data set.

Testing for encryption

If your program provides a DCBE, your optional DCB open exit routine can test the DCBE to see whether the data set has the encryption attribute. You can do the same test after OPEN.

The bit to test is DCBE_DS_ENCRYPTION. It means that the data set is encrypted. The data set is a PDSE or is basic, large or extended format.

If your program is reading a sequential concatenation, then this bit is for the current data set.

The system sets this bit on or off as appropriate for each data set. If your program turns on the unlike attributes bit, DCBOFPPC, then the system will call your DCB open exit routine at the beginning of each data set. This allows your program to adjust to each data set. If you do not turn DCBOFPPC on, then the system calls your DCB open exit routine only for the first data set. In this case you might want to provide an EOV exit routine. The system will call it at the beginning of each volume, including the first volume of each data set after the first. This also allows your program to adjust to encryption differences.

Even if two data sets have the same key label, they will encrypt differently. If you pass the DCB to IGGENC for encryption and decryption, the system will take care of this difference in encryption. Instead if you call the IGGENC CONNECT function, you must disconnect it at the end of each data set and call IGGENC CONNECT for the next data set.

Testing after you open the DCB

Learning the maximum block length

For maximum compatibility with the access methods and for maximum reliability in case of a program adding or removing blocks, tracks or volumes, your encrypted basic or large format data sets should have block prefixes. Prefixes are eight bytes long.

The DS1BLKL field in the DSCB contains the maximum block length for the data set. It does not include the length of the block prefix.

Other fields that contain the maximum block size for a data set and do not include the length of the prefix are:

- The JFCBLKSI field in the JFCB.
- The DCBBLKSI field in the DCB for BSAM and QSAM. It is two bytes. This field is not defined in an EXCP DCB.
- The DCBEBLKSI field in the DCBE for any access method. It is four bytes. Your EXCP program can use the field.
- The ARAXLBKS field as mapped by the IHAARA macro as returned by the RDJFCB macro. It is eight bytes.

If you issue the DEVTYPE macro with INFO=AMCAP, it returns the maximum block size for the device. It does not take encryption into account.

Learning whether the data set is encrypted and the block prefix length

In addition to testing the DSCB, there are two ways to learn whether the data set is encrypted:

- In your optional DCB open exit routine, test the DCBE_DS_ENCRYPTION bit in the DCBE. It will remain valid after the DCB is open. In a sequential concatenation, it will remain value (on or off) for each data set.
- Issue the ISITMGD macro after opening the DCB. To learn whether the data set is encrypted, test the ISMENCRP bit.

To learn the length of the block prefix after the DCB is open, code VERSION=2 on the ISITMGD macro and then test the ISMBPRFX byte. For further information about ISITMGD, see *z/OS DFSMS Macro Instructions for Data Sets*.

Opening an encrypted data set

Opening an encrypted data set with a BSAM or QSAM DCB does not require any source code changes or reassembly. If your program issues the EXCP, EXCPVR or XDAP macro for an encrypted data set, the DCB must point to a DCBE that has the DSENCRYPT=OK option set. It also must meet either of these requirements:

- The DCB has a MACRF value as required for BSAM or QSAM. The system checks the DCBE during each invocation of the EXCP or XDAP macro. (You can use the DCB as a task library but in that case it cannot be encrypted.)
- The DCB has MACRF=E to identify it as a DCB that is only for EXCP, EXCPVR or XDAP. The system checks the DCBE during open.

Block prefixes with data set encryption

Statistical analysis to break encryption was discovered in World War II. Block prefixes in data sets are an important part of preventing statistical analysis.

When the system creates the data set's entry in the catalog, it stores a large random number in the catalog entry. The system concatenates that large random number with the prefix for each disk block to create what is called the tweak value to encrypt that disk block. The content of each prefix should be different. This ensures that each block encrypts differently and each copy of a data set will encrypt differently (unless the copying program bypasses the access method and copies the tracks. DFSMSdss is one such program that copies the tracks.). If you write the same record multiple times, the encrypted version of each copy differs unpredictably.

With EXCP you can choose to write some blocks without encryption where the indicator is stored in the prefix. but those blocks still require prefixes.

To learn whether the data set has prefixes, your program can issue the ISITMGD macro after opening the DCB.

Do not encrypt or decrypt the block prefixes. You do encrypt and decrypt the BDWs, block descriptor words, in variable-length blocks.

You pass the block prefixes to the IGGENC macro with the BLKIDLIST parameter.

Table 50. Contents of the block prefix for an encrypted data set as mapped by the IGGEBPFX macro

Offset	Type	Length	Name	Description
0	DSECT		EBP	Encryption block prefix in basic or large format sequential data set
	Bits	1	EBPFlag1	Flags
	1...		EBPEncrypted	This block is encrypted
	.xxx xxxx			Reserved, should be zero
1	Binary	2		Reserved, should be zero
3	Integer	4	EBPTTTT	Relative track number across all volumes, first is zero
7	Integer	1	EBPR	Block number on track, first is 1

Encrypted blocks without prefixes

Creating data sets without prefixes should be rare and controlled. You can create an encrypted data set without prefixes if all of these are true:

- Programs used to modify the data set can be controlled and are enhanced to understand how to read and write such a data set without block prefixes.
- All blocks in the data set are assumed to be encrypted.

- The relative location of every physical block within the data set can NEVER change.
  - Blocks can NEVER be added or deleted within a track, other than from the end.
  - Tracks can NEVER be added or deleted from the data set, other than from the end of the last volume that contains data.
- You understand that programs that use BSAM or QSAM will not be able to open this type of data set. An attempt to do so will result in an abend 213-99, with diagnostic information at the end of the message to identify the reason.
  - Not allowing access by a BSAM or QSAM application will prevent the potential scenario where standard copying programs copy the data set to an encrypted data set. In this case, the target of the copy would then contain block prefixes.

You can set a DCBE option to inform the system that the data set does not have block prefixes and your program can support it. The DCBE option is the DCBEDSENCNP bit. The DCBE macro does not have a keyword for this. This has an effect when the first OPEN macro option is other than INPUT (thus, includes OUTPUT, OUTIN, INOUT, and UPDAT) and the DCB is for EXCP. The system will save the indicator in the ENCRYPTA portion of the encryption cell in the catalog entry for the data set. This DCBE option is required for both reading and writing with EXCP. An attempt to subsequently open a data set of this type with EXCP for reading or writing without the DCBE DCBEDSENCNP option will result in an abend 213-99, with diagnostic information at the end of the message to identify the reason.

When your program calls the IGGENC macro to encrypt or decrypt, it must create and pass the appropriate block prefixes even though they do not exist on the device.

Reading and writing encrypted data

When you call the IGGENC macro to encrypt or decrypt, you pass the prefix for each block so that IGGENC can encrypt or decrypt the blocks correctly.

You might choose for your EXCP program to read and write each block in virtual storage areas where the prefix and encrypted data are contiguous with each other or they might be non-contiguous. When you call the IGGENC macro, you still pass the prefix address list separately from the block address list.

If you choose to read and write the prefixes in areas that are not contiguous with the encrypted data, then your channel program must use one of these methods:

- Data chaining in a CCW channel program.
- MIDAWs in a CCW channel program.
- TIDAWs in a transport mode (zHPF) channel program.

Restrictions

If your program is running in 64-bit addressing mode, the parameter list and all parameters can reside above the 2 GB bar. If your program is running in 24-bit or 31-bit mode, addresses in the parameter list are treated as 31-bit and addresses in the lists that the parameter list points to are treated as 64-bit.

Keys

Encrypted data sets cannot have hardware keys. That means that KEYLEN in the DSCB, DCB, DD statement or dynamic allocation must be 0.

Minimum length of a block

The minimum length of an encrypted block is 16 bytes.

Reading part of a block

It is valid for your program to suppress data transfer with data chaining to skip over part of a block but that technique will not work with an encrypted data set because the encryption algorithm requires data from the beginning of the block.

If you read from the beginning of a block, you can read less than the whole block but there are special restrictions to decrypt it. These principles apply both when you intentionally read less than the whole block and when the device fails to return the whole block. In both cases you might want to pass the data length to IGGENC rounded up to a multiple of 16.

This is because the XTS encryption mode always encrypts groups of 16 bytes. You can encrypt any number of bytes that is at least 16. If you pass a data length that is not a multiple of 16, the algorithm encrypts the last 16 bytes overlapped with previous bytes. Let us say that the actual block length is 80 bytes, not counting the prefix. Let us say that your channel program read only the first 50 bytes. If you pass a data length of 50 to IGGENC, the last two bytes will be returned as garbage and this will not be detected unless you have some way to test validity. This is because the algorithm will incorrectly decrypt the last two bytes. If you want to decrypt those two bytes, you must read at least 64 bytes (the next multiple of 16) and pass those 64 bytes to IGGENC.

In this example if the number of bytes that your program read is 65 to 79, you cannot decrypt the bytes after the first 64 bytes. You can pass those bytes to IGGENC for decryption but those decrypted bytes will contain garbage. In this example you should read the whole 80 bytes.

Input register information

Before issuing the IGGENC macro, you do not have to place information into any register unless you use it in register notation for a particular parameter or use it as a base register.

You do not have to point register 13 to a register save area before issuing the IGGENC macro.

Output register information

Table 51. Output register information

Register	Register Contents
0	Reason code (eight bytes even if your program is not running in 64-bit). Also returned as the third parameter.
1	Used as a work register by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code (four bytes). Also returned as the second parameter.
Interrupt Status	Enabled for I/O and external interrupts.
Locks	Holding the local lock of the home address space is optional. When the caller holds that lock during encrypt or decrypt calls, ICSF captures statistics but does not write them out until a call without the local lock or when the disconnect function is called. The system calls EXCP appendages while holding the local lock.

Syntax of IGGENC standard invocation

Table 52. Syntax of IGGENC standard invocation

Syntax	Description
Function (positional parameter)	CONNECT, ENCRYPT, DECRYPT or DISCONNECT. It can be in mixed case such as Connect.
BLKIDLIST=	*list addr*: A-type address, or register (2) - (12).
BUFCOUNT=	*count addr*: A-type address, or register (2) - (12).

Table 52. Syntax of IGGENC standard invocation (continued)

Syntax	Description
BUFLENLIST=	*list addr*: A-type address, or register (2) - (12).
DCB=	*dcb addr*: A-type address, or register (2) - (12).
ENCRYPTA=	*encrypta addr*: A-type address, or register (2) - (12).
INBUFLIST=	*list addr*: A-type address, or register (2) - (12).
OPTIONS=	*area addr*: A-type address, or register (2) - (12).
OUTBUFLIST=	*list addr*: A-type address, or register (2) - (12).
RETCODE=	*return code addr*: A-type address, or register (2) - (12).
RSNCODE=	*reason code addr*: A-type address, or register (2) - (12).
TCBADDR=	*tcb addr*: A-type address, or register (2) - (12).
TOKEN=	*token area*: A-type address, or register (2) - (12).

Parameters for the standard form of the IGGENC macro

Table 53. Required and optional keywords for the IGGENC macro

Keyword	Function		
	CONNECT	**ENCRYPT/DECRYPT**	**DISCONNECT**
BLKIDLIST=	-	Required	-
BUFCOUNT=	-	Required	-
BUFLENLIST=	-	Required	-
DCB=	-	Optional	-
ENCRYPTA=	Required	-	-
INBUFLIST=	-	Required	-
OPTIONS=	Required	Required	Required
OUTBUFLIST=	-	Optional	-
RETCODE=	Required	Required	Required
RSNCODE=	Required	Required	Required
TCBADDR=	Optional	-	-
TOKEN=	Required	Optional	Required

Functional (positional parameter)
Required on standard form. Must be CONNECT, ENCRYPT, DECRYPT or DISCONNECT. Mixed case is allowed.

BLKIDLIST= A-type address, or register (2) - (12).
List of addresses of block prefixes. Each address and each block prefix is eight bytes even if the program is running in 24-bit or 31-bit. See BUFCOUNT.

BUFCOUNT= A-type address, or register (2) - (12).
A halfword containing the number of entries in the input buffer list (INBUFLIST), buffer length list (BUFLENLIST), output buffer list (OUTBUFLIST) and prefix list (BLKIDLIST). The value must be 1 or greater.

BUFLENLIST= A-type address, or register (2) - (12).
List of words that contain the length of each block. See BUFCOUNT.

DCB= A-type address, or register (2) - (12).
DCB associated with the encrypted data set. Use DCB only with the ENCRYPT or DECRYPT function. The DCB or TOKEN keyword is required for the ENCRYPT and DECRYPT functions. This keyword is mutually exclusive with the TOKEN keyword.

While the DCB is open, you can pass the DCB to the ENCRYPT and DECRYPT functions under a task that is different from the task that opened the DCB.

ENCRYPTA= A-type address, or register (2) - (12).
96-byte ENCRYPTA cell from the catalog. This keyword is required for the CONNECT function. You can retrieve a copy of this area from the data set catalog by calling IGGCSI00, the catalog search interface. See *z/OS DFSMS Access Method Services Commands* and *z/OS DFSMS Using Data Sets* for more detailed information.

INBUFLIST= A-type address, or register (2) - (12).
List of addresses of the input areas. Identifies areas containing data to encrypt or decrypt. Each must be eight bytes and can point above the bar even for a 24-bit or 31-bit caller. See BUFCOUNT.

OPTIONS= A-type address, or register (2) - (12).
Specifies an area containing the eight-byte options block as mapped by the IGGENCOPTS DSECT in the IGGENCPL macro.

Table 54. An area containing the eight-byte options block as mapped by the IGGENCOPTS DSECT in the IGGENCPL macro

Offset	Type	Length	Name	Description
0	DSECT		IGGENCOpts	IGGENC options block
	Integer	1	IGGENCLength	Length of this block. Must be 8.
1	Integer	1	IGGENCFunction	Function code. Values for the function code:
			IGGENCConnect	1 – Connect
			IGGENCEncrypt	2 - Encrypt
			IGGENCDecrypt	3 - Decrypt
			IGGENCDisconnect	4 - Disconnect
2	Bits	1	IGGENCFlag1	Flags
	1...		IGGENCBypassAuth	Bypass security check. Valid only if connect and caller is APF auth, supervisor state or system key.
	.1..		IGGENCENCRYPTA	The fifth parameter is ENCRYPTA and not an open DCB (CONNECT only). If you code the ENCRYPTA or DCB keyword on the execute form and also code the OPTIONS keyword, the macro sets or resets this bit.
	..xx xxxx			Reserved, should be zero
3		5		Reserved, should be zero
8	EQU		IGGENCOptsLen	Length of options block

OUTBUFLIST= A-type address, or register (2) - (12).
Optional list of output areas. Identifies areas to be used as the target of an encrypt or decrypt. The default is to use the input areas. Each must be eight bytes and can point above the bar even for a 24-bit or 31-bit caller. See BUFCOUNT.

RETCODE= A-type address, or register (2) - (12).
Four-byte area for the return code. The IGGENC macro also returns the return code in register 15.

RSNCODE= A-type address, or register (2) - (12).
Eight-byte area for the reason code. The IGGENC macro also returns the reason code in register 0. It is eight bytes even if your program is not running in 64-bit mode.

TCBADDR= A-type address, or register (2) - (12).
Word containing the address of a TCB, task control block. Code this only for the CONNECT function. If the word contents are non-zero, it must point to a TCB in the address space and the calling program must be in supervisor state or a system key or the job step must have APF authorization.

If the address in the parameter list is zero or the word that it points to contains zero, the connect will be for the current task. Therefore a way to denote the current task is to code TCBADDR=0 or to omit it.

TOKEN= A-type address, or register (2) - (12).
Eight-byte area for the token. This is required for the CONNECT and DISCONNECT functions. For CONNECT, the area must contain binary zeroes. It does not have to remain in the same location. You can make a copy to pass to other calls to IGGENC. The token content will remain valid until you call the DISCONNECT function or the task terminates, whichever is first. For DISCONNECT, the token must contain what the CONNECT function stored in it. The DISCONNECT function will clear the passed token to zeroes.

The ENCRYPT and DECRYPT functions require that you code either TOKEN or DCB. They are mutually exclusive. You can call the ENCRYPT and DECRYPT functions under a task that is different from the task that called the CONNECT function but the token becomes invalid when the connecting task does the disconnect or that task terminates.

IGGENC parameter list

You can use the IGGENCPL macro to map the parameter list for the IGGENC macro. There is one form of parameter list for callers in 24-bit or 31-bit and a different form for 64-bit callers. If you wish the IGGENC macro (standard, list, execute or modify form) to expand the 64-bit version, you must precede the macro call by calling the SYSSTATE macro with AMODE64=YES. That macro affects IGGENC on a line-by-line basic without regard to branching.

Three of the parameters are lists of addresses of the input and output areas and the block prefixes. These addresses always are 64-bit (eight bytes each) even if the parameter list is the 31-bit form.

The parameter list consists solely of a series of addresses. All of the addresses are either four bytes or eight bytes. The first address points to an options block. A byte in the options block specifies the purpose of the call.

Table 55. IGGENC parameter list

Function of Call	Number of addresses in parameter list
CONNECT	6
ENCRYPT	9
DECRYPT	9
DISCONNECT	4

The IGGENCPL macro supports these keywords:

- DSECT={YES | NO}. Whether to generate a DSECT statement for each area. The default is YES.
- PLIST={YES | NO}. Whether to map both forms of the parameter list. The default is YES.
- OPTIONS={YES | NO}. Whether to map the options block. The default is YES.
- CODES={YES | NO}. Whether to define the return and reason code constants. The default is YES.

Table 56. Format of the parameter list for 24-bit and 31-bit callers

Offset	Type	Length	Name	Description
0	DSECT		IGGENCParmL	IGGENC parameter list

Table 56. Format of the parameter list for 24-bit and 31-bit callers (continued)

Offset	Type	Length	Name	Description
0	Address	4	DSEP1Opt	Address of options block
4	Address	4	DSEP2RC	Address of return code (four bytes)
8	Address	4	DSEP3ReaC	Address of reason code (eight bytes)
12	Address	4	DSEP4CT	Address of connect token (eight bytes)
16	Address	4	DSEP5Enc	CONNECT: Address of 96-byte ENCRYPTA area from the catalog entry
	Address	4	DSEP5Blk	ENCRYPT, DECRYPT: Address of list of 64-bit addresses of block prefixes. The number of addresses is pointed to by DSEP8Num.
20	Address	4	DSEP6TCBAddr	CONNECT: Address of a word containing the address of a TCB, task control block. If the word in the parameter list or the word that it points to contains zero, the token will be associated with the current task.
	Address	4	DSEP6InB	ENCRYPT, DECRYPT: Address of list of 64-bit addresses of data areas to encrypt or decrypt. The number of addresses is pointed to by DSEP8Num.
24	Address	4	DSEP7BL	ENCRYPT, DECRYPT: Address of list of data lengths. Each is four bytes. The number of addresses is pointed to by DSEP8Num.
28	Address	4	DSEP8Num	ENCRYPT, DECRYPT: Address of a halfword that contains the number of entries in the lists of input and output (if specified) addresses and the list of area lengths. This also denotes the number of block identifiers in DSEP5Blk.
32	Address	4	DSEP9OutB	ENCRYPT, DECRYPT: Address of list of 64-bit addresses of data areas to contain the result of encryption or decryption. The number of addresses is pointed to by DSEP8Num. If this address is zero, the result of the encryption or decryption will be in the input areas.
36	EQU		DSELength	Maximum length of 31-bit parameter list

Table 57. Format of the parameter list for 64-bit callers

Offset	Type	Length	Name	Description
0	DSECT		IGGENCParmL	IGGENC parameter list
0	Address	8	DSE64P1Opt	Address of options block
8	Address	8	DSE64P2RC	Address of return code (four bytes)

Table 57. Format of the parameter list for 64-bit callers (continued)

Offset	Type	Length	Name	Description
16	Address	8	DSE64P3ReaC	Address of reason code (eight bytes)
24	Address	8	DSE64P4CT	Address of connect token (eight bytes)
32	Address	8	DSE64P5Enc	CONNECT: Address of 96-byte ENCRYPTA area from the catalog entry
	Address	8	DSE64P5Blk	ENCRYPT, DECRYPT: Address of list of addresses of block prefixes. The number of addresses is pointed to by DSE64P8Num.
40	Address	8	DSE64P6TCBAddr	CONNECT: Address of a word containing the address of a TCB, task control block. If the doubleword in the parameter list or the word that it points to contains zero, the token will be associated with the current task.
	Address	8	DSE64P6InB	ENCRYPT, DECRYPT: Address of list of addresses of data areas to encrypt or decrypt. The number of addresses is pointed to by DSE64P8Num.
48	Address	8	DSE64P7BL	ENCRYPT, DECRYPT: Address of list of data lengths. Each is four bytes. The number of addresses is pointed to by DSE64P8Num.
56	Address	8	DSE64P8Num	ENCRYPT, DECRYPT: Address of a halfword that contains the number of entries in the lists of input and output (if specified) addresses and the list of area lengths. This also denotes the number of block identifiers in DSE64P5Blk.
64	Address	8	DSE64P9OutB	ENCRYPT, DECRYPT: Address of list of addresses of data areas to contain the result of encryption or decryption. The number of addresses is pointed to by DSE64P8Num. If this address is zero, the result of the encryption or decryption will be in the input areas.
72	EQU		DSELength64	Maximum length of 64-bit parameter list

Return and reason codes for IGGENC macro

The return code is four bytes. It is returned in register 15 and in the area provided by the second parameter.

The reason code is eight bytes. It is returned in register 0 and in the area provided by the third parameter.

*Table 58. *Return and reason codes for the IGGENC macro*

Return Code	Reason Code (Hexadecimal)	Meaning
0	0	The call was successful.
4	00000D5F 00001611	The CONNECT request was made using an archived key that can only be used for decryption operations due to XFACILIT SAF profile CSF.KDS.KEY.ARCHIVE.DATA.DECRYPT. Any attempt to issue ENCRYPT will fail.
8		Return code IGGENCRC8. Bad parameters passed in parameter list. The *x* digit identifies the function, where 1: CONNECT, 2: ENCRYPT, 3: DECRYPT and 4: DISCONNECT.
	00000000 0000011x	IGGENCReaAddr. A required address is 0.
	00000000 0000012x	IGGENCReaFunc. Invalid function code in options. The invalid byte is ORed into the low order byte of the reason code so the high half might not be X'2'.
	00000000 0000013x	IGGENCReaLength. Length byte in options is less than eight. The function code is in the *x* digit.
	00000000 00xx014x	IGGENCReaToken. Bad token. Diagnostic codes for the *xx* byte: IGGENCReaTokenDiag32 X'20' Token not zero for connect IGGENCReaTokenDiag33 X'21' Token is 0 but function not connect
	00000000 00000151	IGGENCReaDEBErr. Invalid DEB.
	00000000 0000016*x*	IGGENCReaLockErr. Local lock not held. Internal system error.
	00000000 *xx*000211	Bad fields passed in ENCRYPTA area IGGENCReaType. Bad encryption type in ENCRYPTA. *xx* is the type code.
	00000000 *xx*000221	IGGENCReaKLen. Bad key length code in ENCRYPTA. *xx* is the incorrect length.
	*xxxxxxxx xx*000231	IGGENCReaKLab. Bad key label in ENCRYPTA. The first five bytes are the first five bytes of the incorrect key label
	00000000 *xx*000241	IGGENCReaMode. Bad encryption mode in ENCRYPTA.
	00000000 *xx*000311	Error when getting or freeing virtual storage. IGGENCReaGet31. Unable to get storage below the bar. The *xx* is the STORAGE macro return code.
	00000000 *xx*000334	IGGENCReaFree31. Unable to free storage below the bar. The *xx* is the STORAGE macro return code.
	00000000 00000411	IGGENCReaNoEncCell. Encryption cell does not exist.
	00000000 00001622	An attempt was made to encrypt data with an archived key that can only be used for decryption operations due to XFACILIT SAF profile CSF.KDS.KEY.ARCHIVE.DATA.DECRYPT.

*Table 58. *Return and reason codes for the IGGENC macro (continued)*

Return Code	Reason Code (Hexadecimal)	Meaning
8 continued		
	00000000 0000051*x*	IGGENCReaSetLock. Error obtaining local lock
	xxyyyyyy 00000611	IGGENCReaICSFErr. Error encountered during CSNEKRR2. The return code is in the *xx* byte and its reason code is in the *yyyyyy* bytes.
	00000000 00000711	IGGENCReaCrypKL. Incorrect crypto key length.
	xxyyyyyy 0000081*xxxyyyyyy* 0000081*x*	IGGENCReaBCFErr. Error encountered during BCFXCRYPT. The return code is in the *xx* byte and its reason code is in the *yyyyyy* bytes.
	00000000 00000911	IGGENCReaVerErr. Incorrect verification bytes
	00000000 00000A11	IGGENCReaCatErr. Error occurred during Catalog call.
	xxyyyyyy 00000B11	IGGENCReaKeyAcc. Not authorized to read key label. The return code from RACROUTE REQUEST=FASTAUTH is in the *xx* byte and its reason code is in the *yyyyyy* bytes.
	00000000 00000C1*x*	IGGENCReaBuffCnt. Encryption Buffer Count < 1
	00000000 00*xx*0D1*x*	IGGENCReaBuffCnt. Encryption Buffer Count < 1
	00000000 00*xx*0E1*x*	IGGENCReaBuffAddr. Encryption Buffer Addr is zero. The number of the buffer is in the *xx* byte.
	00000000 00*xx*0F1*x*	IGGENCReaPfxAddr. Block ID Address is Zero. The number of the block is in the *xx* byte.
	00000000 00*xx*101*x*	IGGENCReaBuffInv. Encryption Buffer invalid. The number of the buffer is in the *xx* byte.
	00000000 0000111x	IGGENCReaNonEncDS. Not an encrypted data set.
	00000000 00001211	IGGENCReaEncNotOk. EXCP caller without a DCBE that has DSENCRYPT=OK.
	00000000 00001310	IGGENCReaEncSysErr. Encryption service error.
	00000000 00001411	IGGENCReaTCBAddr. Caller unauthorized to specify TCB.
	00000000 00001511	IGGENCReaNpDCBE. DCBEDSENCNP bit is on, but the non-prefixed bit in the ENCRYPTA is off.
	00000000 00001521	IGGENCReaNpCtlg. Non-prefixed bit in the ENCRYPTA is on, but DCBEDSENCNP bit is off.
	00000000 00001531	IGGENCReaNpInvDCB. OPEN issued for encrypted basic/large format data set that has non-prefixed blocks but the DCB is not EXCP.

Examples of IGGENC macro

Example 1 - Standard forms

```
         ISITMGD DCB=SecretDCB,Version=2
         USING ISM,R1
         LTR   R15,R15             Branch if unable to determine
         JZ    EncFail               encryption status
         TM    ISMOFLG2,ISMENCRP   Branch if the data set is
         JZ    NotEncErr             not encrypted
         SR    R15,R15             Prepare for IC
         IC    R15,ISMBPRFX        Get the length of the
         STH   R15,PrefixLen          encryption prefix
         DROP  R1
         AH    R15,DCBBLKSI-IHADCB+SecretDCB  Add max block length
         GETMAIN R,LV=(R15)        Get prefix and buffer
         STG   R1,PrefixAd         Save prefix address
         AH    R1,PrefixLen        Point to place to read the block
         STG   R1,InBufList        One entry in buffer list
         LH    R15,DCBBLKSI-IHADCB+SecretDCB  Get user data length
         ST    R15,BufferLen       Set length to decrypt
        (Build channel program to read a block prefix and block.)
         EXCP  MyIOB               Read a block and its prefix
         WAIT  ECB=ECB
         IGGENC Decrypt,RetCode=MyRC,RsnCode=MyReason,              *
               BLKIDLIST=PrefixAd,INBUFLIST=InBufList,              *
               BUFLENLIST=BufferLen,BUFCOUNT=BufCount,              *
               DCB=SecretDCB
         LTR   R15,R15
         JNZ   DecryptErr
        (More code.)
R1       EQU   1           Register 1
R15      EQU   15          Register 15
SecretDCB DCB  DSORG=PS,MACRF=R,DDName=SECRET,EXLST=MyList
OptBlock DC    0XL8
         DC    AL1(L'OptBlock)  Length of options block
         DC    XL7'0'      Rest of options block
MyRC     DC    F'0'        Return code from IGGENC
MyReason DC    XL8'0'      Reason code from IGGENC (eight bytes)
PrefixAd DC    AD(0)       List of prefix addresses
InBufList DC   AD(0)       List of block addresses
BufferLen DC   F'0'        Length of block to decrypt
PrefixLen DC   H'0'        Length of prefix
BufCount DC    H'1'        Number of blocks to decrypt
        (The IOB, channel program and ECB are here.)
```

Before issuing the IGGENC macro to connect to encryption, this program opens the BSAM DCB. It is valid to issue EXCP with a BSAM DCB if the data set is a basic format or large format data set or is a PDS. If the data set is an encrypted basic format or large format data set, you can issue the IGGENC macro for data read or written with EXCP. The READ and WRITE macros take care of encryption and decryption. IGGENC is needed only for data read or written with EXCP, EXCPVR or XDAP.

The DD name identifies a DD statement or dynamically allocated data set that is basic format or large format and has the encrypted attribute. In this case the DD name is SECRET. The data set name, the key label and the encryption key do not matter to this program. Data set allocation, the OPEN macro and the IGGENC macro take care of them.

Before this program calls the IGGENC macro, it has initialized the IOB and channel program.

Example 2 - A call from a High Level Language

This is an assembler program that can be called by a program written in any high level language that can define the option block and lists of addresses.

```
IGGENC   TITLE 'IGGENC – Subroutine callable from compiled language'
*   The calling program must build a parameter list consisting only
* of addresses and pass its address in register 1.  The IGGENC macro
* does not use a register save area so this module can use it.
         USING SaveR,R13    Addressability to caller's save area
         ST    R14,SAVGRS14 Save the address of the caller
         IGGENC MF=(E,(1)) Call CONNECT, ENCRYPT, DECRYPT, DISCONNECT
         L     R14,SAVGRS14 Restore address of the caller
         BR    R14          Return to caller with return & reason
R13      EQU   13           Register 13
R14      EQU   14           Register 14
         IGGSAVER ,         Standard register save area
         END
```

Figure 24. Assembler subroutine callable from a High Level Language

IGGENC macro - list form

Syntax of the list form

Each parameter that is shown in Table 53 on page 219 as required must be coded on the list, modify or execute form. The execute form takes precedence over the modify form. The modify form takes precedence over the list form.

The list form of the IGGENC macro is written as shown in Table 59 on page 227.

Table 59. List form of the IGGENC macro

Syntax	Description
Function (positional parameter)	CONNECT, ENCRYPT, DECRYPT or DISCONNECT
BLKIDLIST=	*list addr*: A-type address
BUFFERCOUNT=	*count addr*: A-type address
BUFLENLIST=	*list addr*: A-type address
DCB=	*dcb addr*: A-type address
ENCRYPTA=	*encrypta addr*: A-type address
INBUFLIST=	*list addr*: A-type address
MF=	L
OPTIONS=	*area addr*: A-type address
OUTBUFLIST=	*list addr*: A-type address
RETCODE=	*return code addr*: A-type address
RSNCODE=	*reason code addr*: A-type address
TCBADDR=	*tcb addr*: A-type address
TOKEN=	*token area*: A-type address

Parameters for the list form

The parameters are explained under the standard form of the IGGENC macro, with the following exceptions:

Function (positional parameter)
Required on list form only when no parameter for ENCRYPT or DECRYPT is coded. Must be CONNECT, ENCRYPT, DECRYPT or DISCONNECT. Mixed case is allowed.

MF=L
This identifies the invocation as being the list form of the macro. Normally you code a symbol at the beginning of the IGGENC macro.

OPTIONS= A-type address
In a reentrant program the options block can be in read-only storage if you code the OPTIONS keyword on the list form before copying it and you do not code OPTIONS on the execute form. If you do that, the macro does not modify the options block but you risk the possibility that future functions will require the execute form to modify the options block. Therefore it is advisable for the options block to be in storage that the macro can modify. An exception is if you code ENCRYPTA or DCB on the CONNECT call and also code the OPTIONS parameter on the execute call.

IGGENC macro - execute form

Syntax of the execute form

Each parameter that is shown in Table 53 on page 219 as required must be coded on the list, modify or execute form. The execute form takes precedence over the modify form. The modify form takes precedence over the list form.

The execute form of the IGGENC macro is written as shown in Table 60 on page 228.

Table 60. Execute form of the IGGENC macro

Syntax	Description
Function (positional parameter)	CONNECT, ENCRYPT, DECRYPT or DISCONNECT
BLKIDLIST=	*list addr*: RX-type address, or register (2) - (12).
BUFFERCOUNT=	Halfword count of number of entries in the input buffer list (INBUFLIST), buffer length list (BUFLENLIST), output buffer list (OUTBUFLIST) and prefix list (BLKIDLIST).
BUFLENLIST=	*list addr*: RX -type address, or register (2) - (12).
DCB=	*dcb addr*: RX -type address, or register (2) - (12).
ENCRYPTA=	*encrypta addr*: RX -type address, or register (2) - (12).
INBUFLIST=	*list addr*: RX -type address, or register (2) - (12).
MF=	(E,*xxxx*[,COMPLETE]). The *xxxx* can be RX-type or register (1) – (12).
OPTIONS=	*area addr*: RX -type address, or register (2) - (12).
OUTBUFLIST=	*list addr*: RX -type address, or register (2) - (12).
RETCODE=	*return code addr*: RX -type address, or register (2) - (12).
RSNCODE=	*reason code addr*: RX -type address, or register (2) - (12).
TCBADDR=	*tcb addr*: RX-type address, or register (2) - (12).
TOKEN=	*token area*: RX-type address, or register (2) - (12).

Parameters for the execute form

The parameters are explained under the standard form of the IGGENC macro, with the following exceptions:

Function (positional parameter)
Required on execute form only when no parameter for ENCRYPT or DECRYPT is coded. Must be CONNECT, ENCRYPT, DECRYPT or DISCONNECT. Mixed case is allowed.

MF=E
This identifies the execute form of the macro.

COMPLETE specifies that the system is to check for required parameters and supply defaults for omitted optional parameters. This sets all fields in the parameter list. If you code COMPLETE, there is no need for your program to set any field in the parameter list before calling the execute form. The macro will initialize the parameter list before storing into it. You must ensure that the area is long enough

IGGENC macro - modify form

Each parameter that is shown in Table 53 on page 219 as required must be coded on the list, modify or execute form. The execute form takes precedence over the modify form. The modify form takes precedence over the list form. The purpose of the modify form is to update a parameter list without calling the function. It updates the parameter list with minimal checking. Later your program will call the execute form with the consolidated parameter list.

Syntax of the modify form

The modify form of the IGGENC macro is written as shown in Table 61 on page 229.

Table 61. Modify form of the IGGENC macro

Syntax	Description
Function (positional parameter)	CONNECT, ENCRYPT, DECRYPT or DISCONNECT
BLKIDLIST=	*list addr*: RX-type address, or register (2) - (12).
BUFFERCOUNT=	Halfword count of number of entries in the input buffer list (INBUFLIST), buffer length list (BUFLENLIST), output buffer list (OUTBUFLIST) and prefix list (BLKIDLIST).
BUFLENLIST=	*list addr*: RX -type address, or register (2) - (12).
DCB=	*dcb addr*: RX -type address, or register (2) - (12).
ENCRYPTA=	*encrypta addr*: RX -type address, or register (2) - (12).
INBUFLIST=	*list addr*: RX -type address, or register (2) - (12).
MF=	(M,*xxxx*[,COMPLETE]). The *xxxx* can be RX-type or register (1) – (12).
OPTIONS=	*area addr*: RX -type address, or register (2) - (12).
OUTBUFLIST=	*list addr*: RX -type address, or register (2) - (12).
RETCODE=	*return code addr*: RX -type address, or register (2) - (12).
RSNCODE=	*reason code addr*: RX -type address, or register (2) - (12).
TCBADDR=	*tcb addr*: RX-type address, or register (2) - (12).
TOKEN=	*token area*: RX-type address, or register (2) - (12).

Parameters for the modify form

The parameters are explained under the standard form of the IGGENC macro, with the following exceptions:

Function (positional parameter)
Not required on the modify form. Mixed case is allowed.

MF=(M,*xxxx*), or (M,*xxxx*,COMPLETE)
This identifies the modify form of the macro. The *xxxx* is the name of the parameter list to modify without calling the function.

The *xxxx* is a symbol or an RX form of address.

If you code the ENCRYPTA or DCB keyword on the execute form and also code the OPTIONS keyword, the macro sets or resets the IGGENCENCRYPTA bit.

COMPLETE means that you are coding all required parameters for an invocation. The macro will initialize the parameter list before storing into it.

Chapter 5. Using XDAP to Read and Write to Direct Access Devices

The execute direct access program (XDAP) is a macro instruction used to read, verify, or update a block on a direct access volume. This information covers the XDAP macro instruction, including information for compatibility with other IBM operating systems, what the XDAP macro does, and how to use it. It also discusses the macro instructions used with XDAP and the control blocks that are generated. However, IBM suggests using an access method such as VSAM in place of XDAP.

Because most of the specifications for XDAP are similar to those for the execute channel program (EXCP) macro instruction, you should be familiar with the information in "IDAW Requirements for EXCP Requests" on page 161. You should also be familiar with the information in *z/OS DFSMS Using Data Sets* that provides how-to information for using the access method routines of the system control program.

If you are not using the standard IBM data access methods, by issuing XDAP you can generate the control information and channel program necessary for reading or updating the records of a data set.

Restriction: XDAP does not support partitioned data set extended (PDSEs) , extended-format data sets, UNIX files, UNIX system services data sets, or SYSIN and SYSOUT data sets.

While XDAP cannot be used to add blocks to a data set, it can be used to change the keys of existing blocks. Any block configuration and any data set organization can be read or updated.

XDAP requires less storage than do the standard access methods. However, XDAP does not provide many of the control program services that are included in the access methods. For example, when XDAP is issued, the system does not block or unblock records and does not verify block length.

All virtual addresses that you use with XDAP except the OPEN and CLOSE parameter lists, DCBE, and IOBE must be 24–bit.

Using XDAP

To issue XDAP, provide the actual track address of the track containing the block to be processed. Also provide either the block identification or the key of the block, and specify which you should use to locate the block. If a block is located by identification, both the key and data portions of the block can be read or updated. If a block is located by key, only the data portion can be processed.

For additional control over I/O operations, you can write appendages that must be entered into the LPA library. Descriptions of these routines and their coding specifications are included under "IDAW Requirements for EXCP Requests" on page 161.

When using the XDAP macro instruction, code a DCB macro instruction in your program to generate a data control block (DCB) for the data set to be read or updated. Also code an OPEN macro instruction that initializes the data control block and produces a data extent block (DEB). The OPEN macro instruction must be executed before any XDAP macro instructions are executed.

When the XDAP macro instruction is assembled, a control block and the following executable code are generated. This control block can be logically divided into three sections:

- An event control block (ECB) that is supplied with a completion code each time the direct access channel program is terminated.
- An input/output block (IOB) that contains information about the direct access channel program.
- A direct access channel program consisting of three or four channel command words (CCWs). The type of channel program generated depends on the parameters of the XDAP macro instruction. When executed, it locates a block by either its actual address or its key and reads, updates, or verifies the block.

When the channel program has terminated, a completion code is placed into the event control block. After issuing XDAP and the direct access program has terminated, regain control by using a WAIT macro instruction to specify the address of the event control block. If volume switching is necessary, issue an EOV macro instruction. Once processing of the data set is completed, issue a CLOSE macro instruction to restore the data control block.

The parameters of the XDAP macro instruction are described in "Executing Direct Access Programs" on page 233.

Macro Instructions Used with XDAP

When using the XDAP macro instruction, you must also code DCB, OPEN, CLOSE and, in some cases, the EOV macro instructions. Special requirements or options for the other required macro instructions are explained in the following:

- "Defining a Data Control Block (DCB)" on page 232
- "Initializing a Data Control Block (OPEN)" on page 232
- "End of Volume (EOV)" on page 232
- "Restoring a Data Control Block (CLOSE)" on page 233

See "Data Control Block (DCB) Fields" on page 177 for listings of the parameters for DCB, OPEN, CLOSE and EOV.

Defining a Data Control Block (DCB)

Issue a DCB macro instruction for each data set to be read, updated, or verified by the direct access channel program. To learn which macro instruction parameters to code, see "Data Control Block (DCB) Fields" on page 177.

Initializing a Data Control Block (OPEN)

The OPEN macro instruction initializes one or more data control blocks so that their associated data sets can be processed. Issue OPEN for all data control blocks used by the direct access program. Some of the procedures performed when OPEN is executed include the following actions:

- Constructing a data extent block
- Transferring information from DD statements and data set labels to the DCB
- Verifying or creating standard labels
- Loading programmer-written appendage routines.

The two parameters of the OPEN macro instruction are the address or addresses of the data control blocks to be initialized and the method of I/O processing of the data set. The processing method can be specified as INPUT, OUTPUT, UPDAT, or EXTEND; however, if nothing is specified, INPUT is assumed. The parameters and different forms of the OPEN macro instruction are described in *z/OS DFSMS Macro Instructions for Data Sets*.

End of Volume (EOV)

The EOV macro instruction identifies end-of-volume (EOV) and end-of-data-set conditions. For an end-of-volume condition, EOV causes switching of volumes and verification or creation of standard labels. For an end-of-data-set condition (except when another data set is concatenated), EOV causes your end-of-data-set routine to be entered. When using XDAP, issue EOV if switching of direct access volumes is necessary or if secondary allocation is to be performed for a direct access data set opened for output. For details about the parameters of the EOV macro instruction, see "Handling End of Volume and End-Of-Data-Set Conditions" on page 175.

Note: Should EOV call DADSM EXTEND to extend on the same volume or a new volume (which implies DASD output), EXTEND will issue an enqueue on SYSVTOC. If the EOV request is issued for a data set on a volume where the application already holds the SYSVTOC enqueue, this will cause abnormal termination.

For this case, a circumvention would be to allocate the output data set large enough to not require a secondary extent or request the output data set be on a volume that is different than the one for which it holds the SYSVTOC enqueue.

Restoring a Data Control Block (CLOSE)

The CLOSE macro instruction restores one or more data control blocks so that processing of their associated data sets can be terminated. Issue CLOSE for all data sets used by the direct access channel program. When CLOSE is executed, some of the following procedures are performed:

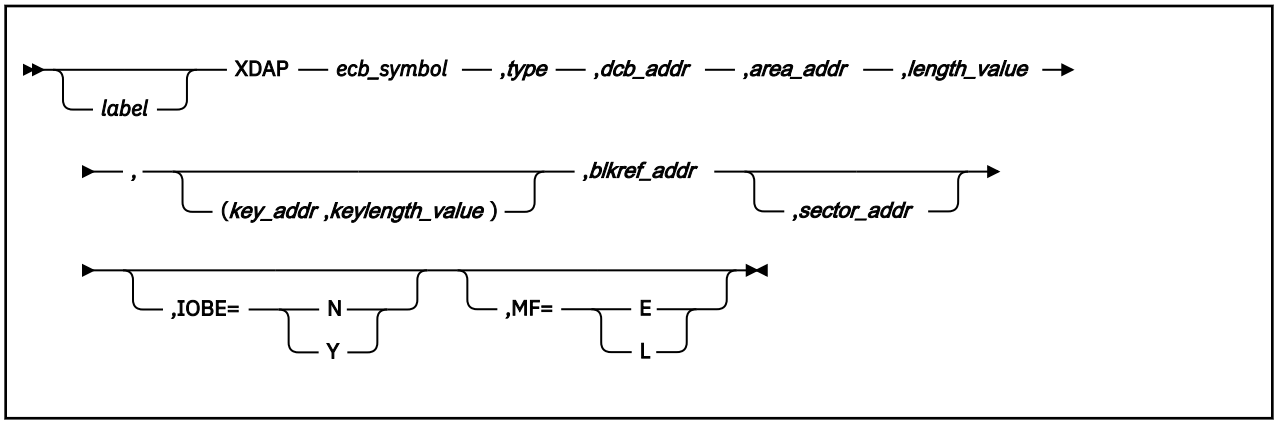
- Releasing the data extent block
- Removing information that was transferred to data-control block fields when OPEN was executed
- Verifying or creating standard labels
- Releasing programmer-written appendage routines.

The CLOSE macro instruction must identify the address of at least one data control block to be restored, and can specify other parameters. The parameters and different forms of the CLOSE macro instruction are described in *z/OS DFSMS Macro Instructions for Data Sets*.

Executing Direct Access Programs

The XDAP macro instruction produces the XDAP control block (that is, the ECB, IOB, and channel program) and executes the direct access channel program.

The format of the XDAP macro instruction is:



ecb_symbol—symbol or (2-12)
The symbolic name to be assigned to the XDAP event control block. Registers can be used only with MF=E.

***type*— RI or RK or WI or WK or VI or VK**
The type of I/O operation intended for the data set and the method by which blocks of the data set are to be located. One of the combinations shown must be coded in this field. The codes and their meanings are:

R
Read a block.

W
Update a block.

V
Verify that the device is able to read the contents of a block, but do not transfer data.

I
Locate a block by identification. (The key portion, if present, and the data portion of the block are read, updated, or verified.)

K
Locate a block by key. (Only the data portion of the block is read, updated, or verified.) If you code this value, code the *key_addr,keylength-value* operands.

***dcb_addr*—A-type address or (2-12)**
The address of the data control block for the data set.

***area_addr*—A-type address or (2-12)**
The address of an input or output area for a block of the data set.

***length_value*—absexp or (2-12)**
The number of bytes to be transferred to or from the input or output area. If blocks are to be located by identification and the data set contains keys, the value must include the length of the key. The maximum number of bytes transferred is 32 767.

***key_addr*—RX-type address or (2-12)**
When blocks are to be located by key, the address of a virtual storage field that contains the key of the block to be read, updated, or verified.

***keylength_value*—absexp or (2-12)**
When blocks are to be located by key, the length of the key. The maximum length is 255 bytes.

***blkref_addr*—RX-type address or (2-12)**
The address of a field in virtual storage that contains the actual track address of the track containing the block to be located. The actual address of a block is in the form MBBCCHHR, where:

M
Indicates which extent entry in the data extent block is associated with the direct access program

BB
Must be zero

CC
The cylinder address

HH
The actual track address

R
The block identification. R is not used if blocks are to be located by key.

The track address of the block reference (CCHH) may contain 28-bit cylinder numbers for devices with more than 65,520 cylinders. Showing nibbles it is in the form of CCCCcccH, where ccc represent bits 0-11 of the 28-bit cylinder number and CCCC represents bits 12-27 the 28-bit cylinder number. Use the TRKADDR macro to manipulate 16-bit and 28-bit cylinder numbers correctly.

(For more detailed information, see "Converting a Relative Track Address to an Actual Track Address" on page 207.)

***sector_addr*—RX-type address or (2-12)**
The address of a 1-byte field containing a sector value. The *sector_addr* parameter is used for rotational position sensing (RPS) devices. The parameter is optional, but its use will improve channel performance. When the parameter is coded, a set-sector CCW (using the sector value indicated by the data address field) precedes the search-ID-equal command in the channel program. The *sector_addr* parameter is ignored if the *type* parameter is coded as RK, WK, or VK. If a sector address is specified in the execute form of the macro, then a sector address, not necessarily the same, must be specified in the list form. The sector address in the executable form will be used.

Exception: No validity check is made on either the address or the sector value when the XDAP macro is issued. However, a unit check/command reject interruption will occur during channel-program execution if the sector value is not valid for the device or if the *sector_addr* operand is used when accessing a device without RPS.

IOBE=N
Indicates that an I/O Block Extension control block (IOBE) is not provided to EXCP.

IOBE=Y
Indicates that an IOBE is provided to EXCP; the address of the IOBE is in register 0. Use this to identify the application that is updating the VTOC, for the purposes of audit logging of VTOC I/O. The name of the application is in field IOBEUSER.

MF=
You can use the L-form of the XDAP macro instruction for a macro expansion consisting of a parameter list, or the E-form for a macro expansion consisting of executable instructions.

MF=E
The first operand (*ecb_symbol*) is required and can be coded as a symbol or supplied in registers 2 through 12. The *type*, *dcb_addr*, *area_addr,* and *length_value* operands can be supplied in either the L- or E-form. The *blkref_addr* operand can be supplied in the E-form or moved into the IOBSEEK field of the IOB by your program. The *sector_addr* is optional; it can be coded either in both the L- and E-form or in neither.

MF=L
The first two operands (*ecb_symbol* and *type*) are required and must be coded as symbols. If you choose to code *length_value* or *keylength_value*, they must be absolute expressions. Other operands, if coded, must be A-type addresses. (*blkref_addr* is ignored if coded.)

The XDAP macro builds a channel program containing A-type addresses. These addresses refer to storage within the L-form of the macro. If you copy the L-form of the macro to a work area so that the program can be reentrant, the E-form of the XDAP macro does not update the A-type addresses. This results in an invalid channel program.

The *dcb_addr*, *area_addr*, *blkref_addr*, and *sector_value* operands can be coded as RX-type addresses or supplied in registers 2 through 12. The *length_value* and *keylength_value* operands can be specified as absolute expressions or decimal integers or supplied in registers 2 through 12.

Control Blocks Used with XDAP

The control blocks generated during execution of the XDAP macro instruction consist of the following:

Input/Output Block

The input/output block is 40 bytes in length and immediately follows the event control block. “Control Block Fields” on page 177 contains a diagram of the IOB (Figure 18 on page 190). You might want to examine the IOBSENS0, IOBSENS1, and IOBCSW fields if the ECB is posted with X'41'.

Event Control Block

Refer to “Event Control Block (ECB) Fields” on page 197 for information on event control block fields.

Direct Access Channel Program

The direct access channel program is 24 bytes in length (except when set sector is used for RPS devices) and immediately follows the input/output block. Depending on the type of I/O operation that is specified in the XDAP macro instruction, one of four channel programs can be generated. The three channel command words for each of the four possible channel programs are shown in Table 62 on page 235.

Table 62. Channel Program Command Words Used by XDAP

Type of I/O Operation	CCW	Command Code
Read by identification	1	Search ID Equal
	2	Transfer in Channel
Verify by identification	3	Read Key and Data
Read by key	1	Search Key Equal
	2	Transfer in Channel

Table 62. Channel Program Command Words Used by XDAP (continued)

Type of I/O Operation	CCW	Command Code
Verify by key	3	Read Data
Write by identification	1	Search ID Equal
	2	Transfer in Channel
	3	Write Key and Data
Write by key	1	Search Key Equal
	2	Transfer in Channel
	3	Write Data

Note: For verifying operations, the third CCW is flagged to suppress the transfer of information to virtual storage.

When a sector address is specified with an RI, VI, or WI operation, the channel program is 32 bytes in length. Each of the channel programs in Table 62 on page 235 would be preceded by a set sector command.

RPS Device Sector Numbers

To obtain the performance improvement given by rotational position sensing, specify the *sector_addr* parameter in the XDAP macro. For programs that can be used with both RPS and non-RPS devices, test the UCBRPS bit to determine whether the device has rotational position sensing. If the UCBRPS bit is off, do not issue a channel program with a set sector command to the device.

The *sector_addr* parameter on the XDAP macro specifies the address of a 1-byte field in virtual storage. Store the sector number of the block to be located in this field. You can obtain the sector number of the block by using a resident system conversion routine. See “Obtaining the Sector Number of a Block on an RPS Device” on page 211

Chapter 6. Using Password Protected Data Sets

This information covers password protection for data sets. The use of password protection is not recommended, but is provided for compatibility with other IBM operating systems. You should use RACF protection (using SAF) instead.

The password protection described **does not** apply to data sets and catalogs managed by the Storage Management Subsystem (SMS) or to VSAM data sets. SMS ignores passwords. In addition, the PROTECT macro and SVC does not support a volume on a unit defined as dynamic.

If a SAF (system authorization facility)-compliant security product is active and provides protection for the data set, then the system bypasses password protection for that data set. Additionally, the system always bypasses password protection for VSAM and for SMS-managed data sets. The system provides SMS-managed data set and catalog protection through the SAF interface. For more SAF information, see "System Authorization Facility" in *z/OS MVS Programming: Assembler Services Guide*, and *z/OS MVS Programming: Assembler Services Reference ABE-HSP*.

For information about VSAM data set protection, see *z/OS DFSMS Using Data Sets* and *z/OS DFSMS Access Method Services Commands*.

The following are some reasons to use SAF instead of password protection:

- If you give a password to someone, you have no control over to whom they choose to give it.
- Data sets tend to have various passwords, making you write them down. This is less secure than if you can memorize one SAF password.
- Batch job access or interactive non-TSO access requires that a system operator supply a password. Your communication to the operator is likely to be insecure. That operator might not be present when your job runs. The operator might have to give each data set's password to other operators.
- The program is halted while each password is supplied. This is contrary to the increased automation of modern systems.
- There is no way to know who has used a particular password.
- It is human nature not to change passwords, especially if there are many. As time passes, there is a greater danger of them being exposed.
- If more than a small number of data sets have passwords, then the time for the system to find the PASSWORD data set entry increases greatly. With RACF, the increase is much less. With a RACF generic profile there is no increase in search time when a new data set uses the same profile.
- With DASD shared between systems, the password definitions on each system are independent. They can get out of synchronization.
- The PASSWORD data set entry contains the data set name but not the volume serial number. If you create a data set before defining a password, you could find that someone has already defined a password for that data set name. Your data set will require the existing password just to scratch or rename it.
- Password protection is not supported on system-managed volumes or on dynamic devices.

To use the data set protection feature of the operating system, create and maintain a PASSWORD data set consisting of records that associate the names of the protected data sets with the passwords assigned to each data set. The ways to maintain the PASSWORD data set consist of:

- Writing your own routines
- Using the PROTECT macro instruction
- Using the utility control statements of the IEHPROGM utility program
- If you have TSO, using the TSO PROTECT command.

This information discusses only the first two methods. The last two methods are discussed in the publications shown in the following list.

Before using this information, you should be familiar with the contents of the following publications:

- *z/OS DFSMS Using Data Sets* describes the data set protection feature.
- The following publications describe the operator messages and replies associated with the data set protection feature:
  - *z/OS MVS System Messages, Vol 1 (ABA-AOM)*
  - *z/OS MVS System Messages, Vol 2 (ARC-ASA)*
  - *z/OS MVS System Messages, Vol 3 (ASB-BPX)*
  - *z/OS MVS System Messages, Vol 4 (CBD-DMO)*
  - *z/OS MVS System Messages, Vol 5 (EDG-GLZ)*
  - *z/OS MVS System Messages, Vol 6 (GOS-IEA)*
  - *z/OS MVS System Messages, Vol 7 (IEB-IEE)*
  - *z/OS MVS System Messages, Vol 8 (IEF-IGD)*
  - *z/OS MVS System Messages, Vol 9 (IGF-IWM)*
  - *z/OS MVS System Messages, Vol 10 (IXC-IZP)*
- *z/OS MVS JCL Reference* describes the data definition (DD) statement parameter used to indicate that a data set is to be password protected. It is a subparameter of the LABEL parameter.
- *z/OS DFSMSdfp Utilities* describes how to maintain the PASSWORD data set using the utility control statements of the IEHPROGM utility program.
- *z/OS TSO/E Command Reference* describes how to use the TSO PROTECT command.

Providing Data Set Security

In addition to the usual label protection that prevents the opening of a data set without the correct data set name, the operating system provides data set security options that prevent unauthorized access to confidential data. Password protection prevents access to data sets until a correct password is entered by the system operator, or, for TSO, by a remote terminal operator.

The following types of access are allowed to password-protected data sets:

- PWREAD/PWWRITE—A password is required for read or write access.
- PWREAD/NOWRITE—A password is required for read access. Writing is not allowed.
- NOPWREAD/PWWRITE—Reading is allowed without a password. A password is required to write.

To prepare for use of the data set protection feature, place a sequential data set named PASSWORD on the system residence volume. This data set must contain at least one record for each data set placed under protection. Each record consists of a data set name, a password for that data set, a counter field, a protection-mode indicator, and a field for recording any information you wish to log. On the system residence volume, these records are formatted as a key area (data set name and password) and a data area (counter field, protection-mode indicator, and logging field). The data set is searched on the key area.

- The area allocated to the data set should not have been previously used for a PASSWORD data set, as this might cause unpredictable results when adding records to the data set.
- If the system residence volume does not contain a PASSWORD data set, the system allows no access to password protected data sets. Do not rely on this for protection because anyone who creates a data set named PASSWORD on the system residence volume can define a password for any data set.
- Data sets on magnetic tape are protected only when standard labels are used.

You can write routines to create and maintain the PASSWORD data set. For information on using the PROTECT macro instruction to maintain the PASSWORD data set, see “Maintaining the PASSWORD Data Set Using PROTECT” on page 241. Using the IEHPROGM utility program to maintain the PASSWORD data set is described in *z/OS DFSMSdfp Utilities*. These routines can be placed in your own library or in the system's library (SYS1.LINKLIB). You can use a data management access method to read from and write to the PASSWORD data set.

Password-protected data sets can only be accessed by programs supplying the correct password. Upon receiving a request to open a protected data set, the operating system checks whether the data set has already been opened for this job step and if the access mode is compatible with the previously used protection mode. If neither condition is satisfied, a message requesting the password is sent to the operator console. If the program attempting to open the data is running under TSO in the foreground, the message is sent to the TSO terminal operator.

PASSWORD Data Set Characteristics

The PASSWORD data set and your operating system must reside on the same volume. That volume is the IPL volume. It is called the system residence volume. The space allocated to the PASSWORD data set must be contiguous. The amount of space allocated depends on the number of data sets you want to protect. Each entry in the PASSWORD data set requires 132 bytes of space. The organization of the PASSWORD data set is physical-sequential; the record format is unblocked, fixed-length records (RECFM=F). Each record that forms the data area is 80 bytes long (LRECL=80,BLKSIZE=80) and is preceded by a 52-byte key (KEYLEN=52). The key area contains the fully-qualified data set name (up to 44 bytes long) and a password, one to eight alphanumeric characters in length, left justified with blanks added to fill the areas.

Restriction: The PASSWORD data set cannot be large format or extended format.

Attention: Data sets on magnetic tape complying with the specifications of the International Organization for Standardization (ISO) 1001-1979 or the American National Standards Institute (ANSI) X3.27-1978 do not include generation and version numbers as part of generation data set names. If included in the PASSWORD data set, generation and version numbers for these data sets are ignored.

You should protect the PASSWORD data set by creating a password record for it when your program initially builds the data set or you should use RACF to control access. Thereafter, the PASSWORD data set cannot be opened (except by the operating system routines that scan the data set) without entering the password. If a problem occurs on a password-protected system data set, maintenance personnel require the password to access the data set and resolve the problem.

Creating Protected Data Sets

The data definition (DD) statement parameter LABEL= can be used to indicate that a data set is to be password protected. For data sets on DASD, an alternative method for a previously allocated data set is to use the PROTECT macro instruction, the IEHPROGM utility, or the TSO PROTECT command. You can create a data set and set the protection indicator in its label without entering a password record for it in the PASSWORD data set. In this case the system allows no access to the data set.

Operating procedures at your installation must ensure that password records for all data sets currently password-protected are entered in the PASSWORD data set. For installations where independent computing systems share common DASD resources, PASSWORD data sets on all systems must contain the appropriate password records for any protected data set on shared DASD.

Under certain circumstances, the order in which data sets are allocated and deallocated from multiple systems on shared DASD could result in loss of protection or corruption of data. For example, if a set is allocated and opened by a user on system A and then scratched by a different user on system B, the first user has a window to the unallocated (free) area. If any data set, protected or unprotected, is allocated in that space by a user on either system while the window is open, the new data set has no protection from the user with the window. The most common solution to this problem is to use GRS (see *z/OS MVS Planning: Global Resource Serialization*).

While the allocation disposition is still NEW, a password-protected data set can be used without supplying a password. Once the data set is deallocated, a subsequent attempt to open it results in termination of the program unless the password record is available and the correct password is supplied. If the protection mode is NOPWREAD and the request is to open the data set for input or read backward, no password is required.

Tape Volumes Containing Multiple Password-protected Data Sets

To password protect a data set on a tape volume containing other data sets, password protect all the data sets on the volume. (Standard labels—SL, SUL, AL, or AUL—are required. For definitions of these label types and the protection-mode indicators that can be used, see *z/OS DFSMS Using Magnetic Tapes*.)

If you issue an OPEN macro instruction to create a data set following an existing password-protected data set, the password of the existing data set will be verified during open processing for the new data set. The password supplied must be associated with a PWWRITE protection-mode indicator.

Protection Feature Operating Characteristics

The topics that follow discuss the protection feature operating characteristics:

- "Terminating the Protection Feature Process" on page 240
- "Password Protection When Switching Volumes" on page 240
- "Password Protection When Concatenating Data Sets" on page 240
- "Maintaining the Counter for Password Protection" on page 241

Terminating the Protection Feature Process

Processing is terminated if:

- The operator cannot supply the correct password for the protected data set within two attempts.
- A password record does not exist in the PASSWORD data set for the protected data set being opened.
- The protection-mode indicator in the password record and the method of I/O processing specified in the open routine do not agree; for example, OUTPUT specified against a read-only protection-mode indicator.
- There is a mismatch in data set names for a data set involved in a volume switching operation. This is discussed in the next paragraph.

Password Protection When Switching Volumes

Password protection is retained when volumes of a multivolume data set are switched. If the following conditions are met, the system accepts a newly-mounted tape volume to be used for input or a newly-mounted direct access volume:

- The data set names are the same in the password record for the data set and the job file control block (JFCB). (This ensures that the problem program has not changed the data set name in the JFCB since the data set was opened.)
- The protection-mode indicator in the password record is compatible with the processing mode, and a valid password has been supplied.

The system accepts a newly-mounted tape volume for output under any of the following conditions:

- The security indicator in the HDR1 label indicates password protection; the data set name in the password record is the same as the data set name in the JFCB; and the protection-mode indicator is compatible with the processing mode. (If the data set name in the JFCB has been changed, a new password is requested from the operator.)
- The security indicator in the HDR1 label does not indicate password protection. (A new label is written with the security indicator indicating password protection.)
- Only a volume label exists. (An HDR1 label is written with the security indicator indicating password protection.)

Password Protection When Concatenating Data Sets

A password is requested for every protected data set that is involved in a concatenation of data sets.

Password Protection SCRATCH and RENAME Functions

To delete or rename a protected data set, the job step making the request must be able to supply the password. The system checks to see if the job step is currently authorized to write to the data set. If the job step is not, a password must be provided that is associated with a WRITE protection-mode indicator.

Maintaining the Counter for Password Protection

The operating system increments the counter in the password record on each usage, but no overflow indication will be given (overflow after 65535 openings). Provide a counter maintenance routine to check and, if necessary, reset this counter.

Maintaining the PASSWORD Data Set Using PROTECT

To use the PROTECT macro instruction, your PASSWORD data set must be on the system residence volume. The PROTECT macro can be used to:

- Add an entry to the PASSWORD data set
- Replace an entry in the PASSWORD data set
- Delete an entry from the PASSWORD data set
- Obtain a list of information about an entry in the PASSWORD data set; this list contains the security counter, access type, and the 77 bytes of security information in the data area of the entry.

The PROTECT macro also updates the DSCB of a protected direct access data set to reflect its protection status; this feature eliminates the need for job control language when you protect a data set.

PROTECT does not support data sets on dynamic devices.

Record Format

When using the PROTECT macro, the PASSWORD data set must contain at least one record for each protected data set. The password (the last eight bytes of the key area), that you assign when you protect the data set for the first time, is called the control password.

You can create as many secondary records for the same protected data set as needed. Passwords assigned to these additional records are called secondary passwords. This feature allows several users to access the same protected data set while you control the way it is used. For example, one user could be given read/write authorization while another user is assigned a password that allows only read access.

Protection-Mode Indicator

You can set the protection-mode indicator (third data byte) in the password record to one of the following values:

- X'00' to indicate that the password is a secondary password and the protected data set is to be read only (PWREAD).
- X'80' to indicate that the password is the control password and the protected data set is to be read only (PWREAD).
- X'01' to indicate that the password is a secondary password and the protected data set is to be read and written (PWREAD/PWWRITE).
- X'81' to indicate that the password is the control password and the protected data set is to be read and written (PWREAD/PWWRITE).

The checking sequence for the protection status of a data set produces the following defaults:

If control password is:	Secondary password must be:
1. PWREAD/PWWRITE or PWREAD/NOWRITE	PWREAD/PWWRITE or PWREAD/NOWRITE

2. NOPWREAD/PWWRITE NOPWREAD/PWWRITE

If the control password is set to either of the settings in item 1 and you try to set the secondary password to NOPWREAD/PWWRITE, the secondary password reverts to PWREAD/PWWRITE.

If the control password is changed from either of the settings in item 1 to the setting in item 2, the secondary password is reset to NOPWREAD/PWWRITE.

If the control password is changed from the setting in item 2 to a setting in item 1, the secondary password is set to PWREAD/PWWRITE.

Because the DSCB of the protected data set is updated only when the control password is changed, protection attributes for secondary passwords might conflict with the protection attributes of the control password.

PROTECT Macro Specification

The format of the PROTECT macro is:

***parameter list address*—A-type address, (2-12), or (1)**
Indicates the location of the parameter list. The parameter list must be created before the PROTECT macro is issued. The address of the parameter list can be passed in register 1, in any of registers 2 through 12, or as an A-type address. The first byte of the parameter list must be used to identify the function (add, replace, delete, or list). See Figure 25 on page 242 through Figure 28 on page 246 for the parameter lists and codes used to identify the functions.

Requirement: The parameter lists and the areas addressed by the list must reside below 16 MB virtual. PROTECT will fail the request if the actual UCB of the system residence volume is above 16 MB or if PROTECT tries to update the data set's DSCB and one of its UCBs is above the 16 MB line.

PROTECT Macro Parameter Lists

ADD Function

0	X'01'	13	Control password pointer
1	00 00 00	16	Number of volumes
4	Data set name length	17	Volume list pointer
5	Data set name pointer	20	Protection code
8	00	21	New password pointer
9	00 00 00	24	String length
10	00	25	String pointer

Figure 25. Parameter List for ADD Function

Table 63.

Offsets	Length or bit pattern	Description
0	X'01'	Entry code indicating the ADD function.

Table 63. (continued)

Offsets	Length or bit pattern	Description
1	00 00 00	
4	Data set name length	
5	Data set name pointer.	
8	00	
9	00 00 00	
10	00	
13	Control password pointer	The control password is assigned when the data set is placed under protection for the first time. The pointer can be 3 bytes of binary zeros if the new password is the control password.
16	Number of volumes	If the data set is not cataloged, to have it flagged as protected, specify the number of volumes in this field. A zero requests that the catalog information be used.
17	Volume list pointer	If the data set is not cataloged, to have it flagged as protected, provide the address of a list of volume serial numbers in this field. Zeros request that the catalog information be used.
20	Protection code	A 1-byte number indicating the type of protection: X'00' indicates default protection (for the ADD function; the default protection is the type of protection specified in the control password record of the data set); X'01' indicates that the data set is to be read and written; X'02' indicates that the data set is read only; and X'03' indicates that the data set can be read without a password, but a password is needed for write access. The PROTECT macro uses the protection code value, specified in the parameter list, to set the protection-mode indicator in the password record.
21	New password pointer	If the data set is being placed under protection for the first time, the new password becomes the control password. If you are adding a secondary entry, the new password is different from the control password.
24	string	The length of the character string (maximum 77 bytes) to be placed in the optional information field of the password record. If you do not want to add information, set this field to zero.
25	String pointer	The address of the character string placed in the optional information field. If you do not want to add information, set this field to zero.

0 X'01'
Entry code indicating ADD function.

4 Data set name length.

5 Data set name pointer.

13 Control password pointer.
The control password is assigned when the data set is placed under protection for the first time. The pointer can be 3 bytes of binary zeros if the new password is the control password.

16 Number of volumes.
If the data set is not cataloged, to have it flagged as protected, specify the number of volumes in this field. A zero requests that the catalog information be used.

17 Volume list pointer.
If the data set is not cataloged, to have it flagged as protected, provide the address of a list of volume serial numbers in this field. Zeros request that the catalog information be used.

20 Protection code.
A 1-byte number indicating the type of protection: X'00' indicates default protection (for the ADD function; the default protection is the type of protection specified in the control password record of the data set); X'01' indicates that the data set is to be read and written; X'02' indicates that the data set is read only; and X'03' indicates that the data set can be read without a password, but a password is needed for write access. The PROTECT macro uses the protection code value, specified in the parameter list, to set the protection-mode indicator in the password record.

21 New password pointer.
If the data set is being placed under protection for the first time, the new password becomes the control password. If you are adding a secondary entry, the new password is different from the control password.

24 String length.
The length of the character string (maximum 77 bytes) to be placed in the optional information field of the password record. If you do not want to add information, set this field to zero.

25 String pointer.
The address of the character string placed in the optional information field. If you do not want to add information, set this field to zero.

REPLACE Function

0	X'02'	13	Control password pointer
1	00 00 00	16	Number of volumes
4	Data set name length	17	Volume list pointer
5	Data set name pointer	20	Protection code
8	00	21	New password pointer
9	Current password pointer	24	String length
12	00	25	String pointer

Figure 26. Parameter List for REPLACE Function

0 X'02'.
Entry code indicating REPLACE function.

4 Data set name length.

5 Data set name pointer.

9 Pointer to current password.
The address of the password that is going to be replaced.

13 Control password pointer.
The address of the password assigned to the data set when it was first placed under protection. The pointer can be set to 3 bytes of binary zeros if the current password is the control password.

16 Number of volumes.
If the data set is not cataloged, to have it flagged as protected, specify the number of volumes in this field. A zero requests that the catalog information be used.

17 Volume list pointer.
If the data set is not cataloged, to have it flagged as protected, provide the address of a list of volume serial numbers in this field. If this field is zero, the catalog information is used.

20 Protection code.
A 1-byte number indicating the type of protection: X'00' indicates default protection (for the REPLACE function the default is the type of protection specified in the control password record of the data set); X'01' indicates that the data set is to be read and written; X'02' indicates that the data set is to be read only; and X'03' indicates that the data set can be read without a password, but a password is needed for write access.

21 New password pointer.
The address of the password to be used to replace the current password.

24 String length.
The length of the character string (maximum 77 bytes) to be placed in the optional information field of the password record. If you do not want to add information, set this field to zero.

25 String pointer.
The address of the character string to be placed in the optional information field. If you do not want to add information, set this field to zero.

DELETE Function

0	X'03'	9	Current password pointer
1	00 00 00	12	00
4	Data set name length	13	Control password pointer
5	Data set name pointer	16	Number of volumes
8	00	17	Volume list pointer

Figure 27. Parameter list for DELETE function

0 X'03'.
Entry code indicating DELETE function.

4 Data set name length.

5 Data set name pointer.

9 Current password pointer.
The address of the password to be deleted. You can delete either a control or secondary entry.

13 Control password pointer.
The address of the password assigned to the data set when it was placed under protection for the first time. The pointer can be 2 bytes of binary zeros if the current password is also the control password.

16 Number of volumes.
If the data set is not cataloged, to have it flagged as protected, specify the number of volumes in this field. A zero requests that the catalog information be used.

17 Volume list pointer.
If the data set is not cataloged, to have it flagged as protected, provide the address of a list of volume serial numbers in this field. Zeros request that the catalog information will be used.

LIST Function

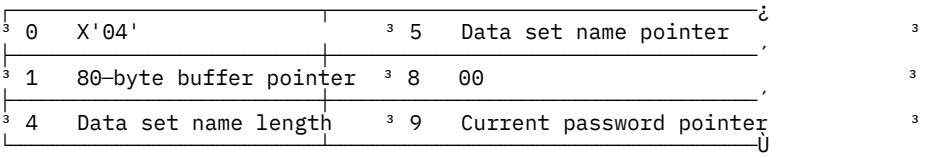


Figure 28. Parameter list for LIST function

0 X'04'.
Entry code indicating LIST function.

1 80-byte buffer pointer.
The address of a buffer to receive the information returned to your program by the macro instruction.

4 Data set name length.

5 Data set name pointer.

9 Current password pointer.
The address of the password of the record to be listed.

Return Codes from the PROTECT Macro

When the PROTECT macro finishes processing, register 15 contains one of the following return codes:

Table 64. PROTECT Return Codes

Return Code	Description
0 (X'00')	Updating of the PASSWORD data set completed successfully.
4 (X'04')	The password of the data set name was already in the PASSWORD data set.
8 (X'08')	The password of the data set name was not in the PASSWORD data set.
12 (X'0C')	A control password is required or the one supplied is incorrect.
16 (X'10')	The supplied parameter list was incomplete or incorrect.
20 (X'14')	There was an I/O error in the PASSWORD data set or the system residence volume (which contains the PASSWORD data set), has an actual UCB that resides above the 16 MB line.
24 (X'18') [1]	The PASSWORD data set was full.
28 (X'1C')	The validity check of the buffer address failed.
32 (X'20') [2]	The LOCATE macro failed. LOCATE's return code is in register 1, and the number of indexes searched is in register 0.
36 (X'24') [2]	The OBTAIN macro for the user data set failed. OBTAIN's return code is in register 1. The user data set resides on a volume that has an actual UCB that resides above the 16 MB line. Register 1 contains a 4, which simulates an OBTAIN return code.
40 (X'28') [2]	The DSCB could not be updated.
44 (X'2C')	The PASSWORD data set does not exist.
48 (X'30') [2]	Tape data set cannot be protected.

Table 64. PROTECT Return Codes (continued)

Return Code	Description
52 (X'32') [2]	Data set in use.
56 (X'38') [2]	The data set uses the virtual storage access method (VSAM).
60 (X'3C')	The data set is cataloged as being on more than 20 volumes. The DSCBs on the first 20 volumes have been updated. PROTECT does not support updating the rest.

Note:

1. A message is written to the console indicating that the PASSWORD data set is full.
2. The PASSWORD data set has been updated, but the DSCB has not been flagged to indicate the protected status of the data set.

Chapter 7. Using System Macro Instructions

This information covers system macro instructions. The macros are grouped functionally where appropriate and perform the stated functions.

- Ensure data security (DEBCHK macro)
- Obtain device characteristics from control blocks and system tables (DEVTYPE macro)
- Modify the JFCB (RDJFCB and OPEN TYPE=J macros)
- Manipulate I/O activity queues (PURGE and RESTORE macros)
- Perform track capacity calculations (TRKCALC macro).
- Perform calculations and conversions with 28-bit cylinder addresses (TRKADDR macro).

Some functions of these macros tend to depend on the internal logic of the system.

Before reading this information, you should be familiar with the publications *High Level Assembler/MVS & VM & VSE Language Reference* They contain the information necessary to code programs in the assembler language.

Ensuring Data Security by Validating the Data Extent Block (DEBCHK macro)

Protecting one user's data from inadvertent or malicious access by an unauthorized user depends on protection of the data extent block (DEB). The DEB is a critical control block because it contains information about the device a data set is mounted on, and describes the location of data sets on direct access device storage volumes.

To ensure that only a valid system-provided DEB (normally built by open) is passed to data management functions, the DEBCHK verify function is used. OPEN places the address of DEBs it creates to a DEB table, which is used by the verify function. If you code a routine that builds a DEB, add the address of the DEB you built to the DEB table. If you code a routine that depends on the validity of a DEB that is passed to your routine, verify that the DEB passed to your routine has a valid entry in the DEB table and points to your DCB or access method control block (ACB). Use the TYPE=ADD and the TYPE=VERIFY operands of the macro, respectively.

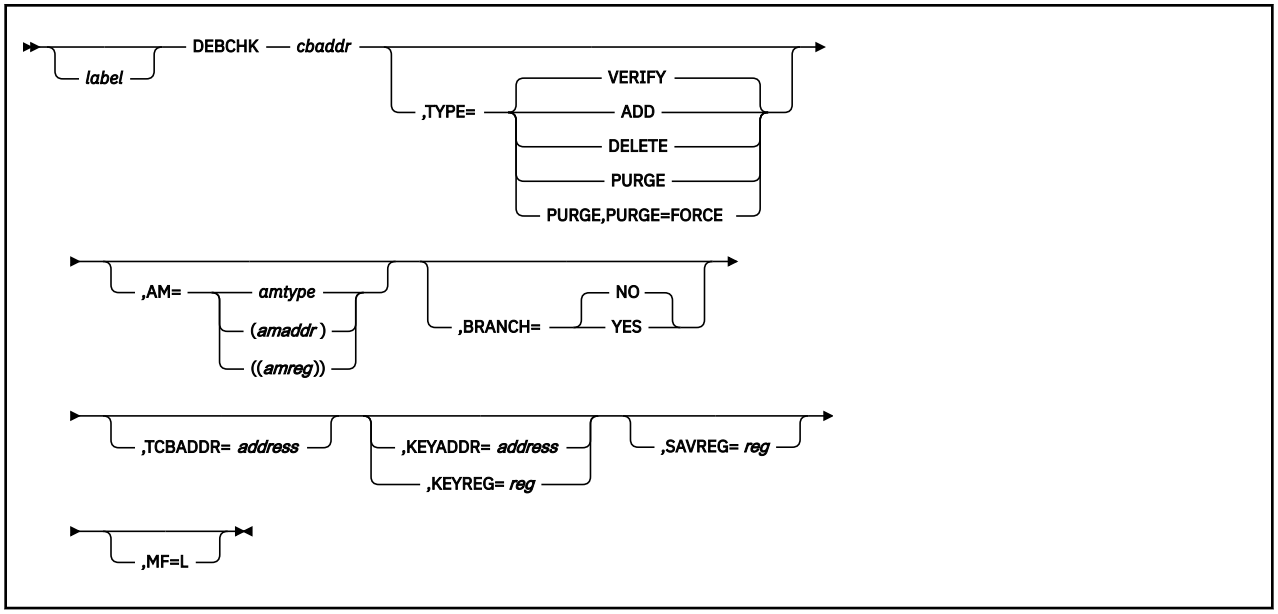
To prevent an asynchronous routine from changing or deleting, or assigning a new DEB to a DCB, hold the local lock. In this case, use the branch entry to the DEBCHK verify routine and use the DEB address returned in register 1, not the DEB address in the DCB. The DCB will remain valid as long as your program holds the local lock or prevents untrustworthy programs from running.

Your program must be executing in 24-bit or 31-bit addressing mode when you call the DEBCHK macro.

The DEB fields used for EXCP and EXCPVR are illustrated in Appendix A, “Control Blocks,” on page 431 (all the DEB fields are illustrated in *z/OS DFSMSdfp Diagnosis*).

DEBCHK Macro Specification

The format of the DEBCHK macro is:



cbaddr
Control block address.

for BRANCH=NO
RX-type address, (2-12), or (1)

A control block address that is passed to the DEBCHK routine. This operand is ignored if MF=L is coded. For verify, add, and delete requests, *cbaddr* is the address of a DCB or ACB that points to the DEB whose address is either verified to be in the DEB table, added to the DEB table, or deleted from the DEB table. For the purge function, *cbaddr* is the address of the DEB whose pointer is to be purged from the table: No reference is made to the DCB or ACB.

Recommendation: A spooled DCB's DEB does not point back to the DCB, but to the spooled ACB; in this case, the DEBCHK should be issued against the ACB.

for BRANCH=YES
The A-type address of a 4-byte field, or a register (3-9) or (12), that points to the DCB or ACB containing the DEB to be verified.

TYPE=VERIFY or ADD or DELETE or PURGE or PURGE,PURGE=FORCE
Indicates the function to be performed. If MF=L is coded, TYPE is ignored. The functions are:

VERIFY
This function is assumed if the TYPE operand is not coded. The control program checks the DEB table to determine whether the DEB pointer is in the table at the location indicated by the DEBTBLOF field of the DEB. The DEB is also checked to verify that DEBDCBAD points to the DCB (or ACB) passed to DEBCHK. The DEBAMTYP field in the DEB is compared to the AM operand value, if given. The two must be equal. TYPE=VERIFY can be issued in either supervisor or problem state.

ADD
The DEB and the DCB (or ACB) must point to each other before the DEB address can be added to the DEB table. Before the DEB pointer can be added to the table, the DEB itself must be queued on the current TCB DEB chain (the TCBDEB field contains the address of the first DEB in the chain). DEBCHK adds the DEB address to the DEB table at some offset into the table. DEBCHK places a value in the DEBTBLOF field of the DEB and inserts the access method type into the DEBAMTYP field of the DEB. DEBCHK places a zero in the DEBAMTYP field if the AM operand is not coded. TYPE=ADD can be issued only in supervisor state.

DELETE
The DEB and the DCB (or ACB) must point to each other before the DEB address can be deleted from the DEB table. TYPE=DELETE can be issued only in supervisor state.

PURGE
DEBCHK removes the DEB pointer from the DEB table without checking the DCB (or ACB). TYPE=PURGE can be issued only in supervisor state.

PURGE,PURGE=FORCE
DEBCHK removes the DEB pointer from the DEB table without checking the DCB (or ACB). The caller must be in system key, supervisor state, hold the local lock, and the passed DEB pointer must exist in the DEB table but not represent a valid DEB.

AM
Specifies an access method value. Each value corresponds to a particular access method type (note that BPAM and SAM have the same values):

Value	Type
(X'00')	NONE
(X'01')	VSAM
(X'02')	EXCP
(X'04')	Not supported
(X'08')	GAM
(X'10')	TAM
(X'20')	BPAM
(X'20')	SAM
(X'40')	BDAM
(X'81')	SUBSYS
(X'84')	Not supported

The operand can be coded in one of the following three ways, only the first of which is valid for the list form (MF=L) of the instruction.

amtype
Refers to the access method: BDAM, SAM, BPAM, TAM (which refers to BTAM only), GAM, EXCP, or VSAM. SUBSYS identifies a subsystem of the operating system, such as a job entry subsystem. NONE indicates that no access method or subsystem is specified.

(*amaddr*)
The RS-type address of the access method value. This format cannot be coded when MF=L is used.

((*amreg*))
One of the general registers 1 through 14 that contains the access method value in its low-order byte (bit positions 24 through 31). The high-order bytes are not inspected. This form cannot be used when MF=L is coded.

The use of *amaddr* and *amreg* should be restricted to those cases where the access method value has been generated previously by the MF=L form of DEBCHK. If MF=L is not coded, the significance of the AM operand depends upon the TYPE.

If TYPE is ADD and AM is specified, the access method value is inserted in the DEBAMTYP field of the DEB, and all subsequent DEBCHK macros referring to this DEB must either specify the same AM or omit the operand. When the AM operand is omitted for TYPE=ADD, a null value (0) is placed in the DEB and all subsequent DEBCHK macros must omit the AM operand.

If AM is specified when the TYPE is PURGE, DELETE, or VERIFY, the access method value is compared to the value in the DEBAMTYP field of the DEB. If AM is omitted, no comparison is made.

BRANCH=NO or YES
Specifies whether you want to use the branch entry to the DEBCHK verify routines.

NO
SAVREG
TCBADDR
KEYADDR
KEYREG
Specifies branch entry is not to be used. The program ignores operands SAVREG, TCBADDR, KEYADDR, and KEYREG. Your program must be running in TCB mode.

YES
Specifies the branch entry is to be used. TYPE=VERIFY must be implicitly or explicitly specified. The operands TCBADDR and KEYADDR/KEYREG are required. AM and MF are ignored. Notes for BRANCH=YES:

- Your program can run in TCB or SRB mode.
- Registers 1, 2, 10, 11, 14, and 15 must not be used for SAVREG=.
- Registers 1, 2, 10, 11, 14, 15, and the register that is specified for SAVREG= must not be used for *cbaddr*, TCBADDR=, or KEYADDR=/KEYREG=.
- The contents of registers 10, 11, and 14 are unpredictable on completion. Also, if you do not specify SAVREG=, the contents of register 2 are unpredictable.
- At completion, register 1 contains the address of the DEB, and register 15 contains either 0, 4, or 16 (see "Return Codes from DEBCHK" on page 252 for codes and their meanings).
- Can be specified when operating in 24-bit or 31-bit addressing mode.

TCBADDR=*address*—RX-type address, (3-9), or (12)
Specifies the word or register containing the address of the TCB to be used by the DEBCHK routine. The purpose is to locate the DEB table. This is useful only if you have reason to believe that there are multiple job step tasks. If you code a register, enclose it in parentheses. Use this operand only with BRANCH=YES. If you code TYPE=VERIFY or omit TYPE, the TCBADDR keyword is required.

KEYADDR=*address*—RX-type address
Specifies the location, or a register that points to the location, of a byte containing the key to be used when accessing the DCB (or ACB). The protection key is in bits 0 to 3. Use this operand only with BRANCH=YES.

KEYREG=*reg*
Specifies the register containing the key value in bit positions 24-27 to be used when accessing the DCB(or ACB). Use this operand only with BRANCH=YES.

SAVREG=*reg*
Specifies the register in which register 2 is to be saved. Use this operand only with BRANCH=YES.

MF=L
Indicates the list form of the DEBCHK macro instruction. When MF=L is coded, a parameter list is built, consisting of the access method value that corresponds to the AM keyword. This value can be referred to by name in another DEBCHK macro by coding AM=(*amaddr*), or it can be inserted into the low-order byte of a register before issuing another DEBCHK macro by coding AM=((*amreg*)).

Return Codes from DEBCHK

Register 15 contains one of the following codes:

Return Code	Meaning
0 (X'00')	The requested function completed successfully and register 1 contains the address of the DEB.
4 (X'04')	Either (a) the DEB table associated with the job step does not exist; or (b) the DEB field that is an index into the DEB table contains an invalid value; or (c) the control block address is not the same as the content of the DEB table entry.
8 (X'08')	An invalid TYPE was specified. (The DEBCHK routine was entered by a branch, not by the macro.)

Return Code	Meaning
12 (X'0C')	Your program was not authorized and TYPE was not VERIFY.
16 (X'10')	DEBDCBAD did not contain the address of the DCB (or ACB) that was passed to the DEBCHK routine.
20 (X'14')	The AM value does not equal the value in the DEBAMTYP field.
24 (X'18')	The DEB is not on the DEB chain and TYPE=ADD was specified.
28 (X'1C')	TYPE=ADD was specified for a DEB that was already entered in the DEB table.
32 (X'20')	The DEB table exceeded the maximum size and TYPE=ADD.
36 (X'24')	TYPE=PURGE,PURGE=FORCE was specified but one or more of the required conditions was not satisfied. The caller must be in system key, supervisor state, hold the local lock, and the passed DEB pointer must exist in the DEB table but not represent a valid DEB.

Obtaining I/O Device Characteristics (DEVTYPE macro)

Use the DEVTYPE macro instruction to request information relating to the characteristics of an I/O device and to cause this information to be placed into a specified area. (The results of a DEVTYPE macro instruction executed before a checkpoint is taken should not be considered valid after a checkpoint/restart occurs.) The IHADVA macro maps the data returned by the DEVTYPE macro - see "IHADVA Mapping macro" on page 270.

The topics that follow discuss the DEVTYPE macro, device characteristics, and the output for specific devices.

Your program can issue the DEVTYPE macro while executing in 24- or 31-bit mode. If your program is executing in 31-bit mode, the parameter list, information list, and the UCB address list can reside above the 16 MB line.

Table 71 on page 266 shows the output for each device type that results from issuing the DEVTYPE macro without the INFOLIST parameter.

For all currently supported devices, DEVTYPE does not return enough information to perform space calculations. TRKCALC should be used to perform space calculations. For information on using the TRKCALC macro, see "Performing Track Calculations (TRKCALC macro)" on page 292.

DEVTYPE Macro Specification

There are four forms of DEVTYPE macro invocation covered here. They are the standard form, execute form, list form, and INFO form.

Restrictions:

- If you do not code the BELOW or ANY value for the UCBLIST parameter, then you can assemble and run DEVTYPE as described in this document on any release of DFSMS.
- If you code the BELOW or ANY value for the UCBLIST parameter, then the program cannot be assembled on MVS/DFP™ Version 3 but the system will ignore the BELOW or ANY value if you assemble it on DFSMS.
- If you code the INFOLIST or INFO parameter, then the macro cannot be assembled or run on MVS/DFP Version 3.
- If you omit INFOLIST, INFO, BELOW and ANY and the PLISTVER parameter, then you can assemble the program on DFSMS but it will not run on MVS/DFP Version 3. If you code PLISTVER=0, then you can assemble the program on DFSMS but not on MVS/DFP and you can run the program on either level of the system. During assembly your program can choose which parameters to code by using the technique described in "Determining DFARELS During Assembler Macro Phase" on page 305.

There are two types of call to the standard form of the DEVTYPE macro.

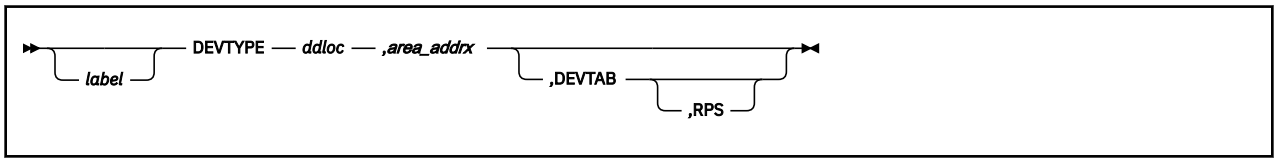
- The minimum type call refers to the DD statement for the device. It has no list or execute forms.
- The UCBLIST/INFOLIST type call requires you to specify either the UCBLIST or INFOLIST parameter or both.

The format of the different types of call to the standard form of DEVTYPE macro follow. The parameter descriptions follow the formats.

Minimum Type Call

The minimum type of call refers to the DD statement for the device. For this type call, as the parameters are passed in general registers no list or execute form exists. It returns the device information to the area you specify.

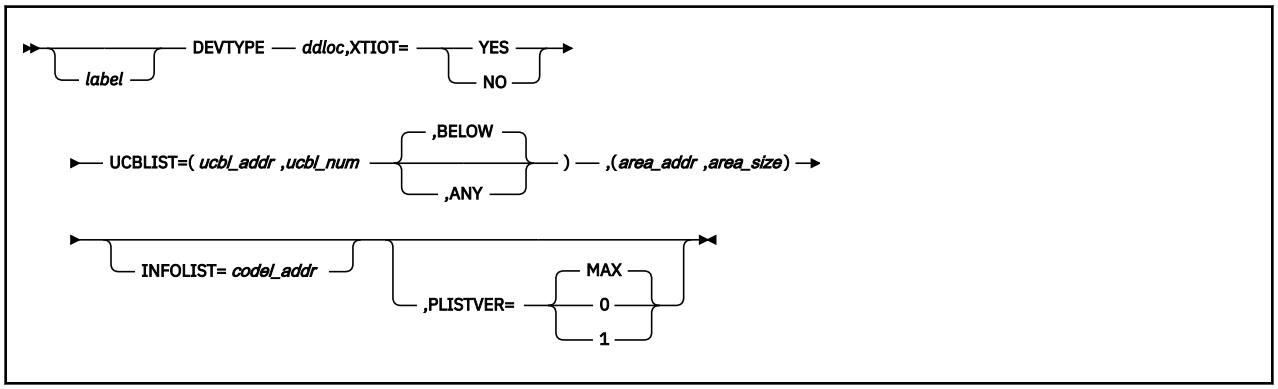
The format of the minimum type call of the DEVTYPE macro is:



UCBLIST or INFOLIST Type of Call

The UCBLIST or INFOLIST type call requires you to specify either the UCBLIST or INFOLIST parameter or both.

The format of the UCBLIST or INFOLIST type call is:



***ddloc*—A-type address or (1-12)**
The name of an 8-byte field that contains the symbolic name of the DD statement to which the device is assigned. The name must be left justified in the 8-byte field, and must be followed by blanks if the name is fewer than eight characters. Each DEVTYPE macro executable invocation must have either *ddloc* to identify an allocated data set or UCBLIST= to identify one or more devices.

You can specify *ddloc* as (1) only if you omit all keywords.

,XTIOT= YES or NO

YES
Standard TIOT entries and XTIOTs are searched to find a matching DD name.

NO
Only entries in the standard TIOTs are searched to find a matching DD name.

***area_addrx*—Rx-type address or (0, 2-12)**
***area_addr*—A-type address or (2-12)**
The name of an area into which the device information is to be placed. If your program does not specify the UCBLIST or INFOLIST function, the area is two, five, or six words long, depending on whether you specify the DEVTAB and RPS operands. If your program specifies the UCBLIST parameter without INFOLIST the area must be 6 words long for each UCB. The area must be on a word boundary.

If your program specifies the INFOLIST parameter, then the length of the area depends on what you coded for INFO on the referenced DEVTYPE macro. The INFO description states how many bytes are returned for each value you specify in the INFO list. To calculate the area size needed, multiply the sum of those values by *ucbl_num* in UCBLIST. If you omit UCBLIST (and specify *ddloc*), do not multiply.

Note that if you specify UCBLIST (and not *ddloc*) then (*area_addr,area_size*) must still be the second positional parameter. If you code (*area_addr,area_size*) before all the keywords, then code it after one comma. If you code (*area_addr,area_size*) after a keyword, then code (*area_addr,area_size*) after two commas.

***area_size*—absolute expression or (2-12)**
The size (in bytes) of the area into which the device information is to be placed. See Table 65 on page 255.

Table 65. Minimum size of area

ddloc specified	UCBLIST specified	INFOLIST specified	Minimum size of area
Yes	No	Omitted or 0	8, 20, or 24 bytes depending on whether DEVTAB and RPS are coded.
Yes	No	Yes	Sum of the number of bytes returned for each code specified with INFO. See also the INFO keyword.
No	Yes	Omitted or 0	24 bytes per UCB
No	Yes	Yes	The product of the number of UCBs specified with UCBLIST and the sum of the number of bytes returned for each code specified with INFO. See also the INFO keyword.

DEVTAB[,RPS]
DEVTAB is only meaningful for direct access devices. If DEVTAB is specified, the following number of words of information is placed in your area:

- For direct access devices: 5 words
- For non-direct access devices: 2 words.

If you do not specify DEVTAB, INFOLIST, or UCBLIST, one word of information is placed in your area if the reference is to a graphics or teleprocessing device; for any other type of device, two words of information are placed in your area.

RPS
If RPS is specified, DEVTAB must also be specified. The RPS parameter causes one additional word of rotational position sensing information to be included with the DEVTAB information.

UCBLIST=*(ucbl_addr,ucbl_num* [,BELOW or ANY])
UCBLIST provides a list service in which the caller passes a list of 4-byte UCB addresses and specifies the number of UCB address entries that are in the list. You must specify the UCBLIST parameter or *ddloc*. The BELOW or ANY are optional keywords that indicate whether the address passed by the UCB parameter contains a 3-byte or 4-byte UCB address. This keyword only applies to callers running in AMODE 31. If the caller is running in AMODE 24, the keyword ANY is ignored and the high-order byte is treated as X'00'.

If you do not specify INFOLIST, then the information returned is always returned in 6-word entries (one entry per UCB address) regardless of the device type. The words that would contain information not applicable to the device for that entry are not altered.

The DEVTYPE macro will accept a captured UCB or an actual UCB address above or below the 16 MB line, via the UCBLIST = parameter. It will also accept 24- or 31-bit addresses of UCB copies. The UCB copy must be on a word boundary, but can be above or below the 16 MB line. Unauthorized programs can get a copy of the UCB by using the UCBSCAN macro and specifying the COPY and UCBAREA keywords. Refer to *z/OS HCD Planning* for details.

***ucbl_addr*—A-type address or (2-12)**
Name of an area containing a list of 4-byte UCB addresses.

***ucbl_num*—absolute expression or (2-12)**
Number of 4-byte UCB address entries in the list.

BELOW
The UCB parameter contains addresses of UCBs that reside in storage below 16 MB, or a captured UCB. This is the default. If BELOW is specified, the high-order byte of the UCB address is treated as X'00'.

ANY
The UCB parameter contains a 4-byte UCB address. If ANY is specified when invoking in 31-bit mode, DEVTYPE treats each word in the UCB address list as a 31-bit address.

Note: The XTIOT keyword is not allowed with UCBLIST.

INFOLIST=*codel_addr*
The name of an area that specifies the types of information DEVTYPE is to return. Coding INFOLIST=0 has the same effect as omitting it. If you specify INFOLIST, then you must also specify *ddloc* or UCBLIST and you must specify *(area_addr,area_size)*. The format of the returned words is described under DEVTYPE, DASD, and SUFFIX (see page "DEVTYPE" on page 261).

***codel_addr*—A-type address or (2-12)**
Specifies an area which contains an instance of the DEVTYPE macro where you coded only the INFO keyword.

PLISTVER=0 or 1 or MAX
Specifies the version of the parameter list for the macro to generate.

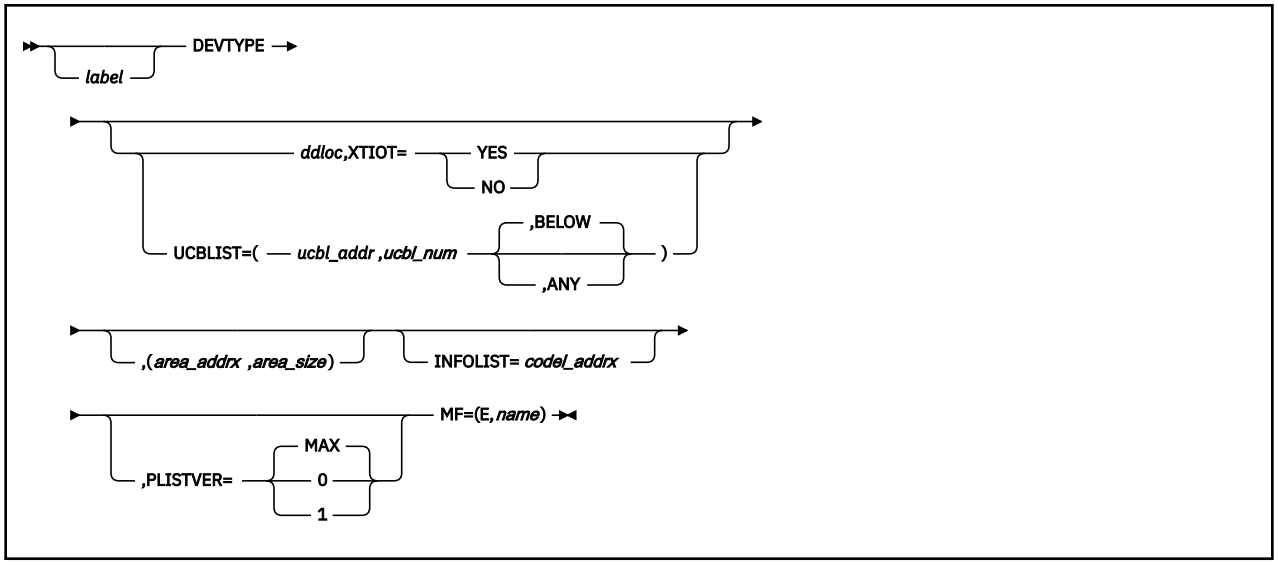
0
The program cannot be assembled on a level of the system earlier than DFSMS/MVS Release 3, which was released in 1996. It can run on any system level that supports the coded parameters. You cannot code this with the INFOLIST or INFO parameter. The parameter list will be 20 bytes long.

1
The program can be assembled and run only on DFSMS/MVS Release 3 or later. The parameter list will not be acceptable for systems prior to DFSMS/MVS. This specification generates a 24-byte parameter list.

MAX
In the current release, this has the same effect as coding 1. In a future release, this value might allow a parameter list that is incompatible with an earlier release. This is the default and generates a 24-byte parameter list.

DEVTYPE—Execute Form

The execute form of the DEVTYPE macro is:



***ddloc*—RX-type address or (1-12)**
Has the same meaning as the standard form of the macro (see also the UCBLIST parameter below for execute form requirements of the *ddloc* parameter).

,XTIOT= YES or NO

YES
Standard TIOT entries and XTIOTs are searched to find a matching DD name.

NO
Only entries in the standard TIOTs are searched to find a matching DD name.

***area_addrx*—RX-type address or (2-12)**
Has the same meaning as the standard form of the macro. This parameter must be coded on the list and/or the execute form.

This must be coded on the execute form.

***area_size*—RX-type address or (2-12)**
This form has the same meaning as the standard form of the macro. This parameter must be coded on the list and/or on the execute form.

This must be coded on the execute form.

UCBLIST=*(ucbl_addrx,ucbl_num*,BELOW or ANY)
This format has the same meaning as the standard form of the macro. You must specify either the UCBLIST or the *ddloc* parameter on either the execute or list form. You can code them on both forms. Either parameter on the execute form overrides the same parameter on the list form.

***ucbl_addrx*—RX-type address or (2-12)**
This format has the same meaning as the standard form of the macro.

***ucbl_num*—absolute expression or (2-12)**
This format has the same meaning as the standard form of the macro except that in the execute form, the maximum value of an absolute expression is 4095. You can supply a larger value in a register.

BELOW
This format has the same meaning as the standard form of the macro. If you write two or three values for UCBLIST on the execute form, it replaces and overrides all three values in the list form. If you omit UCBLIST on the execute form, DEVTYPE will use the values in the list form. This is the default.

ANY
This format has the same meaning as the standard form of the macro. If you write two or three values for UCBLIST on the execute form, it replaces and overrides all three values in the list form. If you omit UCBLIST on the execute form, DEVTYPE will use the values in the list form.

Note: The XTIOT keyword is not allowed with UCBLIST.

INFOLIST=*codel_addrx*
This format has the same meaning as the standard form of the macro. You can code INFOLIST=0 to remove a previous INFOLIST value that is in the list form.

***codel_addrx*—RX-type address or (2-12)**
This format has the same meaning as the standard form of the macro.

PLISTVER=0 or 1 or MAX
This format has the same meaning as on the standard form of the macro. If you code a non-default value for PLISTVER on the list form, then code the same value on the execute form.

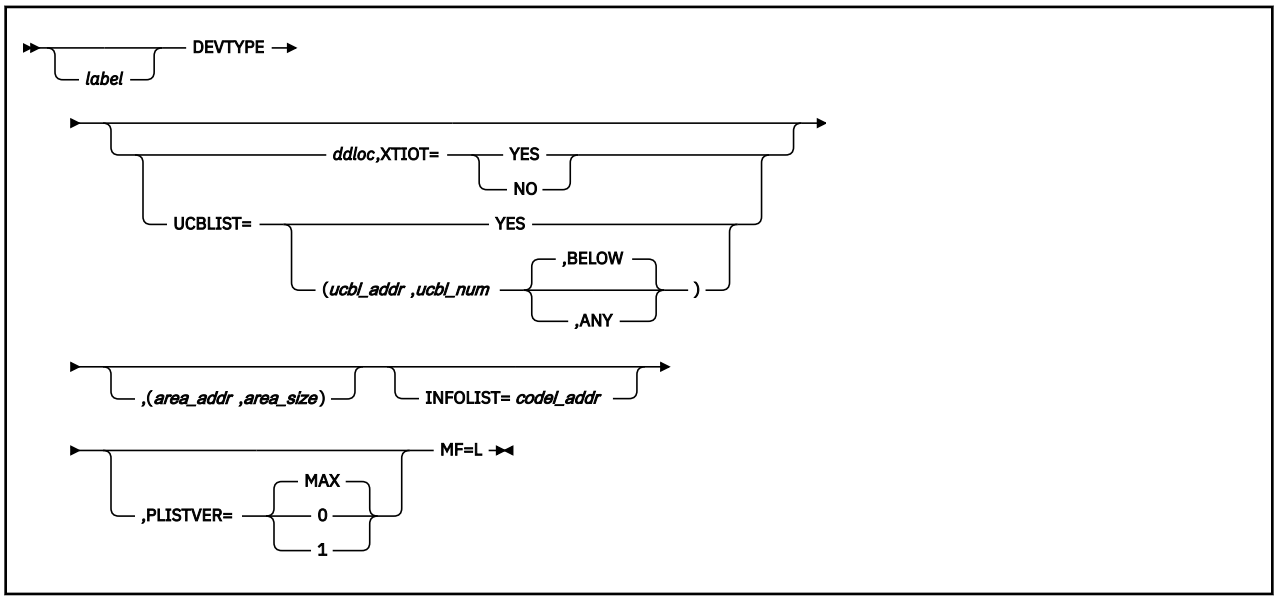
MF=(E,*name*)
Specifies the execute form of DEVTYPE. The execute form of the DEVTYPE macro is valid only with the UCBLIST or INFOLIST function.

E
Specifies the execute form of the macro. Use the execute form to modify a parameter list and call the DEVTYPE function.

***name*—RX-type address or (1-12)**
Label of the parameter list constructed by the corresponding MF=L form.

DEVTYPE—List Form

The list form of the DEVTYPE macro follows:



label
Label of the parameter list to be used. When you specify *label* with MF=L, *label* must be the same as *name* on MF=(E,*name*).

***ddloc*—A-type address**
This format has the same meaning as the standard form of the macro.

,XTIOT= YES or NO

YES
Standard TIOT entries and XTIOTs are searched to find a matching DD name.

NO
Only entries in the standard TIOTs are searched to find a matching DD name.

UCBLIST=YES or (*ucbl_addr,ucbl_num*,BELOW or ANY)
This format has the same meaning as the standard form of the macro. This is the default.

YES
DEVTYPE allows UCBLIST=YES as a place holder when MF=L is coded. It has no effect on the macro expansion.

***ucbl_addr*—A-type address**
This format has the same meaning as the standard form of the macro.

***ucbl_num*—absolute expression.**
This format has the same meaning as the standard form of the macro.

BELOW
This format has the same meaning as the standard form of the macro. This is the default.

ANY
This format has the same meaning as the standard form of the macro.

Coding UCBLIST=YES has no effect on the macro expansion. Do not code UCBLIST=YES with a DD name parameter on the same macro.

The XTIOT keyword is not allowed with UCBLIST.

***area_addr*—A-type address**
This format has the same meaning as the standard form of the macro. This parameter must be coded on the list and/or the execute form.

***area_size*—absolute expression**
This format has the same meaning as the standard form of the macro. This parameter must be coded on the list and/or the execute form.

INFOLIST=*codel_addr*
This format has the same meaning as the standard form of the macro.

***codel_addr*—A-type address or (2-12)**
This format has the same meaning as the standard form of the macro.

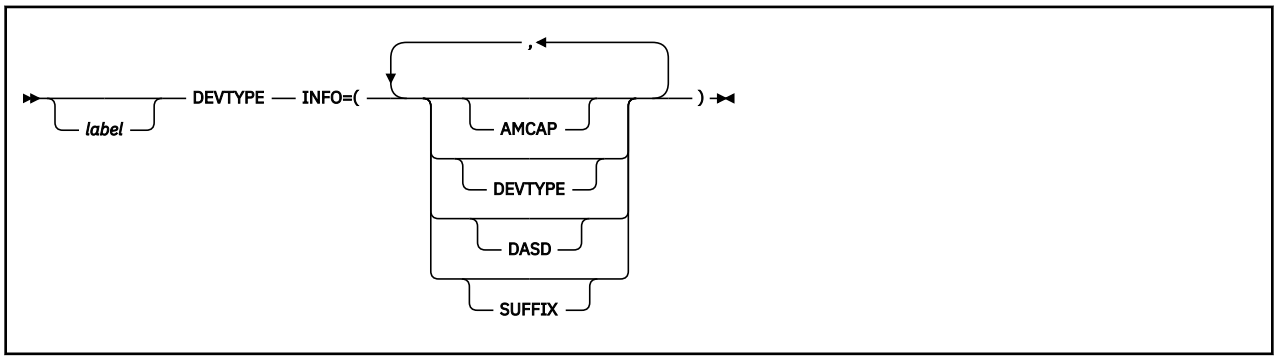
PLISTVER=0 or 1 or MAX
This format has the same meaning as on the standard form of the macro. If you code a non-default value for PLISTVER on the list form, then code the same value on the execute form.

MF=L
Specifies the list form of DEVTYPE. The list form of the DEVTYPE macro is valid only with the UCBLIST or INFOLIST function. By specifying MF=L you construct a parameter list, and you can subsequently supply the values by specifying the execute form of the macro.

DEVTYPE—Info Form

The INFO form of the DEVTYPE macro is used to generate a code list for the INFOLIST parameter. The INFO form is not executable. It has no list or execute form. When you specify the INFOLIST parameter of DEVTYPE, it refers to an expansion of DEVTYPE with only the INFO parameter.

The INFO form of the DEVTYPE macro is:



INFO=(AMCAP or DEVTYPE or DASD or SUFFIX)

Specifies the types of information that you wish to retrieve. Specify any combination of the indicated values in any order. You can specify the same value more than once in the list. The parentheses around the INFO value can be omitted if there is only one value.

Specifying the INFO keyword causes the macro expansion to be a list, and it will not be executable. If you specify the INFO keyword, then omit the MF keyword or specify MF=L. Note that the E-form does not provide for updating any of these values. This means that you can specify INFOLIST to refer to a DEVTYPE expansion that is assembled in a reentrant CSECT.

DEVTYPE will return information in the area that you supply in the *area_addrx* parameter. Each type of information that you request is of a fixed length, as stated below. At execution time, the DEVTYPE macro will check whether the area is long enough. The information will be returned in the order in which you specify the values with the INFO= keyword. If the requested information has no meaning for the device or data set, then DEVTYPE clears the appropriate storage. This means that each piece of information will be returned at a predictable offset in your area.

IBM can add support to DEVTYPE for new INFO= codes in a future level of the system. DEVTYPE in the current level is expected to tolerate object code assembled on such a future level. DEVTYPE in the current level is expected to return the appropriate number of bytes. They should be binary zeroes or partial information padded on the right with binary zeroes. In either case DEVTYPE will also set a return code=0 with a reason code=4 to indicate this condition. If the information returned by the future code cannot validly be zero, then you can test for zeroes to determine which field or fields are not supported. You can also test the DFA to determine the level of the system.

DEVTYPE is not designed to support downward compatibility of source code; that is, you will not be able to assemble source code containing future INFO values on the current level.

AMCAP

Returns 32 bytes for the access method capacity as follows:

Table 66. INFO=AMCAP 32–byte return data

Offset	Bytes	Description
0(0)	1	Flags.
0(0)	1...	BSAM, QSAM, and (if DASD) BPAM support the large block interface and the block size limit is in the next doubleword.
1(1)	7	Reserved, currently set to zeros.
8(8)	8	Maximum block size supported. If you specify a DD name to DEVTYPE for a data set concatenation, this value is the largest for any of the DDs. This value might exceed 32760 for a magnetic tape or dummy data set and therefore require EXCP or the access method large block interface. On output, OPEN does not allow a block size that exceeds this value except with EXCP. On certain cartridge tape drives, exceeding this limit can cause bypassing of hardware buffering. In the future, IBM might support values that exceed 32760 for other device types.
16(10)	8	Recommended maximum block size. This is less than or equal to the maximum block size supported. Above this length the device might be less efficient or less reliable. If you specify a DD name to DEVTYPE for a data set concatenation, this value is the largest for any of the DDs (refer to Table 67 on page 261). Consult hardware documentation for further information.
24(18)	8	Maximum unspanned logical record length supported by BSAM, QSAM, or BPAM. Various types of data sets on the device might have various maximum record lengths. Therefore, if UCBLIST was coded on DEVTYPE and not a DD name, this value is the smallest for the possible data set types for BSAM, QSAM, and BPAM.

Table 67. Optimum and Maximum Block Size Supported When Using EXCP or the Access Method Large Block Interface

Device Type	Optimum	Maximum
DASD	Half track	32 760
Reel tape	32 760	32 760
3480, 3490	65 535	65 535
3490 Emulation (VTS)	262 144 (256 KB)	262 144 (256 KB)
3590	262 144 (256 KB) except on some older models on which it is 229 376 (224 KB)	262 144 (256 KB)
DUMMY	16	5 000 000

DEVTYPE
Returns a copy of the four-byte UCBTYP field of the UCB. See the description of word 0 in "Device Characteristics Information" on page 262.

DASD
Returns 16 bytes as follows:

Bytes 0-3
Number of cylinders on the device, excluding alternates. For a VIO data set, this number is the number of simulated cylinders needed to contain the data set.

Bytes 4-7
Number of tracks per cylinder.

Bytes 8-9
Flags (two bytes)

1... ....
ECKD supported. This means that the following commands are supported:

- Define Extent (X'63') at the beginning of the channel program, except with VIO.
- Locate Record (X'47')
- Read Multiple Count, Key and Data (X'5E')
- Write Count, Key and Data Next Track (X'9D')

For VIO data sets, this bit is on because these commands are always supported. For a non-VIO DASD, this bit also means that the device supports the Define Extent command, but EXCP allows it only at the beginning of your channel program. See "DASD Channel Program Prefix CCW Commands" on page 170.

.1.. ....
Locate Record Extended CCW is supported. For VIO data sets, this bit is on because those commands are always supported.

..1. ....
Controller cache supported.

...1 ....
Flag DVAIXVLD. Flags DVACYLMG (byte 8 bit 4) and DVAEADSCB (byte 8 bit 5), along with field DVAVIRSZ (bytes 14-15) valid and possibly zero.

.... 1...
Flag DVACYLMG. Cylinder-managed space exists on this volume and begins at DVALCYL (bytes 10-11) in multicylinder units of DVAMCU (byte 9). DVAEADSCB (byte 8 bit 5) is also set with this flag on. Valid when DVAIXVLD (byte 8 bit 3) is set.

.... .1..
Flag DVAEADSCB. Extended attribute DSCBs, Format 8 and 9 DSCBs, are allowed on this volume. Valid when DVAIXVLD (byte 8 bit 3) is set.

.... ..1.
Flag DVASSDEV. The device is solid state.

....1
Flag DVACRYPT. Data encrypted device.

Byte 9
Field DVAMCU. Minimum allocation size in cylinders for cylinder-managed space. Each extent in this space must be a multiple of this value. Also referred to as the multicylinder unit (MCU). This is the smallest unit of disk space in cylinders that can be allocated in cylinder managed space. Valid when DVACYLMG (byte 8 bit 4) is set. This field is zero on releases before z/OS 1.10 or if the status is not yet known. In these two cases DVAIXVLD (byte 8 bit 3) is not set.

Bytes 10-11
Field DVALCYL. First cylinder address divided by 4095 where space is managed in multicylinder units. Valid when DVACYLMG (byte 8 bit 4) is set. When valid and zero the volume has no cylinder-managed space. This field is zero on releases before z/OS 1.10 or if the status is not yet known. In these two cases DVAIXVLD (byte 8 bit 3) is not set.

Byte 12
Track set size. Zero if device does not support read-any or write-any. See *3990 Reference* for more information.

Bytes 13
Reserved. DEVTYPE currently returns zeroes but could return something different in a future release.

Bytes 14-15
Block size of the index data set. Valid when byte 8 bit 3 is on. When valid and zero the volume has no working VTOC index. This field is zero on releases before z/OS 1.10 or if the status is not yet known. In these cases byte 8 bit 3 is not set.

SUFFIX
Returns in two bytes the length of the suffix that the system adds to each block in an extended format data set that can be stored on the device. Use this information for space calculations such as with the TRKCALC macro and for determining optimal block size. For device types that support extended format data sets, DEVTYPE returns 32. A non-zero value does not mean that the data set actually is extended format or that the device supports extended format data sets. Not all storage controllers can support extended format data sets. In a future release this value might change.

Device Characteristics Information

The following information is placed into your area as a result of issuing a DEVTYPE macro if you do not code the INFOLIST parameter or if you code INFO=DEVTYPE.

Word 0: Describes the device as defined in the UCBTYP field of the UCB. See the definition of UCBTYP in an expansion of the IEFUCBOB mapping macro. It calls an inner macro, IECDUCBD, that defines additional bits for UCBTYP that are unique for DASD. See the comment that refers to UCBTBYT2. These bit definitions include the read-only device attributes. The IHADVA macro maps these four bytes this way:

Table 68. Device Characteristics Information

Offset	Length	Symbol	Description
0	2	DVAOPTS	Model and option bits that depend on the device
2	1	DVACLASS	Device class. Exactly one bit is on except that X'41' means a channel-to-channel adapter. X'80'=magnetic tape, X'40'=unit record, X'20'=DASD, X'10'=display and X'08'=character reader. A value of X'01' indicates a simulated device that does not have a UCB. These meanings are described in Table 69 on page 263.

Table 68. Device Characteristics Information (continued)

Offset	Length	Symbol	Description
3	1	DVAUNIT	Device type. Depends on the device class. These are common examples: • DASD X'04': 9345 X'0E': 3380 X'0F': 3390 • Tape X'03': 3420 and 3430 X'80': 3480 and 3490 X'81': 3490 Magnetic Tape Subsystem Enhanced Capability X'83': 3590 • Unit record X'01': 2540 Card Read Punch X'03': 1442 or 2596 Card Read Punch X'06': 3505 Card Reader X'08': 1403 Printer X'09': 3211 Printer X'0B': 3203 Printer X'0C': 3525 Card Punch X'0E': 3800 Printer X'0F': 3263, 4245 or 4248 Printer

Simulated Device Characteristics:

Some types of data set reside on simulated devices and do not have a UCB. If you do not code UCBLIST or INFOLIST, DEVTYPE will return data as described by Table 2 in the first two words. The UCB type codes in this table can be returned only if you identify the device by DD name and not if you code UCBLIST. DEVTYPE also can return the word 0 described in Table 69 on page 263 if you code INFO=DEVTYPE.

Table 69. Simulated Device Characteristics Information

Data Set Type	Word 0 in Hexadecimal	Word 1 in Hexadecimal
DUMMY application process queue	0000 0000	0000 0000
TSO terminal	0000 0101	0000 7FF8
SYSIN, SYSOUT, or subsystem (SUBSYS=)	0000 0102	0000 7FF8
z/OS UNIX file or directory	0000 0103	0000 7FF8

The system also uses a copy of the UCBTYP word in these places:

- In catalog entries for DASD and tape data sets. Some of the model and option bits are zero. See output of the IDCAMS LISTCAT command.
- As the device code returned by the LOCATE macro in the volume list. See “Retrieving Information from a Catalog” on page 143. Some of the model and option bits are zero.
- Input to the SCRATCH macro. See “Deleting a Data Set from the VTOC” on page 133 and “SCRATCH and CAMLST Macro Specification” on page 134.
- Input to the RENAME macro. See “Renaming a Data Set in the VTOC” on page 138 and “RENAME and CAMLST Macro Specification” on page 139.

Word 1

Maximum block size without using the large block interface of the access method. The maximum value is 32760 bytes. For direct access devices, this value is the smaller of either the maximum size of a nonkeyed block or the maximum block size allowed by the operating system; for magnetic tape

devices, this value is the maximum block size allowed by the access methods. For these and other device types, see Table 70 on page 265.

If your program specifies either DEVTAB or UCBLIST without INFOLIST, the next three words contain the following information about direct access devices:

Word 2

Bytes 0-1
The number of physical cylinders on the device, including alternates. Treat this as an unassigned 16-bit number.

Recommendation: Before you use bytes 0 and 1, read the description of word 4, byte 1, bit 0. For a VIO data set, that bit is zero, and the number of cylinders is as many as are needed to contain the simulated data set. This can differ from the number for the real device being simulated.

Bytes 2-3
The number of tracks per cylinder.

Word 3

Bytes 0-1
Maximum track length. Note that this value is not equal to the value in word 1 (maximum block size).

Byte 2
Block overhead, keyed block—the number of bytes required for gaps and check bits for each keyed block other than the last block on a track.

Recommendation: Before using bytes 2 and 3, read the description of word 4.

Byte 3
Block overhead—the number of bytes required for gaps and check bits for a keyed block that is the last block on a track.

Bytes 2-3
Block overhead—the number of bytes required for gaps and check bits for any keyed block on a track including the last block. Use of this form is indicated by a 1 in bit 4, byte 1 of word 4.

Basic overhead—the number of bytes required for the count field. Use of this form is indicated by a 1 in bit 3, byte 1 of word 4.

Word 4

Byte 0
Block overhead, block without key—the number of bytes to be subtracted from word 3, bytes 2 or 3 or bytes 2 and 3, if a block is not keyed.

If bit 3, byte 1 of word 4 is 1, this byte contains the modulo factor for a modulo device.

Byte 1

Bit 0
If on, the number of cylinders, as indicated in word 2, bytes 0 and 1 is not valid. If the number of cylinders on the volume exceeds 65520, then this bit is on. To retrieve the number of cylinders for any DASD, you can use the INFO=DASD operand of the DEVTYPE macro.

Bit 1
If on, ECKD supported. This means that the following commands are supported:

- Define Extent (X'63') at the beginning of the channel program, except with VIO.
- Locate Record (X'47')
- Read Multiple Count, Key and Data (X'5E')
- Write Count, Key and Data Next Track (X'9D')

For VIO data sets, this bit is on because these commands are always supported. For a non-VIO DASD, this bit also means that the device supports the Define Extent command, but EXCP

allows it only at the beginning of your channel program. See "DASD Channel Program Prefix CCW Commands" on page 170.

Bits 2-3
If both on, indicates the drive is attached to a cache storage control.

Bit 3
If on, indicates a modulo device (such as 3380, 3390).

Bit 4
If on, bytes 2 and 3 of word 3 contain a halfword giving the block overhead for any block on a track, including the last block.

Bit 5
If on, the device supports paging CCWs.

Bit 6
If on, the device has no alternate cylinders.

Bit 7
If on, a tolerance factor must be applied to all blocks except the last block on the track.

Bytes 2-3
Tolerance factor—this factor is used to calculate the effective length of a block. The calculation should be performed in the following order:

Step 1
Add the block's key length to the block's data length.

Step 2
Test bit 7 of byte 1 of word 4. If bit 7 is 0, perform step 3. If bit 7 is 1, multiply the sum computed in step 1 by the tolerance factor. Shift the result of the multiplication 9 bits to the right.

Step 3
Add the appropriate block overhead to the value obtained above.

If bit 3, byte 1 of word 4 is 1, bytes 2 and 3 contain the overhead for the data or key field.

If your program specifies DEVTAB and RPS, or specifies UCBLIST without INFOLIST, the next word contains the following information:

Word 5

Bytes 0-1
R0 overhead for sector calculations

Byte 2
Number of sectors for each track

Byte 3
Number of data sectors for each track

Table 70 on page 265 and Table 71 on page 266 show the output for each device type that results from issuing the DEVTYPE macro.

If your program specifies UCBLIST and not INFOLIST, the output consists of one 6-word entry for every UCB address contained in the UCB list.

Table 70. Output from DEVTYPE Macro

IBM Device[1]	Maximum Access Method Record Size When Not Using Large Block Interface
2540 Reader	80
2540 Punch	80
2501 Reader	80

Table 70. Output from DEVTYPE Macro (continued)

IBM Device[1]	Maximum Access Method Record Size When Not Using Large Block Interface
3890 Document Processor	80
3505 Reader	80
3525 Punch	80
1403 Printer	120 [1]
3203 Model 5 Printer	132
3211 Printer	132 [1]
3262 Model 5 Printer	132
4245 Printer	132
4248 Printer	132 [2]
3800 or 3900 Printing Subsystem	136 [3]
3410, 3420, 3422, 3424 [4] 3430, 3480, 3490, 3590 Tape Units	32760

Note:

1. Although certain models can have a larger line size, the minimum line size is assumed.
2. The IBM 4248 Printer returns 132 characters even if the 168 Print Position Feature is installed on the device.
3. The IBM 3800 Printing Subsystem can print 136 characters per line at 10-pitch, 163 characters per line at 12-pitch, and 204 characters per line at 15-pitch. The machine default is 136 characters per line at 10-pitch.
4. The 3424 Magnetic Tape Unit is available only in Brazil, S.A.

Table 71. Output from DEVTYPE Macro — DASD Devices

IBM Device	Maximum Record Size (Word 1, Decimal)	DEVTAB (Words 2, 3, and 4, in Hexadecimal)	RPS (Word 5, in Hexadecimal)
3380 Models AD4, AJ4, BD4, BJ4, and CJ2 Disk Storage	32 760	0376 000F BB60 0100 2010 010B	04E0 DED6
3380 Models AD4, AJ4, BD4, BJ4, Disk Storage (attached to a cache storage control)	32 760	0376 000F BB60 0100 2030 010B	04E0 DED6
3380 Models AE4 and BE4 Disk Storage	32 760	06EB 000F BB60 0100 2010 010B	04E0 DED6
3380 Models AE4 and BE4 Disk Storage (attached to a cache storage control)	32 760	06EB 000F BB60 0100 2030 010B	04E0 DED6
3380 Models AK4 and BK4 Disk Storage	32 760	0A60 000F BB60 0100 2010 010B	04E0 DED6
3380 Models AK4 and BK4 Disk Storage (attached to a cache storage control)	32 760	0A60 000F BB60 0100 2030 010B	04E0 DED6
3390 Model 1 (attached to a 3990 Model 2)	32 760	0459 000F E5A2 0000 0052 0000	0594 E000
3390 Model 1 (attached to a 3990 Model 3)	32 760	0459 000F E5A2 0000 0072 0000	0594 E000
3390 Model 2 (attached to a 3990 Model 2)	32 760	08B2 000F E5A2 0000 0052 0000	0594 E000
3390 Model 2 (attached to 3990 Model 3)	32 760	08B2 000F E5A2 0000 0072 0000	0594 E000
3390 Model 3 (attached to a 3990 Model 2)	32 760	0D0B 000F E5A2 0000 0052 0000	0594 E000
3390 Model 3 (attached to a 3990 Model 3)	32 760	0D0B 000F E5A2 0000 0072 0000	0594 E000
3390 Model 3 (attached to a 3990 Model 6)	32 760	0D0B 000F E5A2 0000 0072 0000	0594 E000
3390 Model 9 (attached to a 3990 Model 2)	32 760	2721 000F E5A2 0000 0052 0000	0594 E000
3390 Model 9 (attached to a 3990 Model 3)	32 760	2721 000F E5A2 0000 0052 0000	0594 E000

Table 71. Output from DEVTYPE Macro — DASD Devices (continued)

IBM Device	Maximum Record Size (Word 1, Decimal)	DEVTAB (Words 2, 3, and 4, in Hexadecimal)	RPS (Word 5, in Hexadecimal)
3390 Model 9 (attached to a 3990 Model 6)	32 760	2721 000F E5A2 0000 0052 0000	0594 E000
9345 Model 1	32 760	05A0 000F BC98 0000 0052 0000	04A0 D500

Recommendation: For all currently supported devices, DEVTYPE does not return enough information to perform space calculations. Use the TRKCALC macro and the sector conversion routine to perform space calculations. For information on using the TRKCALC macro, see "Performing Track Calculations (TRKCALC macro)" on page 292. For information on the sector conversion routine, see "Obtaining the Sector Number of a Block on an RPS Device" on page 211.

DEVTYPE—Return Codes and Reason Codes

Control is returned to your program at the next executable instruction following the DEVTYPE macro instruction. Register 15 contains a return code from the DEVTYPE macro, and register 0 contains the reason code. Registers 2 to 14 contents are unchanged. Register 1 contents are unpredictable. The return codes and their meanings are as follows:

Return Code	Meaning
0 (X'00')	Information has been successfully stored in your work area. **Reason Code** **Meaning** **0 (X'00')** All the information is available. **4 (X'04')** DEVTYPE did not recognize one or more INFO parameter codes. DEVTYPE cleared the appropriate amount of the return area and processed the rest of the INFO codes.
4 (X'04')	Invocation error. **Reason Code** **Meaning** **4 (X'04')** DD name not defined. **8 (X'08')** Parameter list not valid. The error might be version code, length, zero field, or return area is not large enough. DEVTYPE does not do a complete UCB check in the current release. The UCB address might not be valid because the correct value was not coded for the third value of the UCBLIST parameter.
8 (X'08')	Unsupported device class. **Reason Code** **Meaning** **12 (X'0C')** DEVTYPE does not support the device class. It must be DASD, tape, subsystem (including spooled), unit record, TSO terminal, dummy, communications, graphics or channel-to-channel adapter. If UCBLIST was coded, then DEVTYPE has ignored the rest of the list.

DEVTYPE—Example 1—Referring to a DD Statement

```
        DEVTYPE MYDD,DEVINFO,DEVTAB
        .
```

```
         .
         .
MYDD     DC      CL8'DATATAB'
DEVINFO  DC      5F'0'
```

Example 1 of DEVTYPE returns 20 bytes of device information if DATATAB is a DASD data set or 8 bytes otherwise.

DEVTYPE—Example 2—Includes Building a Parameter List

```
         MVC   DTLIST,KDTLIST     BUILD PARAMETER LIST IN DYNAMIC STORAGE
*********************************************************************
* RETRIEVE FOUR BYTE UCBTYP FOR SYSUT1 DEVICE
*********************************************************************
         DEVTYPE MF=(E,DTLIST),,(AREA,L'AREA)
         .
         .
         .
*********************************************************************
* RETRIEVE 20 BYTES (DASD INFO AND UCBTYP) FOR THE UNIT DESCRIBED BY THE
* UCB THAT UCBAD POINTS TO.  THE AREA ADDRESS AND LENGTH ARE STILL
* IN THE PARAMETER LIST FROM THE DEVTYPE EXECUTION PERFORMED ABOVE
*********************************************************************
         DEVTYPE UCBLIST=(UCBAD,1),INFOLIST=ILIST2,MF=(E,DTLIST)
         .
         .
         .
KDTLIST  DEVTYPE FIRSTDD,MF=L,INFOLIST=ILIST1  NON-MODIFIABLE PARAMETER
*                                                 LIST
LDTLIST  EQU   *-KDTLIST
FIRSTDD  DC    CL8'SYSUT1'
ILIST1   DEVTYPE INFO=DEVTYPE
ILIST2   DEVTYPE INFO=(DASD,DEVTYPE)  REQUEST DATA AT AREA
DYNAMIC  DSECT
UCBAD    DS    A                      ADDRESS OF UCB
DTLIST   DS    CL(LDTLIST)            DEVTYPE PARAMETER LIST(MODIFIABLE)
         DS    0F                     ALIGNMENT FOR TRKCYL
AREA     DS    0CL20                  INFORMATION FROM
*                                       DEVTYPE INFO=(DASD,DEVTYPE)
TYP1     DS    0CL4                   INFORMATION FROM
*                                       DEVTYPE INFO=DEVTYPE (UCBTYP)
*                                       FOR FIRSTDD
NUMCYL   DS    F                      NUMBER OF CYLINDERS ON VOLUME
TRKCYL   DS    F                      NUMBER OF TRACKS PER CYLINDER
         DS    CL8                    MISCELLANEOUS
TYP2     DS    CL4                    UCBTYP FROM UCB POINTED TO
*                                       FROM UCBAD
```

Example 2 of DEVTYPE builds a parameter list in dynamically-acquired storage so the program can be reentrant. It then supplies additional parameters in the first execute form and overrides some of them in the second execute form. In effect, the first specification of DEVTYPE is:

```
DEVTYPE FIRSTDD,(AREA,L'AREA),INFOLIST=ILIST1
```

ILIST1 describes four bytes to be returned. They are at the beginning of a 20-byte area - DEVTYPE clears the extra 16 bytes. The list form at KDTLIST specifies parameters that will not be overridden by the first execute form. The execute form specifies parameters that are determined during execution. In effect, the second specification of DEVTYPE is:

```
DEVTYPE ,(AREA,L'AREA),INFOLIST=ILIST2,UCBLIST=(UCBAD,1)
```

The INFOLIST describes 20 bytes to be returned.

The first execute form illustrates an unusual technique of coding a keyword parameter (MF) before two positional parameters. The first positional value is null, and the second position is (AREA,L'AREA). This

generally is not a good technique because it is confusing. It is used here only to show the flexibility that Assembler H and High Level Assembler allow.

For another example of DEVTYPE, see Figure 34 on page 306.

DEVTYPE—Example 3—Building a Parameter List and Using IHADVA

This example is the same as the previous one, but it uses the IHADVA macro to map the DEVTYPE output and several unusual techniques to demonstration the possibilities.

```
         MVC DTLIST,KDTLIST    Build parameter list in dynamic storage
*************************************************************************
* Retrieve four-byte UCBTYP for SYSUT1 device in 20-byte area.
*************************************************************************
         DEVTYPE ,(DVAIDASD,L'DVAIDASD+LenDEVTYPE),MF=(E,DTLIST)
DTUsing  USING DVAUCBTY,DVAIDASD        Map DSECT on CSECT
           *
           *
*************************************************************************
* Retrieve 20 bytes (DASD INFO and UCBTYP) for the unit described by
* the UCB that UCBAD points to.  The area address and length are still
* in the parameter list from the DEVTYPE execution performed above.
*************************************************************************
         DEVTYPE UCBLIST=(UCBAD,1),INFOLIST=ILIST2,MF=(E,DTLIST)
DTUsing  USING DVAUCBTY,DVAIDASD+L'DVAIDASD   Map DSECT on CSECT
           *
           *
KDTLIST  DEVTYPE FIRSTDD,MF=L,INFOLIST=ILIST1  Read-only parameter list
LDTLIST  EQU   *-KDTLIST         Length of parameter list
FIRSTDD  DC    CL8'SYSUT1'       DD name
ILIST1   DEVTYPE INFO=DEVTYPE    DEVTYPE INFO list
ILIST2   DEVTYPE INFO=(DASD,DEVTYPE)  Another INFO list
* End of CSECT.
DYNAMIC  DSECT      **** Dynamic storage for reentrancy
UCBAD    DS    A                  Address of UCB
DTLIST   DS    CL(LDTLIST)        DEVTYPE parameter list (modifiable)
         DS    0F                 Alignment for efficiency
* Output from INFO=(DASD,DEVTYPE) begins here.
* Next line defines symbols for DVAIDASD, which is 16 bytes.
         IHADVA DSECT=NO,INFO=DASD Info from DEVTYPE INFO=DASD
         DS    CL(LenDEVTYPE) Info from DEVTYPE INFO=DEVTYPE
* Next line defines the DVAUCBTY DSECT, which is 4 bytes.
         IHADVA DSECT=YES,INFO=DEVTYPE Info from DEVTYPE INFO=DEVTYPE
LenDEVTYPE EQU *-DVAUCBTY          Length of DSECT (4 b
```

Example 3 of DEVTYPE builds a parameter list in dynamically-acquired storage so the program can be reentrant. It then supplies additional parameters in the first execute form and overrides some of them in the second execute form. In effect, the first specification of DEVTYPE is:

```
DEVTYPE FIRSTDD,(DVAIDASD,L'DVAIDASD+LenDEVTYPE),INFOLIST=ILIST1
```

ILIST1 describes four bytes to be returned at the beginning of a 20-byte area. The length attribute of the INFO=DASD area is L'DVAIDASD, which is 16. The constant LenDEVTYPE is the length of the INFO=DEVTYPE area, which is four bytes. DEVTYPE clears the extra 16 bytes. The list form at KDTLIST specifies parameters that will not be overridden by the first execute form. The execute form specifies parameters that are determined during execution.

The `DTUsing USING DVAUCBTY,DVAIDASD` line is a labeled dependent USING statement. It applies the 4-byte DVAUCBTY DSECT on top of the storage at DVAIDASD. One purpose of the symbol DTUsing could be to allow a later DROP statement to end the addressability but that is not the purpose of the label here. The purpose here is to avoid an assembler warning message ASMA303W. Multiple address resolutions may result from this USING and the USING on statement number nn". It would be on the second labeled USING if it did not have the same label.

In effect, the second specification of DEVTYPE is:

```
DEVTYPE ,(L'DVAIDASD+LenDEVTYPE),INFOLIST=ILIST2,UCBLIST=(UCBAD,1)
```

The INFOLIST describes 20 bytes to be returned. The second USING with the label DTUsing defines addressability to the last four of the 20 bytes. The first 16 bytes have no USING because they are a part of the DYNAMIC DSECT and already have addressability that is not shown. This technique of using one DSECT (defined by `IHADVA DSECT=YES,INFO=DEVTYPE`) apply to two places in different areas of the program is allowed by named USINGs without loading another base register.

The `IHADVA DSECT=NO,INFO=DASD` line defines 16 bytes of variables that DEVTYPE sets. The first symbol is DVAIDASD and its length attribute is 16.

The `DS CL(LenDEVTYPE)` line defines variables that are described by the DSECT generated by the `IHADVA DSECT=YES,INFO=DEVTYPE` line.

IHADVA Mapping macro

The IHADVA macro supports two parameters:

DSECT={YES|NO}
If you code DSECT=YES, you get a single area with a DSECT. This is the default. Its name depends on whether you code INFO= and what you code for it. The DSECT name depends on the first value that you code for INFO=.

The following applies if you code DSECT=NO:

- If you omit INFO= or you code INFO=NONE, then the area begins with the symbol DVAREA and it is not a DSECT.
- If you code any combination of INFO values other than NONE, then DVAREA is not defined and there is no DSECT.

INFO={NONE|DEVTYPE|DASD|SUFFIX|AMCAP}
If you omit the INFO keyword, then the mapping is for all of the following at the same origin:

- - the minimum type of call
- UCBLIST= without INFO=
- INFO=DASD
- INFO=DEVTYPE
- INFO=SUFFIX

INFO=NONE.
This generates the mapping for the minimum type of call or when you code UCBLIST= without INFO=. You cannot code NONE in combination with any other value.

INFO=DASD
Generate the mapping for the area returned by coding INFO=DASD. The DSECT name or first symbol is DVAIDASD.

INFO=DEVTYPE
Generate the mapping for the area returned by coding INFO=DEVTYPE. The DSECT name or first symbol is DVAUCBTY.

INFO=SUFFIX
Generate the mapping for the area returned by coding INFO=SUFFIX. The DSECT name or first symbol is DVASUFFX.

INFO=AMCAP
Generate the mapping for the area returned by coding INFO=AMCAP. The DSECT name or first symbol is DVAAMCAP.

```
=====================================================================
              DEVTYPE return area (mapping macro IHADVA)
THIS MACRO MAPS THE AREA RETURNED TO THE CALLER BY THE DEVTYPE SVC
=====================================================================


 OFFSET
DEC(HEX)  TYPE      LEN  NAME             DESCRIPTION
```

```
======== ========  ===== =========        ============================
 0  (0)  STRUCTURE   24  DVAREA
 0  (0)  CHARACTER    8   DVAPREFX        Area if no INFOLIST=, DEVTAB
                                          or RPS

            Following four bytes are also returned for INFO=DEVTYPE
 0  (0)  CHARACTER    4    DVAUCBTY       UCB TYPE FIELD
 0  (0)  BITSTRING    2     DVAOPTS       UCB OPTIONS
 2  (2)  BITSTRING    1     DVACLASS      DEVICE CLASS
 3  (3)  BITSTRING    1     DVAUNIT       UNIT TYPE
 4  (4)  SIGNED       4    DVAMAXRC       MAXIMUM RECORD SIZE
 8  (8)  CHARACTER   12    DVATAB         SECTION INCLUDED BY DEVTAB
 8  (8)  UNSIGNED     2    DVACYL         PHYS NUMBER CYL PER VOLUME
10  (A)  SIGNED       2    DVATRK         NR OF TRACKS PER CYL
12  (C)  SIGNED       2    DVATRKLN       TRACK LENGTH ( BYTES)
14  (E)  SIGNED       2    DVAOVHD        BLOCK OVERHEAD IF DVA2BOV IS
                                          ON

            IF DVA2BOV IS OFF USE INSTEAD THE FOLLOWING TWO VALUES
14  (E)  ADDRESS      1     DVAOVNLB      OVERHEAD NOT LAST BLOCK
15  (F)  ADDRESS      1     DVAOVLB       OVERHEAD LAST BLOCK
16 (10)  ADDRESS      1    DVAOVNK        OVERHEAD DECR IF NOT KEYED
17 (11)  BITSTRING    1    DVAFLAGS       FLAG BYTE
         1... ....          DVABDCYL      IF 1, DVACYL IS INVALID
                                          YL02130
         .1.. ....          DVADEFLR      DEFINE EXTENT/LOCATE RECORD
                                          AND RELATED TRANSFER COMMANDS
                                          ARE IMPLEMENTED
         ..1. ....          DVADEFEX      DEFINE EXTENT IMPLEMENTED
         ...1 ....          DVAMODL       IF ON, USE MODULO TRACK
                                          ALGORTIHM
         .... 1...          DVA2BOV       IF ON, USE DVAOVHD ELSE USE
                                          DVAOVNLB & DVAOVLB
         .... .1..          DVAPAGES      IF ON DEVICE SUPPORTS PAGING
                                          CCWS
         .... ..1.          DVANOALT      NO ALT TRKS AVAILABLE
         .... ...1          DVAFTOL       IF ON, APPLY TOLERANCE FACTOR
18 (12)  SIGNED       2    DVATOL         TOLERANCE FACTOR
            (BLKSI+KEYLE) DVATOL/DVADVSR GIVES THE ADJUSTED BLOCK SIZE
            TO WHICH APPROPRIATE OVERHEADS ARE THEN ADDED.

20 (14)  CHARACTER    4   DVARPS         RPS SECTION
20 (14)  SIGNED       2    DVAOVR0       OVERHEAD BYTES FOR RECORD 0
22 (16)  ADDRESS      1    DVASECT       NUMBER OF SECTORS IN FULL
                                          TRACK
23 (17)  ADDRESS      1    DVASECTD      NUMBER OF DATA SECTORS

   ================================================================
   THE FOLLOWING SECTION IS RETURNED BY DEVTYPE FOR INFO=DASD.
   ================================================================

 0  (0)  STRUCTURE   16  DVAIDASD
 0  (0)  UNSIGNED     4   DVAICYL        NUMBER OF CYLINDERS
 4  (4)  UNSIGNED     4   DVAITRK        TRACKS PER CYLINDER
 8  (8)  UNSIGNED     1   DVAIFLAG       FLAGS
         1... ....          DVAECKD1     ECKD SUPPORTED, ALSO ON FOR
                                          VIO DATA SETS
         .1.. ....          DVALRE1      LOCATE RECORD EXTENDED
                                          SUPPORTED
         ..1. ....          DVACACHE1    DEVICE IS CACHED
         ...1 ....          DVAIXVLD     DVACYLMG, DVAEADSCB, DVAVIRSZ
                                          valid.
         .... 1...          DVACYLMG     Cylinder-managed space exists
                                          on this volume and begins at
                                          DVALCYL in multicylinder units
                                          of DVAMCU. DVAEADSCB is also
                                          set with this flag on. Valid
                                          when DVAIXVLD is set.
         .... .1..          DVAEADSCB    Extended attribute DSCBs,
                                          Format 8 and 9 DSCBs, are
                                          allowed on this volume. Valid
                                          when DVAIXVLD is set.
         .... ..1.        DVASSDEV       The device is solid state
         .... ...1        DVACRYPT       Data encrypted device.
 9  (9)  UNSIGNED     1   DVAMCU         Minimum allocation size in
                                          cylinders for cylinder-managed
                                          space. Each extent in this
                                          space must be a multiple of
                                          this value. space. Also
                                          referred to as the
                                          multicylinder unit (MCU). This
                                          is the smallest unit of disk
```

```
                                          space in cylinders that can be
                                          allocated in cylinder-managed
                                          space. Valid when DVACYLMG is
                                          set. This field is zero on
                                          releases before z/OS 1.10 or
                                          if the status is not yet
                                          known. In these two cases
                                          DVAIXVLD is not set.
10  (A)  UNSIGNED     2   DVALCYL        First cylinder address divided
                                          by 4095 where space is managed
                                          in multicylinder units.
                                          Cyl-managed space begins at
                                          this address. Valid when
                                          DVACYLMG is set. This field is
                                          zero on releases before z/OS
                                          1.10 or if the status is not
                                          yet known. In these two cases
                                          DVAIXVLD is not set.
12  (C)  UNSIGNED     1   DVAITSET       TRACK SET SIZE
13  (D)  UNSIGNED     1   *              Reserved. DEVTYPE currently
                                          returns zeroes but could
                                          return something different in
                                          a future release.
14  (E)  UNSIGNED     2   DVAVIRSZ       Block size of the index data
                                          set. Valid when DAVIXVLD is
                                          set on. When valid and zero
                                          the volume has no working VTOC
                                          index. This field is zero on
                                          releases before z/OS 1.10 or
                                          if the status is not yet
                                          known. In these cases DVAIXVLD
                                          is not set.

   ================================================================
   THE FOLLOWING SECTION IS RETURNED BY DEVTYPE FOR INFO=AMCAP.
   ================================================================
 0  (0)  STRUCTURE   32  DVAAMCAP        ACCMETH
                                          CAPABILITY
 0  (0)  BITSTRING    1   DVAAMFLG       FLAGS
         1... ....         DVAAMLBI      BSAM, QSAM AND (IF DASD) BPAM
                                          SUPPORT THE LARGE BLOCK
                                          INTERFACE & THE LIMIT IS IN
                                          THE NEXT DOUBLEWORD.
         .1.. ....         DVAAM_XTIOT   This data set allocation has an
                                          XTIOT. Either all or none of the
                                          entries for a concatenation are
                                          XTIOT.
         ..1. ....         DVAAM_XTIOTAM BSAM, QSAM and BPAM (if DASD)
                                          support XTIOT for this device,
                                          and the NON_VSAM_XTIOT option in
                                          PARMLIB allows it. DEVTYPE will
                                          turn this on if the UCB is DASD
                                          or tape or the DD is dummy and
                                          the PARMLIB option allows
it.
         ...1 ....         DVAAM_31UCB   One or more UCB addresses for
                                          this data set allocation (or
                                          concatenation) point above the 16
                                          MB and have not been captured for
                                          the allocation. If this bit is
                                          off, the data set still might be
                                          extended to another volume and
                                          gain a 31-bit address UCB.
         .... 1...         DVAAM_31UCBAM BSAM, QSAM and BPAM (if DASD)
                                          support 31-bit UCB addresses in
                                          the DEB and the NON_VSAM_XTIOT
                                          option in PARMLIB allows it.

         .... .1..         DVAAM_DSAB    DSAB is above the line.
         .... ..1.         DVAAM_DSABAM  BSAM, QSAM and BPAM (if DASD)
                                          support DSAB above the line and
                                          the NON_VSAM_XTIOT option in
                                          PARMLIB allows it.
 1  (1)  CHARACTER    7   *              RESERVED
 8  (8)  BITSTRING    8   DVAMAXBLK      MAXIMUM BLOCK SIZE SUPPORTED
                                          WITH
                                          SAM LBI
16 (10)  BITSTRING    8   DVAOPTBLK      RECOMMENDED MAXIMUM BLOCK SIZE
                                          LONGER BLOCKS MIGHT BE LESS
                                          EFFICIENT OR LESS RELIABLE.
                                          LESS THAN OR EQUAL TO PREVIOUS
                                          FIELD.
```

```
24 (18)  BITSTRING    8   DVAMAXLR        MAXIMUM UNSPANNED LOGICAL
                                           RECORD
                                           LENGTH SUPPORTED BY BSAM, QSAM
                                           AND BPAM
   ================================================================
   THE FOLLOWING SECTION IS RETURNED BY DEVTYPE FOR INFO=SUFFIX.
   ================================================================
0  (0)   SIGNED       2   DVASUFFX        SUFFIX LENGTH
```

Reading and Modifying a Job File Control Block (RDJFCB Macro)

To accomplish the functions that are performed as a result of an OPEN macro instruction, the open routine requires access to information that you have supplied in a data definition (DD) statement. This information is stored by the system in a job file control block (JFCB). Some information is placed into the JFCB when the data set is allocated, while other information is placed there only when the data set is opened. Which fields are updated and when they are updated will vary depending upon factors such as what is specified in the JCL DD statement, what the application does, whether the data set is SMS managed or not, and so on. Fields that have not been updated yet will contain binary zeroes.

In certain applications, you might find it necessary to modify the contents of a JFCB (previously specified in the allocation parameters) before issuing an OPEN macro instruction against a data set. For example, lets suppose that you are adding records to the end of a sequential data set. You might want to add a secondary allocation quantity to allow the existing data set to be extended when the space currently allocated is exhausted. To assist you, the system provides the RDJFCB macro instruction. This macro instruction causes a JFCB to be moved to an area specified in an exit list. Use of the RDJFCB macro instruction with an exit list is shown under “Example” on page 276. When you subsequently issue the OPEN macro instruction, you can specify the TYPE=J operand to open the data set using the JFCB in the area you specified.

You can use RDJFCB and a DCB to learn the data set name, AMP parameters and volume serials of a VSAM data set. You can use any valid combination of MACRF and DSORG in the DCB. The simplest would be DSORG=PS,MACRF=E. That is for an EXCP DCB. DSORG=PS,MACRF=R also will work but it requires more system storage below the 16 MB line.

If you specify the XTIOT, UCB NOCAPTURE or DSAB-above-the-line options of dynamic allocation, then the system creates an XTIOT. With these options, if the access method is not EXCP and the data set is not VSAM, then RDJFCB requires that you code the LOC=ANY option on the DCBE macro, and the NON_VSAM_XTIOT=YES option in the DEVSUPxx member of SYS1.PARMLIB is in effect.

The RDJFCB macro also allows you to retrieve allocation information for the data sets in a concatenation. You can either select data sets or, by default, retrieve the information for all data sets in the concatenation.

You can retrieve the following items:

- All JFCBs
- All volume serial numbers
- Block size limit
- The path name (PATH=) associated with any DD. In these cases the data set name in the JFCB is a dummy value.

“Type 07 JFCB Exit List Entry” on page 277 describes how you can use RDJFCB to retrieve this information.

Tip: If you set the bit JFCNWRIT in the JFCBTSDM field to 1 before you issue the OPEN macro instruction, the JFCB is not written back at the conclusion of open processing. OPEN TYPE=J normally moves your program's modified copy of the JFCB, to replace the system copy. To ensure that this move is done, your program must set JFCBMODJ (bit zero of the JFCBMSK5 byte to 1. IBM recommends not setting on JFCNWRIT. If the user JFCB (which the system used to open the data set) is not written back, errors can occur during termination processing for EOV, CLOSE, or the job/step because OPEN might have updated information in the user JFCB which will not be reflected in the system copy of the JFCB. For example, when a nonspecific tape data set is opened, OPEN will update the user supplied JFCB with the volume

serial number of the tape selected. However, the system copy of the JFCB will not reflect this volume serial number. This could cause errors during termination processing for EOV, CLOSE, or the job/step (for example, the data set might not be cataloged even though the job requested it).

Tip: If your program has a DCBE, which has a value for BLKSIZE, which is too large for the two bytes in the JFCB, the OPEN TYPE=J normally copies it to another system control block. If your program sets JFCNWRIT on, that setting may prevent OPEN from storing the DCBE BLKSIZE value into the system control block. This may lead to further system errors during data set processing.

Your program can also use the SWAREQ macro to access JFCBs and JFCBXs. It requires your program to access the DSAB or DSABs and the TIOT or XTIOTs. You can use the GETDSAB macro for this and you will need several mapping macros. SWAREQ and GETDSAB are documented in *z/OS MVS Programming: Authorized Assembler Services Guide*. The RDJFCB macro is designed to be simpler to use and does not require examining system control blocks.

Some of the modifications that can be made to the JFCB include:

- Moving the creation and expiration date fields of the DSCB into the JFCB
- Modifying the number-of-volumes field in the JFCB

  The number-of-volumes field can be modified only to be a value not greater than the total number of volume serial numbers available in the JFCB and any JFCBXs (extensions). The JFCB can have five volume serial numbers. Each JFCBX can have 15 volume serial numbers. Whether or not a JFCBX is required and how many JFCBXs are required is determined during data set allocation. A JFCBX cannot be dynamically created after allocation except when a tape data set is extended to a new volume. Therefore, the maximum value of the number-of- volumes field is based on the JFCB and how many JFCBXs exist. Setting the number-of-volumes field to a value greater than that maximum will not cause a JFCBX to be dynamically created.
- Moving the DCB fields from the DSCB into the JFCB
- Adding volume serial numbers to the JFCB (see “RDJFCB Security” on page 278)

  Volume serial numbers in excess of five are written to the JFCBX (extension). The JFCBX cannot be modified by user programs.
- Modifying the sequence number field of the volume in the JFCB. For DISP=NEW the modified volume sequence will be honored during OPEN TYPE=J only for tape and only when the file sequence number in the JFCB is also modified.
- Modifying the sequence number field of the data set in the JFCB. This specifies the file for a subsequent OPEN TYPE=J to process. You can use this technique to write many data sets on a tape volume or set of volumes. Use RDJFCB and increase the sequence number of the data set by 1 before each OPEN TYPE=J. If the device supports buffered tape marks, you can obtain significantly better performance by requesting the SYNC=NONE option on the DCBE macro. It allows the device to enter "streaming mode" and to write tape marks much faster. A side effect is that any I/O error can be reflected at an unpredictable time. This can result in an ABEND in OPEN or CLOSE for a user data block. It also means that if a device malfunction occurs, it might result in the loss of more than one data set that the application had closed previously. To position rapidly to a tape data set other than the next data set, you can use the technique described in “High-Speed Cartridge Tape Positioning” on page 286.
- Changing the data set name field or member field in the JFCB. See “RDJFCB Security” on page 278 and “RDJFCB Use by Authorized Programs” on page 278. You can open a VTOC by reading the JFCB, changing the data set name to 44 bytes of X'04' and then issuing the OPEN macro with TYPE=J. Use BSAM or EXCP. If you use BSAM, also code RECFM=F, KEYLEN=44 and BLKSIZE=96 on the DCB macro. For an extended address volume the DCB macro must point to a DCBE where the EADSCB=OK keyword is specified. You'll find examples of opening an EXCP DCB for a VTOC in “Example of Using the CVAFFILT Macro” on page 99 and “Example of using the CVAFSEQ macro to process a volume in physical sequential order” on page 118.
- Setting bit JFCDQDSP in field JFCBFLG3 to invoke the tape volume DEQ at demount facility (see “DEQ at Demount Facility for Tape Volumes” on page 285)

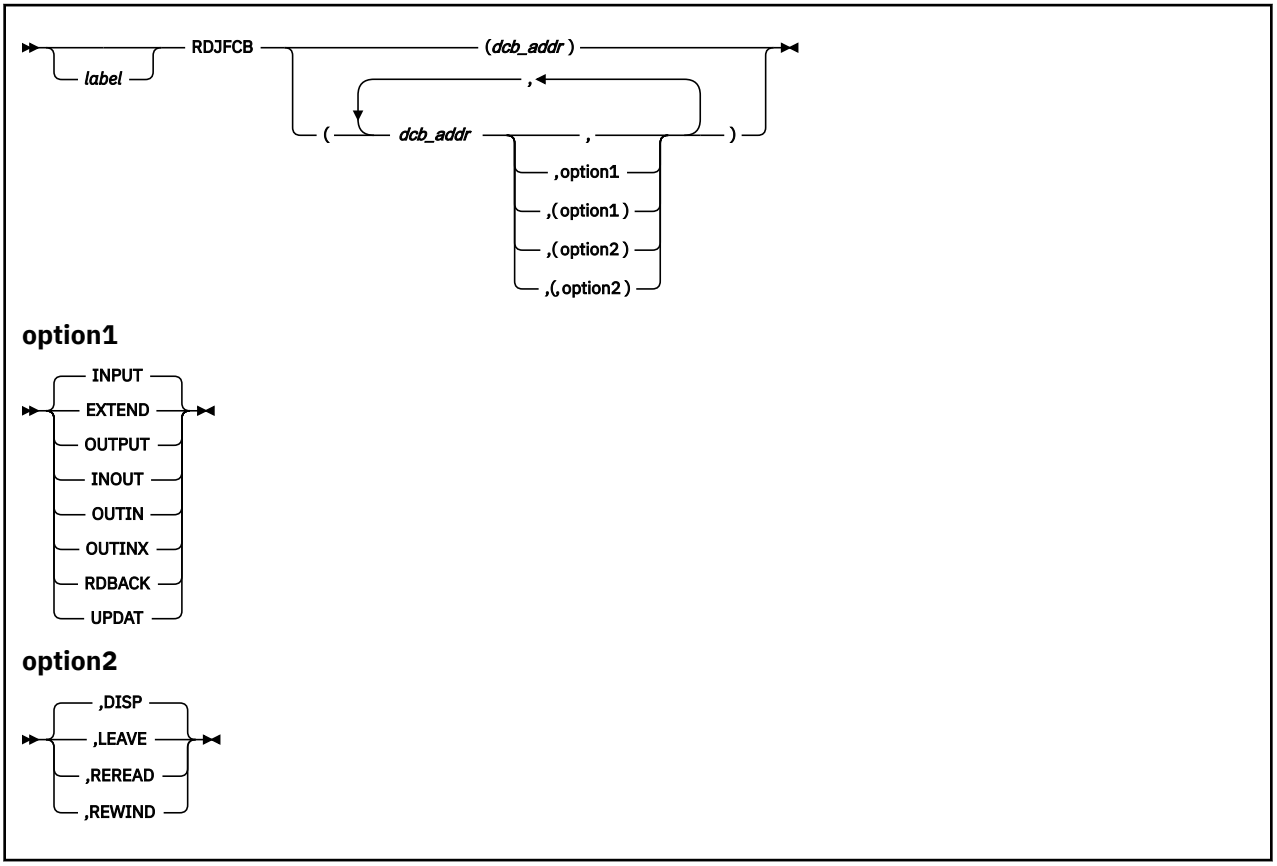
- Modifying the JFCRBIDO field in the JFCB to cause high-speed positioning to a specific data block on a tape volume on a device that supports cartridge tapes (see "High-Speed Cartridge Tape Positioning" on page 286)
- Setting bit JFCNDSCB to prevent OPEN from merging fields from the data set label to the JFCB. This means that you must supply the information from other sources, such as the JFCB or the DCB. A data set label is a DSCB or standard tape label. This bit does not control all fields. For example it does not control the creation and expiration dates. Note that if you set this bit on, it can cause incorrect positioning in the data set after an automatic step restart when using DISP=MOD or the OPEN macro with the EXTEND option. Setting this bit on can interfere with the system correcting JFCBDSCB (TTR of DSCB on first volume). Setting this bit on is not a good programming practice because it depends on internal system logic.
- Setting bit JFCNDCB to prevent OPEN from merging fields from the DCB to the JFCB. This interferes with correct information being set in the data set label and it interferes with OPEN doing certain checks. Setting this bit on is not a good programming practice because it depends on internal system logic.
- Setting bit JFCBLKSZ to indicate that the JFCB block size field has been set to zero by the application and that OPEN needs to set the block size value in the SIOT extension, if applicable, to zero as well. OPEN will also reset JFCBLKSZ off.

The secondary allocation quantity will be moved from the DSCB into the JFCB unless prevented by the setting of JFCNWRIT or JFCNDSCB.

RDJFCB Macro Specification

The RDJFCB macro instruction moves a job file control block (JFCB) into an area of your choice as identified by the EXLST parameter of the DCB macro for each data control block specified.

The format of the RDJFCB macro is:



Tip: If you wish to have multiple DCBs with or without options, code each DCB (and options) as shown in the diagram and precede each additional DCB with a comma. Examples of the standard form of the RDJFCB macro are:

```
RDJFCB (DCB1)
RDJFCB (DCB1,INPUT)
RDJFCB (DCB1,(INPUT))
RDJFCB (DCB1,(INPUT,REREAD))
RDJFCB (DCB1,,DCB2)
RDJFCB (DCB1,,DCB2,(INPUT,REREAD),DCB3,INPUT)
```

Figure 29. Examples of standard form of the RDJFCB macro

dcb_address,* or *(options)
(Same as the *dcb_address*, *option1*, and *option2* operands of the OPEN macro instruction, as shown in *z/OS DFSMS Macro Instructions for Data Sets*), except for the MODE operand, which is not valid with the RDJFCB macro.

The option operands do not affect RDJFCB processing. You can, however, specify them in the list form of the RDJFCB macro instruction and refer to the generated parameter list with the execute form of the macro.

- You can also use the MF parameter on an RDJFCB macro. Its syntax, use, and effect are the same as is documented for the OPEN macro in *z/OS DFSMS Macro Instructions for Data Sets*. In addition, you can code an execute-form RDJFCB macro that refers to a list-form OPEN macro that does not have MODE=31.
- The RDJFCB parameter list, the DCB, and the JFCB area specified in the exit list as well as the exit list itself must reside below 16 MB, although the calling program can be above 16M.

Example

In Figure 30 on page 276, the macro instruction at EX1 creates a parameter list for two data control blocks: INVEN and MASTER. In creating the list, both data control blocks are assumed to be opened for input; option2 for both blocks is assumed to be DISP. The macro instruction at EX2 moves the system-created JFCBs for INVEN and MASTER into the area you specified, thus making the JFCBs available to your problem program for modification. The macro instruction at EX3 modifies the parameter list entry for the data control block named INVEN and indicates, through the TYPE=J operand, that the problem program is supplying the JFCBs for system use.

```
EX1       RDJFCB (INVEN,,MASTER),MF=L
          .
          .
          .
EX2       RDJFCB MF=(E,EX1)
          .
          .
          .
EX3       OPEN (,(RDBACK,LEAVE)),TYPE=J,MF=(E,EX1)
          .
          .
          .
INVEN     DCB      EXLST=LSTA,...
MASTER    DCB      EXLST=LSTB,...
LSTA      DS       0F
          DC       AL1(EXLLASTE+EXLRJFCB)
          DC       AL3(JFCBAREA)
          .
          .
          .
JFCBAREA  DS       0F,176C
          .
          .
          .
LSTB      DS       0F
          .
          .
          .
          IHAEXLST ,      DCB exit list mapping
```

Figure 30. Example code using the RDJFCB macro

Multiple data control block addresses and associated options can be specified in the RDJFCB macro instruction. This facility makes it possible to read several job file control blocks in parallel.

An exit list address must be provided in each DCB specified by an RDJFCB macro instruction. Each exit list must contain an active entry of either or both types supported by RDJFCB.

RDJFCB processes the first of each of the two types of its entries in the exit list. For example, in a three-entry list containing types 07, 07 and 13, RDJFCB will process the first and third entries and ignore the second entry. An ignored entry has no effect on the RDJFCB return code.

Each of the entries is briefly explained in the following text. A full discussion of the exit list and its use is contained in *z/OS DFSMS Using Data Sets*.

After RDJFCB is performed, register 15 contains one of the following codes:

Table 72. Return codes from the RDJFCB macro

Return code	Meaning
0 (X'00')	RDJFCB function completed successfully.
4 (X'04')	One or more DCBs encountered one of the following conditions and were skipped. DCBs that were not skipped were processed successfully. • The DCB was being processed by Open/Close/EOV or a similar function. • No data set with the DDNAME that is in the DCB is allocated. • The DCB is not open and its DDNAME is blank.
8 (X'08')	One or more DCBs had an ARL that could not be processed. Each ARL contains a reason code describing its status. One or more DCBs might have encountered a condition described under return code 4. This type of ARL does not contain a reason code.

Type 07 JFCB Exit List Entry

The type 07 JFCB exit list entry allows you to perform a variety of tasks, as described in the following text.

The format of the type 07 JFCB exit list entry is:

Table 73. Type 07 JFCB exit list entry

Hexadecimal code (high-order byte)	Contents of exit list entry (three low-order bytes)
07	Address of a 176-byte area required if the RDJFCB or OPEN (TYPE=J) macro instruction is used.

The virtual storage area into which the JFCB is read must be:

- Located within the user's region
- On a word boundary
- At least 176 bytes long.

Requirement: Users running in 31-bit addressing mode must ensure that this area is located below 16 MB virtual. Each exit list entry must be 4 bytes long. The system recognizes only the first occurrence of an exit list entry code. Indicate the end of the exit list by setting the high-order bit in the entry code byte to 1.

The DCB can be either open or closed when this macro instruction is executed. If accessing concatenated sequential data sets and the DCB is open, the RDJFCB routine reads the JFCB for the current data set. In all other cases, the RDJFCB routine reads the JFCB for the first or only data set.

If the RDJFCB routine fails while processing a DCB associated with your RDJFCB request, or you do not provide a virtual storage address in the three low-order bytes of the exit list entry, your task is abnormally terminated. None of the options available through the DCB ABEND exit, as described in *z/OS DFSMS Using Data Sets*, are available when a RDJFCB macro instruction is issued.

RDJFCB Security

OPEN TYPE=J compares the volume serial numbers specified in the user-supplied JFCB with the volume serial numbers in the system's copy of the JFCB. Each different volume serial number will be enqueued exclusively. The volumes stay enqueued until the job step terminates, because the CLOSE routines will not dequeue the volumes. If the job step already has the volume open, OPEN TYPE=J continues. If the volume is enqueued by another job step, an ABEND 413 occurs with a return code of X'04'.

Some JFCB modifications can compromise the security of existing password-protected or RACF-protected data sets. The following modifications are specifically not allowed, unless the program making the modifications is authorized (an authorized program is one that is either in supervisor state, executing in one of the system protection keys (keys 0 through 7), or authorized under the authorized program facility) or can supply the password:

- Changing the disposition of a password-protected data set from OLD or MOD to NEW.
- Changing the data set name or one or more of the volume serial numbers when the disposition is NEW.
- Changing the label processing specifications to bypass label processing.

Changing the data set name in the JFCB has no effect on the name of the data set. You are just referring to a different data set. If the DISP setting is NEW, it does not cause the changed data set name to be created. If OPEN allocates the changed data set name, it is as if the DD had DISP=OLD. To create a data set on DASD, your program should call dynamic allocation by issuing SVC 99 (Requesting dynamic allocation functions in *z/OS MVS Programming: Authorized Assembler Services Guide*). Alternately your program might be able to use the REALLOC macro, but note that it requires authorization and carries restrictions. See “Creating (Allocating) a DASD Data Set Using REALLOC” on page 48.

RDJFCB Use by Authorized Programs

Except when opening a VTOC, if you change the data set name in the JFCB and your job step is APF-authorized, you should do a system enqueue on the major name of "SYSDSN" for the substituted data set name. Issuing an ENQ macro for a major name of SYSDSN requires authorization.

If your program changes the data set name or the volume serial number and is not authorized, OPEN TYPE=J calls dynamic allocation for the new name. CLOSE will automatically unallocate the data set.

To use the correct interface with other system functions (for example, partial release), the ENQ macro should include the TCB of the initiator and the length of the data set name (with no trailing blanks). When you complete processing of the data set, you should use the DEQ macro to release the resources. If your program is not authorized and issues an OPEN TYPE=J macro, and the substituted data set name is already enqueued by another job step, an ABEND 913 occurs with a return code of X'1C'.

To open a VTOC data set to change its contents (that is, open it for OUTPUT, OUTIN, INOUT, UPDAT, OUTINX, or EXTEND), your program must be authorized under the Authorized Program Facility (APF). APF provides security and integrity for your data sets and programs. Details on how to authorize your program are provided in *z/OS MVS Programming: Assembler Services Guide*, *z/OS MVS Programming: Authorized Assembler Services Guide*, and *z/OS MVS Programming: Assembler Services Reference ABE-HSP*.

You cannot extend a VTOC by this means.

Restriction: Do not change the data set name to NULLFILE (signifying a dummy data set). Changing the name to NULLFILE can prevent the device allocated for the data set specified on the DD statement from being deallocated at job/step termination.

Using BSAM or EXCP for Random I/O to a Multivolume Data Set

If you open a BDAM DCB for a multivolume data set, OPEN links your program to all volumes simultaneously so that your program can ignore volume boundaries and treat all volumes of the data set as one entity. If you open a BSAM or EXCP DCB for a multivolume data set, OPEN gives your program access to only one volume at a time. This is true for both disk and tape. To switch to another volume, your program issues a CHECK or FEOV macro for BSAM or EOV macro for EXCP. To return to a previous volume, you must close and reopen the data set, which would be slow.

Your program can use RDJFCB and OPEN TYPE=J with one DCB per volume to process all the volumes in parallel. Your program must keep track of which DCB is for each volume. Your program uses the RDJFCB macro to read in the JFCB, and uses OPEN with TYPE=J to open each volume of the data set. The coding example in Figure 31 on page 280 illustrates the procedure with EXCP DCBs.

This technique does not work for a JFCB disposition of NEW because OPEN TYPE=J honors modifications to the JFCB volume sequence number JFCBVLSQ only for tape and only if the JFCB file sequence is also modified.

This technique does not work with a striped data set because OPEN always opens all volumes of a striped data set in parallel as for BDAM.

If you are using BSAM to read non-striped volumes in parallel, you should avoid using the CHECK macro because it can automatically move to the next volume when you reach the end of the current volume. Use WAIT or EVENTS instead of CHECK. Refer to *z/OS DFSMS Using Data Sets* and *z/OS DFSMS Macro Instructions for Data Sets* for information about WAIT or EVENTS. If you optimize I/O with the MULTACC parameter of the DCBE macro, you also must issue TRUNC macros.

With tape, you cannot open more than one DCB per allocated drive. You can calculate the number of allocated drives from the TIOT entry length or by issuing the IEFDDSRV macro. IEFDDSRV returns the number of devices in DVAR_NUM_DVENT. Refer to *z/OS MVS Programming: Assembler Services Reference IAR-XCT*.

If your program does any of the following it will damage the data set:

- If you use BSAM and the OPEN option is not UPDAT and you issue WRITE macros, you might cause the data set to be extended to new tracks or to the next volume. In the latter case you have two DCBs open to one volume and they can interfere with each other.
- If you use BSAM and the OPEN option is not UPDAT and the last operation before CLOSE is WRITE (and CHECK, WAIT or EVENTS), then CLOSE marks that volume as being the last volume of the data set. The result might be every volume marked as being the last one. This is true also if you use EXCP and CLOSE finds that bit 0 of DCBOFLGS is on. For EXCP, see Table 36 on page 175 and “Device-Dependent Parameters” on page 182. If a program later tries to read the volumes of the data set sequentially, such as a program backing it up, that program will not read past the end of this volume. In addition, a later program trying to add records to the end of a volume by opening with the EXTEND or OUTINX option or with the OUTPUT or OUTIN option with DISP=MOD, can add them to the wrong volume.

If the data set is newly allocated on DASD, space has been allocated only on the first volume unless you used the guaranteed space option of SMS. For both DASD and tape, if the data set has not yet been written on the volume, the OPEN fails.

```
ALLVOLS  RMODE 24             Because of DCB exit list & JFCB
              RDJFCB DCB1           Read in the JFCB
              LTR   R15,R15         Branch if the DD name
              BNZ   NODD              is not defined
     * Calculate amount of storage for one DCB per volume and get storage.
              SR    R0,R0           Prepare for IC
              IC    R0,JFCBNVOL     Get number of volumes
              LR    R3,R0           Save number of volumes
              MH    R0,=Y(DCBLNGXE) Mult by EXCP DCB length without append.
              STORAGE OBTAIN,LENGTH=(0),LOC=(BELOW,ANY),ADDR=DCBAddrL
              LR    R4,R1           Point to area for first DCB
              LA    R5,1            Set first volume sequence number
     OpenLoop STH   R5,JFCBVLSQ     Tell OPEN which volume to open
              MVC   0(DCBLNGXE,R4),DCB1   Build a DCB
              OPEN  ((R4),UPDAT),TYPE=J   Use TYPE=J for one volume
              LTR   R15,R15         Branch in the unlikely event
              BNZ   OpenFail          that OPEN failed
              LA    R4,DCBLNGXE(,R4)  Point to place for next DCB
              LA    R5,1(R5)        Increment the volume counter
              BCT   R3,OpenLoop     Loop until all volumes are open
              .
              .
              .
     DCB1     DCB   DDNAME=SYSUT1,MACRF=(E),EXLST=ExitL,DSORG=PS
     * The following is a DCB exit list.
     ExitL    DC    0F'0',AL1(EXLLASTE+EXLRJFCB)    Last entry, for JFCB
              DC    AL3(JFCB)       Address of JFCB area
     DCBAddrL DS    A               Address of DCB list
     JFCB     DS    CL176           JFCB READ IN HERE
              ORG   JFCB+70         Go back to remap
     JFCBVLSQ DS    H               Volume sequence number
              ORG   JFCB+117
     JFCBNVOL DS    FL1             Number of volumes allocated
              ORG   ,
     * Mapping macro IEFJFCBN might be used instead.
              DCBD DSORG=XE,DEVD=(DA,TA)  Map an EXCP DCB
              IHAEXLST ,      DCB exit list mapping
```

Figure 31. Processing a Multivolume Data Set with EXCP

Type 13 JFCB Exit List Entry

The type 13 JFCB exit list entry allows you to retrieve selected allocation information, as described in the following text. The system will accept both a type X'07' and a X'13' exit list entry. RDJFCB uses the first of each of them.

The format of the type 13 JFCB exit list entry is:

Table 74. Format of the type 13 JFCB exit list entry

Hexadecimal code (high-order byte)	Contents of exit list entry (three low-order bytes)
13	Address of an allocation retrieval list.

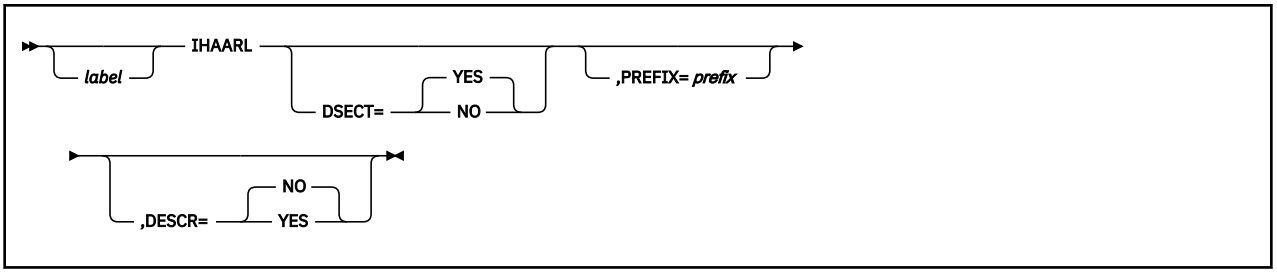
Using RDJFCB to retrieve allocation information

RDJFCB uses DCB exit list entry type 13 to retrieve allocation information (JFCBs and volume serial numbers) for data sets that might be concatenated. The exit list entry code is X'13', and is defined as “retrieve allocation information.” The second through fourth bytes of this entry must point to an allocation retrieval list (ARL), as described in Table 75 on page 281. If you issue RDJFCB, this DCB exit list entry retrieves all JFCBs for the specified concatenated data sets, and lists of all volume serial numbers for these data sets. The block size, as specified on the DD statement of each data set, is put into the extended information segment following the volume serial numbers. If this block size field is 0, the block size of the data set is in the JFCB. You can either select JFCBs in the concatenation or, by default, retrieve all of them. RDJFCB uses the parameter list to receive and return information about the request.

OPEN TYPE=J does not recognize this exit list entry.

You can use the IHAARL macro (shown here) to generate and map the ARL. Your program might issue a GETMAIN or STORAGE macro for the ARL, or, if you specify DSECT=NO, the ARL is generated within your program's storage. The ARL must be below 16 MB. The allocation retrieval area (ARA), acquired by RDJFCB through a GETMAIN macro, can be above or below 16 MB.

The format of the IHAARL macro is:



DSECT=YES or NO
Specifies whether the symbol at the beginning of the generated area appears on a DSECT instruction or a DC instruction. For DSECT=NO, the symbol appears on a DC instruction. The default is DSECT=YES.

PREFIX=*prefix*
Allows you to invoke the macro more than once per assembly. Specifies a character string with which all generated symbols are to be prefixed. Do not specify delimiters, such as quotation marks. If you omit this operand or specify a null value, the prefix defaults to the characters ARL.

DESCR=YES or NO
Specifies whether the macro expansion includes the macro description (prolog). The default is DESCR=NO.

Table 75 on page 281 and Table 76 on page 282 describe the formats of the allocation retrieval *list* and allocation retrieval *area*, respectively.

Table 75. Format of the Allocation Retrieval List (mapped by the IHAARL macro)

Offset	Bytes	Name	Description
The following fields are set by the caller of RDJFCB.			
0 (X'00')	2	ARLLEN	Length of this area. Value should be 36.
2 (X'02')	2	ARLIDENT	EBCDIC 'AR'
4 (X'04')	1	ARLOPT1	Option byte.
	0	ARLLANY	ARA must be below 16 MB.
	1...		ARA can be above 16 MB.
	.1..	ARLUSS	Request ARA have a path name for each entry for which PATH was coded.
	. .xx xxxx		Reserved. Must be zero.
5 (X'05')	7	ARLRSVD1	Reserved. Must be zero.
12 (X'0C')	2	ARLRETRV	Number of data sets for which to retrieve information. If 0, retrieve all in the concatenation.
14 (X'0E')	2	ARLFIRST	Number of first data set in concatenation for which to retrieve information. 0 or 1 specifies retrieval of information beginning with first data set in the concatenation.
The following fields are set by RDJFCB			

Table 75. Format of the Allocation Retrieval List (mapped by the IHAARL macro) (continued)

Offset	Bytes	Name	Description
16 (X'10')	4	ARLAREA	Address of ARA. See Table 76 on page 282.
20 (X'14')	1	ARLPOOL	Storage subpool containing ARA.
21 (X'15')	3	ARLRLEN	Length of ARA.
24 (X'18')	2	ARLRTRVD	Number of concatenated data sets for which JFCBs were retrieved.
26 (X'1A')	2	ARLCONC	Number of concatenated data sets. If no concatenation, this value is 1.
28 (X'1C')	1	ARLRCODE	Reason Code: 0 = Requested information was read. The following reason codes are related to return code 8: 4 = ARLFIRST is greater than ARLCONC. 8 = Insufficient storage to read information. ARLPOOL and ARLRLEN describe what RDJFCB needs.
29 (X'1D')	7	ARLRSVD2	Reserved. Used by RDJFCB.

Table 76. Format of the Allocation Retrieval Area (mapped by the IHAARA macro)

Offset	Bytes	Name	Description
0 (X'00')	2	ARALEN	Length of the information for this data set (including this field). The starting address of the ARA plus the value in the length field designates the address of the ARA for the next data set in the concatenation, if requested. Do not use the length field to calculate the number of volumes.
2 (X'02')	1	ARAFLG	Flags.
	1...	ARAXINF	Extended information segment is present.
	.xxx xxxx		Reserved. Must be zero.
3 (X'03')	1	ARAXINOF	Offset in doublewords from the beginning of the allocation retrieval area for the current data set to the extended information segment.
4 (X'04')	176(Dec)	ARAJFCB	JFCB
180 (X'B4')	variable	*	Sixth and subsequent volume serial numbers. Determined by the value in JFCBNVOL. If the number of volume serial numbers is fewer than the specified volume count, entries at the end of the list might contain all blanks. If the first byte of an entry is X'FF', the JCL-specified VOL=REF and the volume could not be determined.
Extended Information Segment. The DSECT name is ARAXINFO.			
0 (X'00')	2	ARAXINLN	Length of the extended information segment (including this field).
2 (X'02')	2	ARAPATHO	If other than zero, ARAPATHO is the offset from beginning of extended information segment to the path name (ARAPATHNAME); in this case, the data set name in the JFCB is meaningless. If zero, then this entry does not contain a path name.

Table 76. Format of the Allocation Retrieval Area (mapped by the IHAARA macro) (continued)

Offset	Bytes	Name	Description
4 (X'04')	4		Reserved. Must be 0.
8 (X'08')	8	ARAXLBKS	Block size for this data set or 0. If 0 then block size can be found in the JFCB in ARAJFCB. If JFCBLKSZ is set, this field will be returned with a value of 0.
16 (X'10')	8	ARABKSLM	The maximum allowed value for system determined block size (BLKSZLIM or DFABLKSZ value in DFA). This integer value is meaningful only if block size (BLKSIZE) is omitted from all sources and the application program opens for output. It is the first value available from these sources: 1. BLKSZLIM keyword on the DD statement or dynamic allocation. 2. Block size limit in data class. Set by storage administrator. Available even if the data set is not SMS-managed. 3. System default set in TAPEBLKSZLIM keyword in DEVSUPxx member in SYS1.PARMLIB by system programmer. Also available in DFA. 4. 32760 The minimum value in the current level of the operating system is 32760.
24 (X'16')	16	*	Reserved.
The following fields are present only if ARAPATHO is non-zero:			
		ARAPATHNAME	DSECT
x	2	ARAPATHLEN	Length of path name, excluding any trailing blanks. Calculate the value of x by adding the value in the two bytes in ARAPATHO to the address of ARAXINLN.
x+2	255	ARAPATHNAM	Path name.
	257	ARAPNAMLEN	Symbolic length of maximum path name section.

When you have finished using information from the retrieval areas, you should issue FREEMAIN or STORAGE macro to free any areas that were acquired through GETMAIN (including the ARA, acquired by RDJFCB). When coding the FREEMAIN or STORAGE macro, specify the *length*, *subpool*, and *address* values from the ARLRLEN, ARLPOOL, and ARLAREA fields, respectively. The DSECT name is ARAXINFO.

Code the FREEMAIN macro as shown:

```
FREEMAIN RU,LV=length,SP=subpool,A=address
```

If RDJFCB successfully fills in the ARL field, register 15 is set to zero. Otherwise see Table 72 on page 277.

Sample code that retrieves allocation information

In the following code sample, the macro instruction at ALLOCINF creates a parameter list for one DCB (INDCB), assumed to be open for input. The JFCBs and volume serial numbers are retrieved for all data sets allocated to DDNAME SYSLIB.

```
    ***JCL FOR FOLLOWING INVOCATION OF RDJFCB:
     //SYSLIB DD    DISP=SHR,DSN=DEPT61.MACLIB
     //       DD    DISP=SHR,DSN=CORPORAT.MACLIB
```

```
//       DD    PATH='/projects/sasp/maclib',PATHOPTS=ORDONLY
//       DD    DISP=SHR,DSN=SYS1.MACLIB
  ***EXAMPLE CODE TO INVOKE RDJFCB ALLOCATION INFORMATION RETRIEVAL:
*   GET A COPY OF THE JFCB FOR THE FIRST OR ONLY DATA SET ALLOCATED
*   TO SYSLIB AND TRY TO READ THE JFCBS VOLUME SERIAL NUMBERS
*   AND PATH NAMES FOR ALL DATA SETS ALLOCATED TO SYSLIB.
*
ALLOCINF RDJFCB (INDCB)
         LTR   R15,R15          TEST RDJFCB RETURN CODE
         BNZ   NOJFCB           BRANCH IF INFORMATION NOT AVAILABLE
         ICM   R1,X'F',SLBAREA  GET AND TEST ADDRESS OF ARL
         BZ    OLDSYSTM         GO IF SYSTEM DOES NOT SUPPORT ARL
         CLI   SLBRCODE,0       TEST RDJFCB REASON CODE
         BNE   NOJFCB           BRANCH IF INFORMATION NOT AVAILABLE
*
*  LOOP THROUGH THE JFCBS IN THE AREA TO WHICH SLBAREA POINTS.
*  CODE CAN BE INSERTED HERE TO PRINT THE DATA SET NAMES, VOLUME SERIAL NUMBERS
*  AND PATH NAMES.
         LH    R9,SLBRTRVD      GET NUMBER OF JFCB'S RETRIEVED
         L     R2,SLBAREA       POINT TO ARA
         USING ARA,R2
LOOPARA  TM    ARAFLG,ARAXINF   BRANCH IF NO EXTENDED
         BZ    USEJFCB          INFORMATION SEGMENT
         SR    R3,3             PREPARE FOR IC
         IC    R3,ARAXINOF      GET DOUBLEWORD OFFSET
         SLL   R3,3             GET BYTE OFFSET
         AR    R3,R2            POINT TO EXTENDED INFO SEGMENT
         USING ARAXINLN,R3      EXTENDED INFORMATION SEGMENT
         SR    R4,R4            PREPARE FOR ICM
         ICM   R4,B'0011',ARAPATHO BRANCH IF NO PATH
         BZ    USEJFCB          NAME
         USING ARAPATHNAME,R4
*  PRINT PATH NAME
          .
          .
         B     NEXTARA
*  PRINT DATA SET NAME IN JFCB.
USEJFCB   ...
          .
          .
NEXTARA  AH    R2,ARALEN        POINT TO NEXT ARA ENTRY
         BCT   R9,LOOPARA       DECREMENT JFCB COUNTER, LOOP IF MORE
          .
          .
         SR    R2,R2
         IC    R2,SLBPOOL
         SR    R0,R0
         ICM   R0,B'0111',SLBRLEN
         FREEMAIN RU,LV=(0),SP=(R2),A=SLBAREA
         .
         .
OLDSYSTM DS    0H               ROUTINE TO HANDLE JUST LIBJFCB
         .
         .
*
NOJFCB   DS    0H               ROUTINE TO HANDLE INABILITY TO GET THE
*                               JFCB. THE DATA SET MAY NOT BE ALLOCATED.
         .
         .
*


SLBOPNX  DS    0H               DCB OPEN EXIT ROUTINE FOR SYSLIB.
*                               HANDLES RECFM, LRECL, AND BLKSIZE.
         .
         .
INDCB    DCB   DSORG=PO,DDNAME=SYSLIB,MACRF=R,SYNAD=INERROR,           X
               EXLST=INEXLST
INEXLST  DC    0F'0',AL1(EXLDCBEX)    ENTRY CODE FOR OPEN EXIT ROUTINE
         DC    AL3(SLBOPNX)           ADDR OF DCB OPEN EXIT ROUTINE
         DC    AL1(EXLARL)            ENTRY CODE TO RETRIEVE
*                                     ALLOCATION INFORMATION
```

```
            DC    AL3(SLBSTRT)           ADDR OF ALLOCATION RETRIEVAL LIST
            DC    AL1(EXLLASTE+EXLRJFCB) ENTRY CODE TO RETRIEVE FIRST JFCB
*                                        AND INDICATE LAST ENTRY IN LIST
            DC    AL3(LIBJFCB)           ADDR OF JFCB FOR FIRST DATA SET
*
*  AN ALLOCATION RETRIEVAL LIST FOLLOWS, POINTED TO BY DCB EXIT LIST.
*
SLBSTRT  IHAARL DSECT=NO,PREFIX=SLB
            DC    0F'0'
LIBJFCB  DC    CL176' '         FIRST JFCB
            IHAARA ,
            IHAEXLST ,             DCB exit list mapping
```

DEQ at Demount Facility for Tape Volumes

This facility is intended to be used by long-running programs that create an indefinitely long tape data set (such as a log tape). Use of this facility by such a program permits the processed volumes to be allocated to another job for processing (such as data reduction). This processing is otherwise prohibited unless the indefinitely long data set is closed and dynamically deallocated.

You can invoke this facility only through the RDJFCB/OPEN TYPE=J interface by setting bit JFCDQDSP (bit 0) in field JFCBFLG3 at offset 163 (X'A3') to 1. The volume serial of the tape is dequeued when the volume is demounted by OPEN or EOV with message IEC502E when all the following conditions are present:

- The tape volume is verified (where a tape is considered *verified* after file protect, label type, and density conflicts have been resolved) for use by OPEN or EOV (see page "DEQ at Demount Facility for Tape Volumes" on page 285 for more information concerning *verified*).
- JFCDQDSP is set to 1.
- The program is APF authorized (protect key and supervisor/problem state are not relevant).
- The tape volume is to be immediately processed for output. That is, either OPEN verifies the volume and the OPEN option is OUTPUT, OUTIN, or OUTINX; or EOV verifies the volume and the DCB is opened for OUTPUT, OUTIN, INOUT, or EXTEND, and the last operation against the data set was an output operation (DCBOFLWR is set to 1).

For EOV to find JFCDQDSP set to 1, the program must not inhibit the rewrite of the JFCB by setting bit 4 of JFCBTSDM to 1.

The tape volume is considered verified after file protect, label type, and density conflicts have been resolved. The volume is dequeued when demounted after this verification, even if further into OPEN or EOV processing the volume is rejected because of expiration date, security protection, checkpoint data set protection, or an I/O error.

When the volume serial is dequeued, the volume becomes available for allocation to another job. However, because the volume DEQ is performed without deallocating the volume, care must be exercised both by the authorized program and the installation to prevent misuse of the DEQ at demount facility. A discussion of such misuse follows:

- The authorized program must not close and reopen the data set using the tape volume DEQ at demount facility, if it does, one of the following can occur:
  - The dequeued volume can be mounted and in use by another job. When the volume is requested for mounting, for the authorized program, the operator is unable to satisfy the mount. Therefore, the operator must either cancel the requesting job, cancel the job using the volume, wait for the requesting job to time out, or wait for the job using the volume to terminate.
  - The dequeued volume can be allocated to another job, but not yet in use. The operator mounts the volume to satisfy the mount request of the authorized job. When the volume is requested for mounting by the other job, the operator is unable to satisfy the mount request, and is faced with the same choices as in the previous item.

  - The dequeued volume can not yet be allocated to another job and the volume is mounted to satisfy the mount request of the authorized job. Another job can allocate the volume and, when the volume is requested for mounting, the situation is the same as in the previous item.

  It is the responsibility of the installation that permits a program to run with APF authorization to ensure that it does not close and reopen a data set using the DEQ at demount facility.

- Care should be exercised when an authorized program uses the DEQ at demount facility (data set 1), but processes another tape data set (data set 2). Assume the same volume serial numbers have been coded in the DD statements for data set 1 and data set 2. As the volumes of data set 1 are demounted, they are dequeued even though those volumes still might be requested for data set 2. All the problems explained in the preceding list can occur as data set 2 and another job contend for a dequeued volume.

  This problem should not occur, given the intended use of the DEQ at demount facility; that is, a long-running application creating an indefinitely long tape data set. This type of application is not normally invoked through batch execution with user-written DD statements.
- After a volume has been demounted and dequeued because of the DEQ at demount facility, the volume is not automatically rejected by the control program when mounted in response to a specific or nonspecific mount request. Without the use of the facility, the control program can recognize (by the ENQ) that the volume is in use, and reject the volume. Therefore, operations procedures, in effect to prevent incorrect volumes from being mounted, should be reviewed in the light of reduced control program protection from such errors when the DEQ at demount facility is used. Specifically, if a volume is remounted for an authorized program and the volume had been used previously by that authorized program, duplicate volume serial numbers will exist in the JFCB, and the control program will be unable to release the volume during EOV processing.
- Checkpoint/restart considerations are discussed in *z/OS DFSMSdfp Checkpoint/Restart*.

High-Speed Cartridge Tape Positioning

High-speed positioning for cartridge tape is available when opening a tape data set on an IBM standard-labeled tape for either EXTEND (OUTINX, EXTEND, or DISP=MOD). High-speed positioning is also available when opening to the beginning of such a data set. To invoke high-speed positioning, your program must modify certain fields in the JFCB and use OPEN TYPE=J to open the data set. When you write or read on an IBM 3590 Model A60 at the right hardware level it is not important to use the procedure described here. The magnetic tape subsystem gives these performance benefits automatically. This procedure will not degrade performance.

Tip: On an IBM 3480, IBM 3490, or older models of IBM 3590, this technique offers significantly better performance than the technique for setting the data set sequence number. In addition, systems with DFSMSrmm use this faster technique automatically for all cartridge tapes. For the IBM 3590 Model A60, both techniques give high performance.

Use the following procedure to modify the JFCB:

1. Issue the RDJFCB macro to have the system move the JFCB into your work area.
2. Set the JFCPOSID flag in the JFCBFLG3 flag byte to indicate that you are providing a block ID for a high-speed search.
3. Move the block ID into the JFCRBIDO field of the JFCB. If you are opening to the beginning of a data set, use the block ID of the first header label record of that data set. If you are opening to the end of a data set (for example, to extend it), use the block ID of the tape mark immediately following the last block of user data in that data set.
4. Issue the OPEN TYPE=J macro to have the system use your modified JFCB.

After the tape is positioned, OPEN processes the trailer labels for the data set being extended.

If you set the JFCPOSID flag off, OPEN positions the volume normally, as though the high-speed positioning feature were not active.

If you set the JFCPOSID flag on, but do not provide a block ID in the JFCRBIDO field, OPEN positions the volume normally and does one of the following:

- If you are opening to the beginning of a data set, OPEN inserts the block ID of the first header label record of that data set into the JFCRBIDO field.
- If you are opening to the end of the data set, OPEN inserts the block ID of the tape mark immediately following the last block of user data for that data set into the JFCRBIDO field.

OPEN does not update your copy of the JFCB. To retrieve the new value in the system's copy of the JFCB, issue RDJFCB after OPEN.

If the JFCPOSID flag is on during CLOSE processing, (because you set it on before OPEN), CLOSE inserts the block ID for the first header label record of the next data set (which might not exist) into the JFCRBIDC field. Therefore, if you unallocate the cartridge tape device and want to use the current block ID for subsequent processing, save the block ID before you close the data set.

OPEN resets the JFCPOSID flag if any one of the following conditions exists:

- Your program issues an OPEN which is not TYPE=J
- The requested tape volume is not an IBM standard-labeled volume
- The requested unit is not a buffered tape device.

Exceptions:

1. If you specify dynamic deallocation (with SVC99, FREE=CLOSE on the DD statement, or the FREE option on the CLOSE macro), the block ID for the next data set will not be available to your program.
2. When using high-speed positioning, specify the data set sequence number normally, either explicitly by LABEL=(seqno,SL) on the DD statement, or by default.

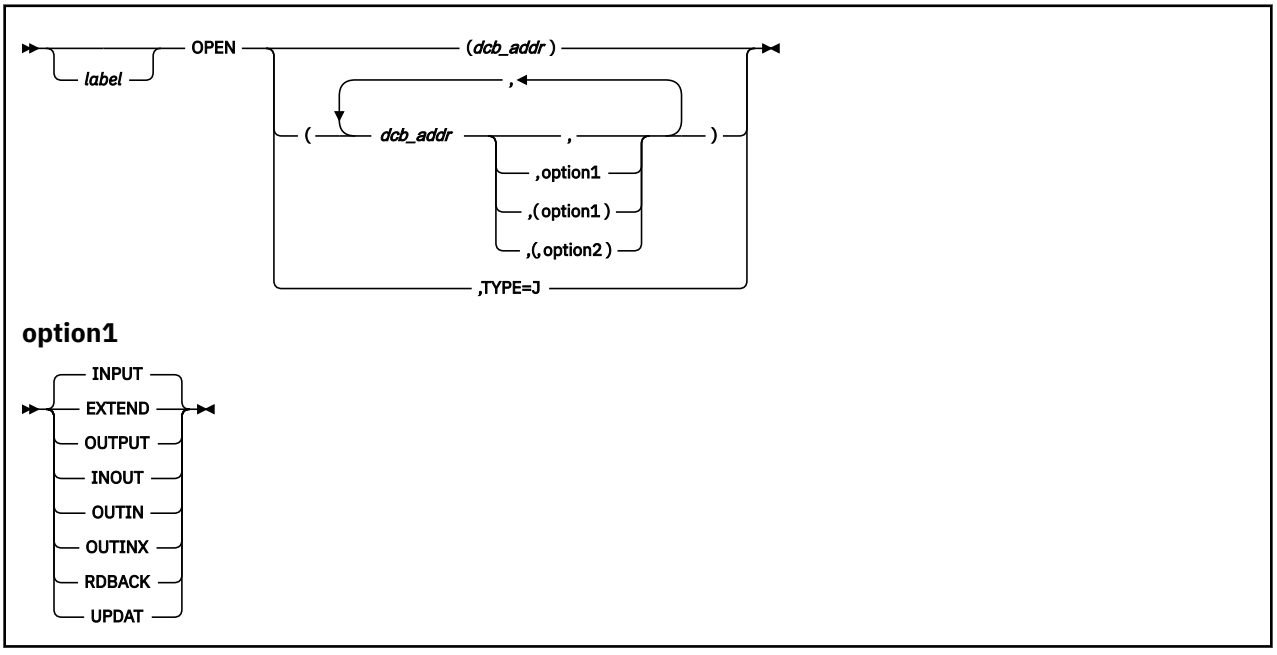
After the system routines have used the JFCRBIDO field for high-speed positioning, they clear JFCRBIDO in the system's copy of the JFCB to prevent misinterpretation during a subsequent OPEN.

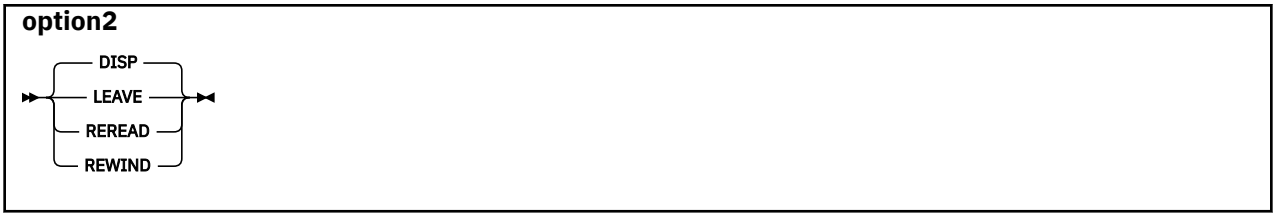
OPEN - Initialize Data Control Block for Processing the JFCB

The OPEN macro instruction initializes one or more data control blocks (DCBs) so that their associated data sets can be processed.

A full explanation of the operands of the OPEN macro instruction is contained in the publication *z/OS DFSMS Macro Instructions for Data Sets*. The TYPE=J option, because it is used in conjunction with modifying a JFCB, should be used only by the system programmer or under the system programmer's supervision.

The format of the OPEN TYPE=J macro is:





Tip: To have multiple DCBs with or without options, code each DCB (and options) as shown in the diagram and precede each additional DCB with a comma.

```
OPEN (DCB1),TYPE=J
OPEN (DCB1,INPUT),TYPE=J
OPEN (DCB1,(INPUT)),TYPE=J
OPEN (DCB1,(INPUT,REREAD)),TYPE=J
OPEN (DCB1,,DCB2),TYPE=J
OPEN (DCB1,,DCB2,(INPUT,REREAD),DCB3,INPUT),TYPE=J
```

Figure 32. Examples of standard form of the OPEN TYPE=J Macro

TYPE=J

Specifies that, for each DCB referred to, you have supplied a job file control block (JFCB) to be used during initialization. A JFCB is an internal representation of information in a DD statement.

During initialization of a data control block, its associated JFCB can be modified with information from the DCB or an existing data set label or with system control information.

When the TYPE=J operand is specified, also supply a DD statement. However, the amount of information that is given in the DD statement is at your discretion, because you can modify many fields of the system-created JFCB. If you specify DUMMY on your DD statement, the open routine ignores the JFCB DSNAME and opens the data set as dummy. (See Figure 30 on page 276 for an example of coding that modifies a system-created JFCB.) The DD statement must specify at least the device allocation (see *z/OS MVS JCL User's Guide* for methods of preventing share status) and a ddname corresponding to the associated DCB DCBDDNAM field.

The MODE operand is not shown here because it is not allowed with the TYPE=J operand of the OPEN macro instruction.

Since OPEN with TYPE=J does not accept a JFCBX from the caller, you cannot change volume serials after the first five volumes.

OPEN TYPE=J will not change the volume attributes (PRIVATE, PUBLIC, or STORAGE) which are assigned to the volume during allocation. For example, if a volume status of PRIVATE is needed but allocation is going to assign a status of PUBLIC, then VOL=PRIVATE should be specified on the DD statement.

Purging and restoring I/O requests (PURGE and RESTORE macros)

The system's purge routines perform either a halt or a quiesce operation. In a halt operation, the purge routines stop the processing of specified I/O requests initiated with an EXCP or EXCPVR macro instruction. In a quiesce operation, the purge routines includes the following procedures:

- Allow the completion of I/O requests (initiated with an EXCP or EXCPVR macro instruction) that were passed to the system for execution and are executing
- Stop the processing of requests that have not yet been initiated or passed to the system, but save the IOBs of the requests so they can be reprocessed (restored) later.

The system's restore routines make it possible to reprocess I/O requests that are quiesced.

Restriction: Purge and restore processing performed for I/O requests that are not initiated by an EXCP or EXCPVR macro is not covered here. User applications that use the PURGE and RESTORE macros with the sequential access method (SAM) against partitioned data sets (PDSs) (for example, to synchronize the

I/O) cannot do so against PDSEs, sequential extended format data sets, or z/OS UNIX files, because SAM does not use EXCP or EXCPVR to access these types of data.

To pass control to the purge and restore routines, build a parameter list and place its address in register 1, then issue the macro instruction.

You can use 24-bit or 31-bit addressing mode for the PURGE or RESTORE macro (and the parameter list).

PURGE macro specification

The PURGE macro is used to halt or finish I/O requests.

Refer to “General-Use Mapping Macros” in *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN* for information about using the 31-bit interface provided by the PURGE function.

The format of the PURGE macro is:

***parameter_list_address*—RX-type address, (2-12) or (1)**

Address of a parameter list that is built on a word boundary in storage. The parameter list is 12, 16, or 32 bytes long. If the X'08' bit in the first byte is 0, the parameter list is 12 or 16 bytes. These forms are described in this section and the name of the mapping macro is IECDPPL. If the X'08' bit in the first byte is 1, the parameter list is 32 bytes and must contain the EBCDIC string "NPPL" in the last word. This form is mapped by the IOSDNPPL macro. For more information about all of these forms, see the IOSDNPPL section in *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

The parameter list address can be specified as an RX-type address or in registers 1 through 12.

The format and contents of the parameter list are as follows:

Byte
Contents

0

A byte that specifies the actions of the purge routines. The bit settings and their meanings are:

1... ....
Purge I/O requests to a single data set. The setting of this bit only takes effect if bit 2 of byte 12 is 0 and bit 6 of byte 0 is 0.

0... ....
Either purge I/O requests associated with a TCB or address space, or purge I/O requests to more than one data set. If bit 2 of byte 12 is 1, then the request is to purge I/O associated with an address space. If bit 2 of byte 12 is 0 and bit 6 of byte 0 is 1, then the request is to purge I/O associated with a TCB. If bit 2 of byte 12 is 0 and bit 6 of byte 0 is 0, then the request is to purge I/O to more than one data set.

.1.. ....
Post ECBs associated with purged I/O requests.

..1. ....
Halt I/O-request processing. (Quiesce I/O-request processing, if 0.)

...1 ....
Purge related requests. (Only valid if a data-set purge is requested.)

.... 0...
Reserved - must be zero.

.... .1..
Do not purge the TCB request-block chain of asynchronously scheduled processing.

.... ..1.
Purge I/O requests associated with a TCB. The setting of this bit takes effect if bit 2 of byte 12 is 0.

....1
This is a 16-byte parameter list. Additional purge options are specified in bytes 12 to 15. (If this bit is off, the list is 12 bytes long, and the purge routines do not put a return code in byte 4 of this list or in register 15.)

1,2,3
The address of a DEB when purging I/O requests to a single data set. The address of the first DEB in a chain of DEBs when purging I/O requests to more than one data set. The DEBDEBB field in each DEB points to the next DEB in the chain; the field in the last DEB contains zeros.

4
A byte of zeros. (If bit 7 of byte 0 is on, the purge routines put a code in byte: X'7F' when the purge operation is successful; X'40' when it is not successful. If bit 7 of byte 0 is off, then X'7F' appears in this byte.)

5,6,7
If you turned on bit 6 of byte 0, the address of the TCB associated with the I/O requests you want purged. Is zeros, if the TCB is the one you are running under.

8
Value of X'00' or X'02' means that EXCP is the owner.

9,10,11
The address of a word in storage or the address of the DEBUSPRG field (that is X'11' bytes more than the DEB address in this parameter list). At the address you specify, the purge routines store a pointer to the purged I/O restore list, that in turn contains a pointer to the first IOB in the chain of IOBs. The location of the pointer and format of the chain are shown in Figure 33 on page 291.

Note: This field is only relevant for quiesce options.

12
A byte that allows you to specify additional purge options. The bit settings and their meanings are:

Note: The following applies only if bit 7 of byte 0 is set to 1.

..1. ....
Purge I/O requests associated with an address space. (Your program must be in supervisor state.) The setting of this bit takes effect regardless of the setting of bit 6 of byte 0 and bit 0 of byte 0.

...1 ....
If this is a data-set purge, check the validity of all the DEBs associated with the purge operation. Validate this parameter list, whatever the type of purge operation, by ensuring that there are no inconsistencies in the selection of purge options. (If your program is in problem state, these actions are taken regardless of the bit setting.)

.... 1...
Ensure that I/O requests are reprocessed (restored) under their original TCB. (If zero, and bit 7 of byte 0 is on, the I/O requests are reprocessed under the TCB of the program making the restore request.)

.... .0..
Must be zero.

13
A byte of zeros.

14,15
If bit 2 of byte 12 is on, the 2-byte ID of the address space associated with the I/O requests you want purged.

Control is returned to your program at the instruction following the PURGE macro instruction.

Return Codes from PURGE

If the purge operation was successful, register 15 contains zeros. Otherwise, register 15 contains one of the following return codes:

Return Code	Meaning
4 (X'04')	Your request to purge I/O requests associated with a given TCB was not honored because that TCB did not point to the job step TCB, while the requester is in problem state.
8 (X'08')	Either you requested an address-space purge operation, but were not in supervisor state, or you requested a data set purge operation, but failed to supply a DEB address in bytes 1, 2, and 3 of the purge parameter list.
20 (X'14')	Another purge request has preempted your request. You might want to reissue your purge request in a time-controlled loop.

Exception: If you set bit 7 in byte 0 of the parameter list to zero, register 15 will contain zeros, regardless of the outcome of the purge operation.

Modifying the IOB Chain

This procedure is not recommended. However, to change the order in which purged I/O requests are restored or prevent a purged request from being restored, you can change the sequence of IOBs in the IOB chain or remove an IOB from the chain. The address of the IOB chain can be obtained from the purge I/O restore list (see Figure 33 on page 291).(The address of the purge I/O restore list is shown at bytes 9 through 11 of the purge parameter list.) Note that some IOBs could be in a different protection key.

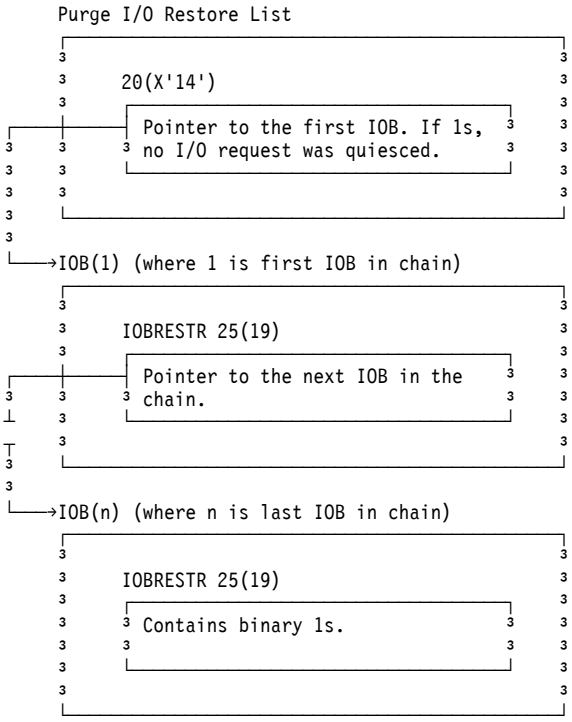


Figure 33. The IOB chain

RESTORE Macro Specification

The RESTORE macro is used to reprocess I/O requests.

The format of the RESTORE macro is:

***restore_address*—RX-type address, (2-12) or (1)**
Address that you specified at byte 9 of the purge parameter list. See "PURGE macro specification" on page 289 for information about byte 9.

Performing Track Calculations (TRKCALC macro)

The TRKCALC macro performs DASD track capacity calculations. This macro is intended for EXCP applications and other advanced applications. You can use TRKCALC to determine:

- The number of equal-length records that can be written on a track
- The total track capacity
- Whether a record can be written in the space remaining on a track and return the new track balance
- What the track balance would be if the last record were removed from a track
- The length of the longest possible record that can be written to a track.

The TRKCALC routine issues no SVC instructions or I/O. TRKCALC can be called in an SRB routine or in TCB mode. It can be called in 24-bit or 31-bit addressing mode and in supervisor or problem state.

TRKCALC works equally well for any track. It does not use the address of the track.

Using TRKCALC

This information provides an overview of how to use TRKCALC to accomplish various tasks. See "TRKCALC Macro Specification" on page 293 for details on how to code the TRKCALC parameters and on how output is returned.

Determining the number of equal-length records that can be written on a track

To determine the number of equal-length records that can be written on a track, code TRKCALC with FUNCTN=TRKCAP. You must specify the number of existing records on the track and the key and data length of the new records (using either the R, K, and DD keywords or the R, K, and DD bytes in the RKDD parameter).

If you wish to regard the track as being empty, specify an R value of 1. Otherwise, specify an R value that is one greater than the number of existing records on the track.

If the length of any existing record differs from the length of the new records (as specified in the DD value), then code the BALANCE parameter. Otherwise, omit the BALANCE parameter.

Determining the total track capacity

To determine the total track capacity, code

```
FUNCTN=TRKBAL,REMOVE=YES
```

and either R=1 or the R byte in RKDD set to 1.

Note: This value is useful only as input for the BALANCE parameter on later calls to TRKCALC to represent an empty track. You cannot write a record of this size.

Determine whether a record can be written in the space remaining on a track and return the new track balance

To determine whether a record can be written in the space remaining on the track and return the new track balance, code

```
FUNCTN=TRKBAL,REMOVE=NO
```

and the BALANCE parameter. You can supply this new track balance with the BALANCE parameter on a later call to TRKCALC.

Determine the track balance if the last record were removed from a track

To determine what the track balance would be if the last record were removed from the track, code FUNCTN=TRKBAL and REMOVE=YES. Use the R, K and DD parameters or the RKDD parameter to identify the record to be removed. It must be the last record on the track.

Determine the length of the longest possible record that can be written on a track

To determine the length of the longest possible record that can be written on a track, code FUNCTN=TRKBAL, REMOVE=NO, MAXSIZE=YES. You must specify the number of existing records on the track (using either the R keyword or the RKDD parameter) and a data length of X'FFFF' (using either the DD keyword or the RKDD parameter). The DD value of X'FFFF' is greater than is supported on any disk. Expect a return code 8, which means that the record does not fit and TRKCALC returned the size of the largest possible record.

If you wish to regard the track as being empty, specify an R value of 1.

If the track is not empty, specify an R value that is one greater than the number of existing records on the track and code the BALANCE parameter. In this case, TRKCALC will give return code 8 and the length of the longest possible record that will fit on the rest of the track.

Note: The value returned might be larger than what is supported by any access method other than EXCP.

Restrictions

Non-EXCP user applications cannot expect consistent information from TRKCALC for PDSEs, because of the unique structure and format of PDSEs. However, processing will complete without error indications.

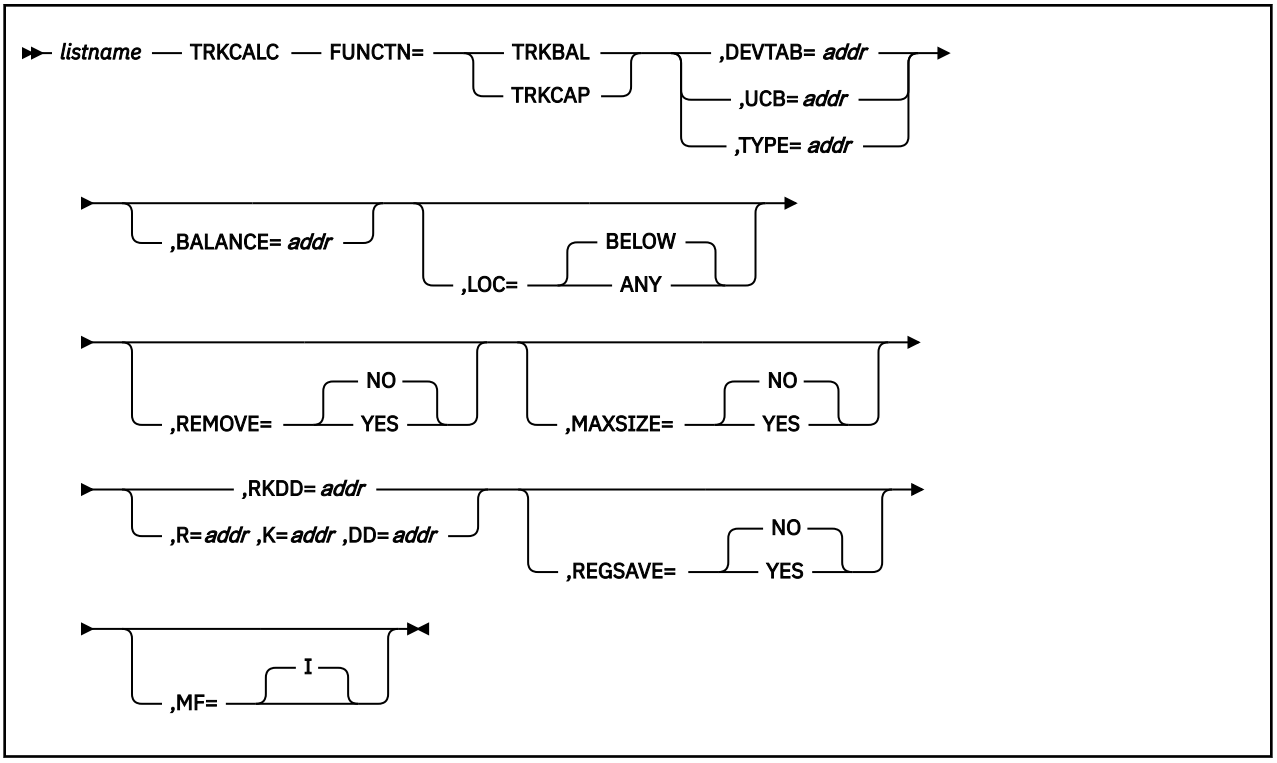
TRKCALC does not support z/OS UNIX files. You will receive unpredictable results if you use TRKCALC for z/OS UNIX files.

TRKCALC Macro Specification

The standard, list, execute, and DSECT forms of the macro are described. Examples of the TRKCALC macro follow the macro descriptions.

TRKCALC—Standard Form

The format of the TRKCALC macro is:



FUNCTN=TRKBAL or TRKCAP

Specifies the function to be performed. Specify one of the three keywords, DEVTAB, UCB, or TYPE, to provide the information source for the macro.

TRKBAL

If REMOVE=NO is specified, TRKBAL calculates whether an additional record fits on the track and what the new track balance would be if the record were added. If REMOVE=YES is specified, TRKBAL calculates what the track balance would be if a record were removed from the track. The record to be added or removed from the track is defined by the RKDD parameter, or by the R, K, and DD parameters.

If R is equal to 1 (or the R value in the RKDD parameter is 1) and REMOVE=NO is specified, TRKCALC will treat record 1 as if it were being added to an empty track; if R is equal to 1 and REMOVE=YES is specified, TRKCAL will treat record 1 as if it were being deleted from the track, leaving an empty track.

If R is not equal to 1, the specified record is added to or removed from the track. If the input track balance is not supplied through the BALANCE parameter, it is assumed that the track contains equal-sized records as specified in the RKDD parameter (or in the R, K, and DD parameters).

When REMOVE=NO is specified, one of the following occurs:

- If the record fits on the track, register 0 contains the new track balance.
- If the record does not fit on the track and MAXSIZE=NO is specified, a record does not fit return code is placed in register 15.
- If the record does not fit and MAXSIZE=YES is specified, one of the following happens:
  - The data length of the largest record that fits in the remaining space is returned in register 0.
  - A code is returned that indicates no record fits in the remaining space.

When REMOVE=YES is specified, one of the following occurs:

- If R is equal to 1, register 0 contains the track capacity.
- If R is not equal to 1, registers 0 contains the input track balance (supplied through the BALANCE parameter) incremented by the track balance used by the input record. If the input

balance is not supplied, register 0 contains the track capacity left after R–1 records are written on the track.

TRKCAP
Calculates, and returns in register 0, the number of fixed-length records that can be written on a whole track (R is equal to 1) or on a partially-filled track (R is not equal to 1). The records are defined by the K and DD values of the RKDD parameter, or by the K and DD parameters.

Depending on the value for R, one of the following occurs:

- If R is equal to 1, TRKCALC ignores the BALANCE parameter and makes the calculation as if the track were empty.
- If R is not equal to 1 and the BALANCE parameter is omitted, the calculation is made for a track that already contains R–1 records of the length defined by the K and DD values.
- If R is not equal to 1 and the BALANCE parameter is supplied, the calculation is made for a track whose remaining track balance is the value of the BALANCE parameter.

DEVTAB=*addr*—RX-type address, (2-12), (0), (14)
addr specifies a word that contains the address of the device characteristics table entry (DCTE). If you specify a register, it contains the actual address of the DCTE. The address of the DCTE can be found in the word beginning at the DCBDVTBL field of an opened DCB.

UCB=*addr*—RX-type address, (2-12), (0), (14)
addr specifies the address of a word that contains the address of the UCB. If you specify a register, it contains the actual address of the UCB.

The TRKCALC macro accepts the address of a UCB or UCB copy. Unauthorized programs can get a copy of the UCB by using the UCBSCAN macro and specifying the COPY and UCBAREA keywords. See *z/OS HCD Planning* for more information.

TYPE=*addr*—RX-type address, (2-12), (0), (14)
You can specify the address of the UCB device type (UCBTBYT4), or you can specify the 1-byte UCB device type in the low-order byte of a register.

LOC=BELOW or ANY
Optional parameter indicating whether the value passed by the UCB parameter is a 4-byte or a 3-byte address. This parameter only applies to callers running in AMODE 31. If the caller is running in AMODE 24, this parameter is ignored and the high-order byte is treated as X'00'.

BELOW
The UCB parameter contains a UCB address for a UCB which resides in storage below 16 megabytes, or a captured UCB. This is the default.

If LOC=BELOW is specified, the high-order byte of the UCB address will be treated as X'00'.

ANY
The address passed in the UCB parameter contains a 3-byte or 4-byte UCB address.

If LOC=ANY is specified when invoking in 31-bit mode, TRKCALC will treat the UCB address as a 31-bit address.

BALANCE=*addr*—RX-type address, (2-12), (0), (14)
You can specify either the address of a halfword containing the current track balance, or you can specify the balance in the low-order two bytes of a register. The value supplied could be the value returned when you last issued TRKCALC. If R is equal to 1, the balance is reset to track capacity by TRKCALC, and your supplied value is ignored. This is an input value and is not modified by the TRKCALC macro. The resulting track balance is returned in register 0 and in the TRKCALC parameter list field STARBAL. The value you supply for this parameter must be a valid value for the device type in use.

REMOVE=YES or NO
Indicates if a record is to be deleted from the track.

YES
Specifies that the record identified by the record number (specified in the R keyword) is being deleted from the track. The track balance is incremented instead of decremented.

YES is valid only on a FUNCTN=TRKBAL call.

NO
Specifies that a record is not to be deleted from the track. NO is the default.

MAXSIZE=YES or NO

YES
If the specified record does not fit, the largest length of a record with the specified key length that fits is returned (register 0).

YES is valid only on a FUNCTN=TRKBAL call.

NO
Maximum size is not returned. NO is the default.

RKDD=*addr*—RX-type address, (2-12), (0), (14)
addr specifies a word containing a record number (1 byte), key length (1 byte), and data length (2 bytes) (bytes 0, 1, and 2 and 3, respectively) or a register containing the record number, key length, and data length. R, K, and DD can be specified by this keyword, or you can use the following three keywords instead.

R=*addr*—RX-type address, (2-12), (0), (14), or n
You can specify either the address of the record number, or you can specify the record number using the low-order byte of a register or immediate data (n). Specify a decimal digit for n (immediate data).

K=*addr*—RX-type address, (2-12), (0), (14), or n
You can specify either the address of a field containing the hexadecimal value of the record's key length, or you can specify the record's key length using the low-order byte of a register or immediate data (n). Specify a decimal digit for n (immediate data).

DD=*addr*—RX-type address, (2-12), (0), (14), or n
You can specify either the address of a field containing the hexadecimal value of the record's data length, or you can specify the record's data length using the low-order two bytes of a register or immediate data (n). Specify a decimal digit for n (immediate data).

REGSAVE=YES or NO
Specifies whether registers are to be saved.

YES
Specifies registers 1 through 14 are saved and restored in the caller-provided save area (pointed to by register 13) across the TRKCALC call. Otherwise, registers 1, 9, 10, 11, and 14 are modified. Registers 0 and 15 are always modified by a TRKCALC call.

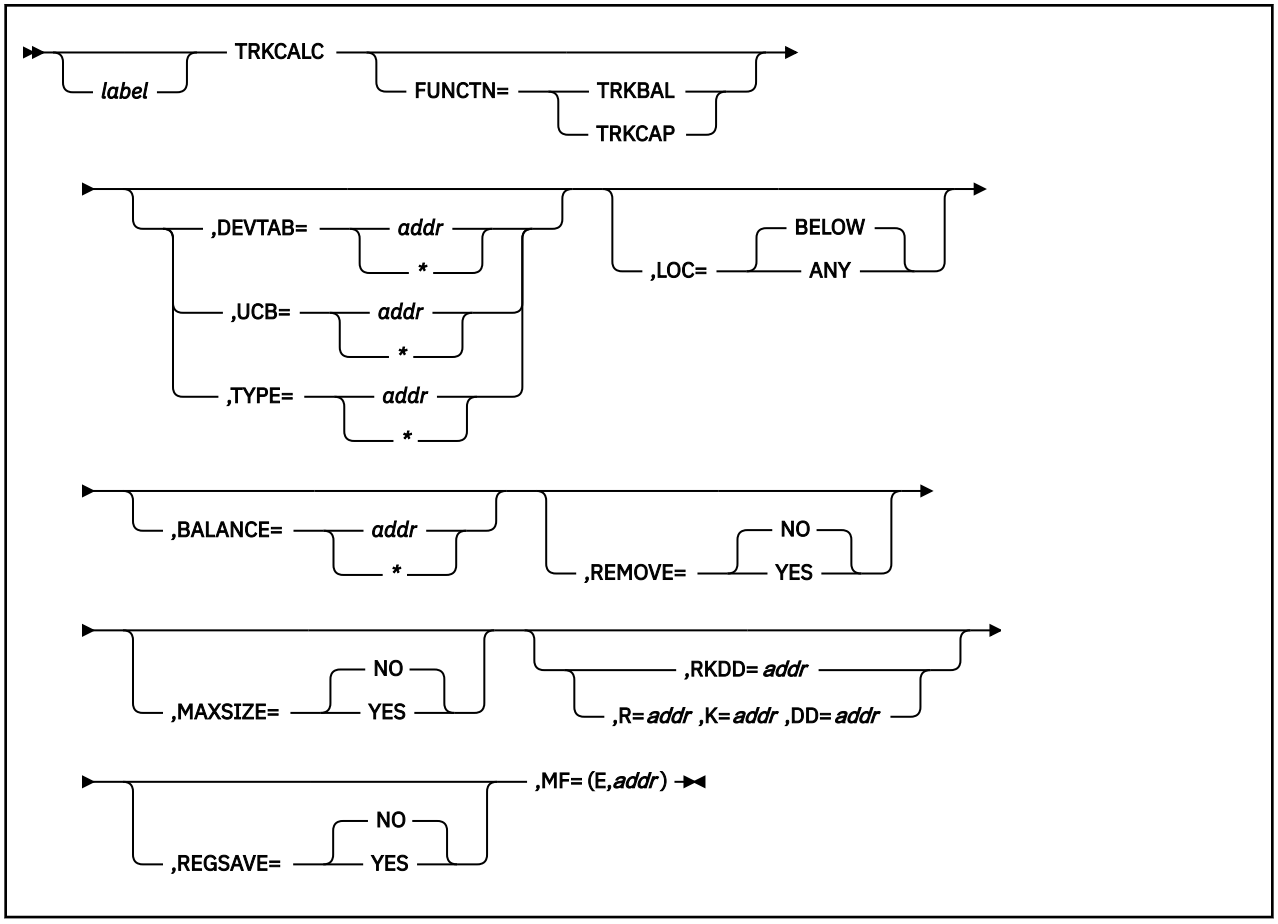
NO
Specifies registers are not saved across a TRKCALC call. NO is the default.

MF=I
Specifies storage definition for the TRKCALC parameter list and parameter list initialization, using the given keywords, then calling the TRKCALC function. MF=I is the default.

TRKCALC—Execute Form

A remote parameter list is referred to and can be modified by the execute form of the TRKCALC macro. The TRKCALC routine is called. The function of the operands is the same as for the standard form.

The format of the execute form of the TRKCALC macro is:



FUNCTN=TRKBAL or TRKCAP
Is coded as shown in the standard form. If this keyword is omitted, any specification of REMOVE, MAXSIZE, LAST, and the RX form of BALANCE is ignored. In addition, DEVTAB is assumed if UCB is coded and a failure occurs, or if TYPE is specified. When you use FUNCTN, one of the keywords (DEVTAB, UCB, or TYPE) must be specified to provide an information source.

DEVTAB=*addr or *—RX-type address, (2-12), (0), (14)*
Is coded as shown in the standard form except for the * subparameter. Specify an * when you have inserted the address of the device characteristics table entry (DCTE) in the parameter list.

UCB=*addr or *.—RX-type address, (2-12), (0), (14)*
Is coded as shown in the standard form except for the * subparameter. Specify an * when you have inserted the address of the UCB in the parameter list.

The TRKCALC macro accepts the address of a UCB or UCB copy. Unauthorized programs can get a copy of the UCB by using the UCBSCAN macro and specifying the COPY and UCBAREA keywords. See *z/OS HCD Planning* for more information.

TYPE=*addr or *—RX-type address, (2-12), (0), (14)*
Is coded as shown in the standard form except for the * subparameter. Specify an * when you have inserted the address of the UCB type (UCBTYP) in the parameter list.

LOC=BELOW or ANY
Is coded as shown in the standard form.

BALANCE=*addr or *—RX-type address, (2-12), (0), (14)*
Is coded as shown in the standard form except for the * subparameter. Specify an * when you have inserted the balance in the parameter list.

REMOVE=YES or NO
Is coded as shown in the standard form.

MAXSIZE=YES or NO
Is coded as shown in the standard form.

RKDD=*addr*—RX-type address, (2-12), (0), (14)
Is coded as shown in the standard form.

R=*addr*—RX-type address, (2-12), (0), (14) or n
Is coded as shown in the standard form.

K=*addr*—RX-type address, (2-12), (0), (14), or n
Is coded as shown in the standard form.

DD=*addr*—RX-type address, (2-12), (0), (14), or n
Is coded as shown in the standard form.

REGSAVE=YES or NO
Is coded as shown in the standard form.

MF=(E,*addr*)
This operand specifies that the execute form of the TRKCALC macro instruction and an existing data management parameter list are used.

E
Coded as shown.

***addr*—RX-type address, (0), (1), (2-12), or (14)**
Specifies the address of the parameter list.

TRKCALC—List Form

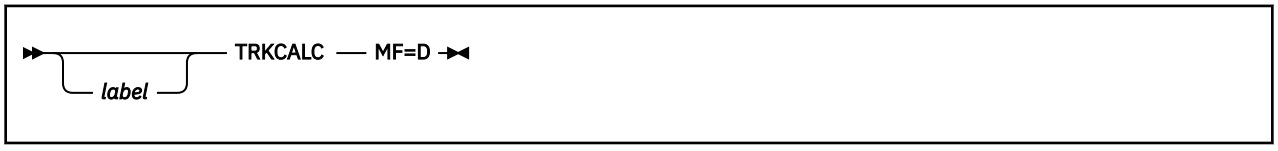
The list form of the TRKCALC macro constructs an empty, in-line parameter list. By coding only MF=L, you construct a parameter list, and the actual values can be supplied by the execute form of the TRKCALC macro. Any parameters other than MF=L are ignored.

The format of the list form of the TRKCALC macro is:

TRKCALC—DSECT Only

This call gives a symbolic expansion of the parameter list for the TRKCALC macro. No DSECT statement is generated. If a name is specified on the macro call, it applies, after any necessary boundary alignment, to the beginning of the list. The macro-generated symbols all begin with the characters STAR.

The format of the DSECT form of the TRKCALC macro is:



Input Register Usage for All Forms of MF

Register
Use

0, 2-12, 14
Available to provide input for keywords.

1
Only to provide the address of the parameter list for an MF=E call.

13
Input for keywords if REGSAVE=YES is not specified.

15
Work register to build the TRKCALC parameter list for the MF=E call; it is not available as an input register.

Output from TRKCALC

FUNCTN=TRKBAL:

Output
Meaning

R15=X'00'
The record fits on the track. Register 0 and STARBAL contain the new track balance.

R15=X'04'
Record does not fit on the track. If MAXSIZE=YES is specified, a partial record does not fit either. Register 0 and STARBAL are set to zero.

R15=X'08'
Record does not fit on the track. MAXSIZE=YES is specified, and a partial record does fit. Register 0 and STARBAL are set to the maximum number of data bytes that fit on the remainder of the track with the specified key length.

The key length is excluded from the count of maximum data bytes.

R15=X'0C'
The user supplied a device type, but the device characteristics table indicated that no device of that type was generated on the system. Register 0 is set to zero.

STARBAL
This is the track balance field of the TRKCALC parameter list. This field is first set to the track capacity if R is equal to 1, or to the supplied BALANCE value if R is not equal to 1, or to the calculated balance if R is not equal to 1 and BALANCE are omitted. STARBAL is updated to the new track balance if the record fits; otherwise, STARBAL is left with the input track balance value.

FUNCTN=TRKCAP :

Output
Meaning

R15=X'00'
Register 0 contains the number of records that fit on the track if R is equal to 1, or the number of records that fit on the remainder of the track if R is not equal to 1.

R15=X'04'
No records of the length specified fit on a full track (R is equal to 1) or a partial track (R is not equal to 1). Register 0 is set to zero.

R15=X'0C'
The user supplied a device type, but the device characteristics table indicated that no device of that type was generated on the system. Register 0 is set to zero.

STARBAL
This is the track balance field of the TRKCALC parameter list. This field is first set to the track capacity if R is equal to 1, or to the supplied BALANCE value if R is not equal to 1, or to the calculated balance if R is not equal to 1 and BALANCE is omitted. STARBAL is updated to the new track balance if the record fits; otherwise, STARBAL is left with the input track balance value when the request specified MAXSIZE=NO or was defaulted to MAXSIZE=NO. If the record does not fit on the track with MAXSIZE=YES specified, STARBAL is set to the maximum number of data bytes of a partial record that can fit on the remainder of the track with the specified key length (Register 15 set to 8) or STARBAL is set to zero when a partial record could not fit on the remainder of the track (Register 15 set to 4).

Return Codes from TRKCALC

The TRKCALC macro passes a return code in register 15. The return codes and their meanings are as follows:

Return Code	Meaning
0 (X'00')	Indicates that register 0 contains the new track balance.
4 (X'04')	Indicates that the record did not fit (register 0 = 0).
8 (X'08')	Indicates that the record did not fit. (Register 0 contains the maximum data length that does fit.)
12 (X'0C')	The system could not find an entry in the device characteristics table whose attributes match those of the user-specified device type. Register 0 is set to zero.

TRKCALC Macro Examples

In this example, TRKCALC is coded to determine how many records of a given size with 10-byte keys fit on an IBM 3390 track. After issuing the macro, the number of records is saved in NUMREC:

```
          TRKCALC  FUNCTN=TRKCAP,TYPE=UTYPE,R=1,K=10,DD=DL,            X
                MF=(E,(1))
          .
          .
          ST   0,NUMREC   SAVE NUMBER OF RECORDS
          .
          .
DL        DC   H'xxxx'    DATA LENGTH
UTYPE     DC   X'0F'
NUMREC    DS   F          MAX # OF RECORDS
```

In this example, TRKCALC is coded to determine whether another record can fit on a track of a 3390, given a track balance.

```
          TRKCALC  FUNCTN=TRKBAL,TYPE=UTYPE,R=REC,K=KL,DD=DD,          X
                BALANCE=BAL,MAXSIZE=YES,MF=(E,(1))
          .
          .
UTYPE     DC   X'0F'
REC       DC   X'xx'
KL        DC   X'xx'
DD        DC   H'xxxx'
BAL       DC   H'xxxx'
```

Perform calculations and conversions with track addresses (TRKADDR macro)

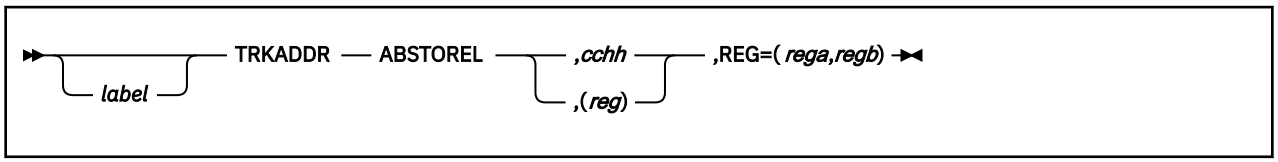
TRKADDR is an assembler macro that performs conversion and compare operations on DASD track addresses in the form *CCCCcccH*, where *CCCC* is the 16 low order bits of the cylinder number and ccc is the 12 high order bits of the cylinder number. This is referred to as a 28-bit cylinder address. TRKADDR works equally well with track addresses that contain a cylinder number less than or greater than 16 bits. It works with all tracks on all DASD types that are supported by z/OS. Its functions include:

- Calculate the relative track number on the volume
- Compare two track addresses
- Extract the 28-bit cylinder number
- Extract the 4-bit track number
- Increment the track address by one track and increments the cylinder number if necessary.
- Normalize cylinder number to permit comparing one *cchh* against another
- Convert a relative track number to a 28-bit cylinder address
- Set the cylinder number in a 28-bit track address
- Convert a normalized track address into an absolute 28-bit track address.

Unless otherwise stated, you can specify any registers from 0 to 15 except that register 0 cannot be used to address storage. TRKADDR does not use any other registers, even register 13. You can invoke TRKADDR in 24-bit, 31-bit or 64-bit mode. If you use the SYSSTATE macro with AMODE64=YES in an earlier source code statement, then TRKADDR might generate more efficient code.

Calculate the relative track number on the volume (TRKADDR ABSTOREL)

The format of the execute form of the TRKADDR ABSTOREL macro is:



Converts absolute track address (*CCCCcccH*) to a relative track number. Calculates the relative track number on the volume and stores the result in the first register. The second register is used as a work register.

Parameters

cchh
Input: Track address in absolute format

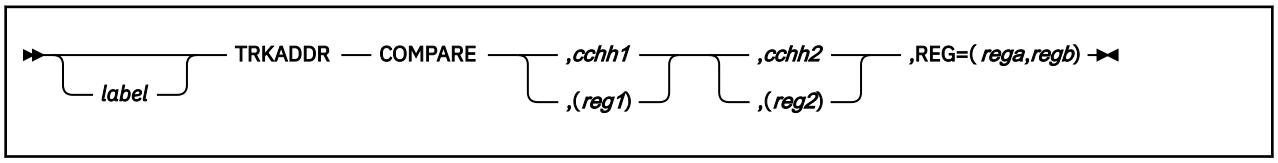
(*reg*)
This is a register from 1 to 15 containing the address of the cchh.

rega
Output: Relative track number

regb
Work register

Compare two track addresses (TRKADDR COMPARE)

The format of the execute form of the TRKADDR COMPARE macro is:



Compares two track addresses in storage using the two registers as work registers. Sets condition code as for CLC machine instruction. Normalizes the two input values (*CCCCcccH* to cccCCCCH) and then compares the two normalized values. The input values are returned unchanged.

Parameters

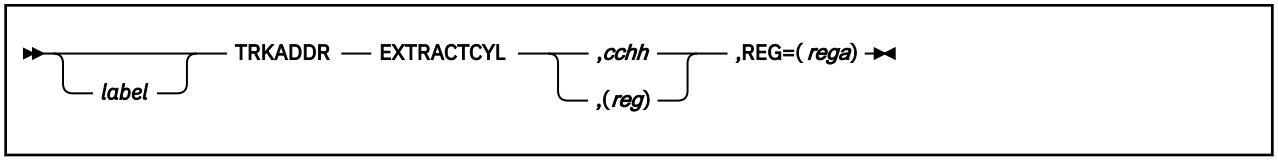
cchh1* and *cchh2
Input: Track addresses in absolute format to be compared. These value are returned unchanged.

(*reg1*) and (*reg2*)
This is a register from 1 to 15 containing the address of the cchh.

rega*, *regb
Work registers

Extract 28-bit cylinder number (TRKADDR EXTRACTCYL)

The format of the execute form of the TRKADDR EXTRACTCYL macro is:



Extracts the 28-bit cylinder number to a register (*CCCCcccH* to 0cccCCCC). The input field is returned unchanged.

Parameters

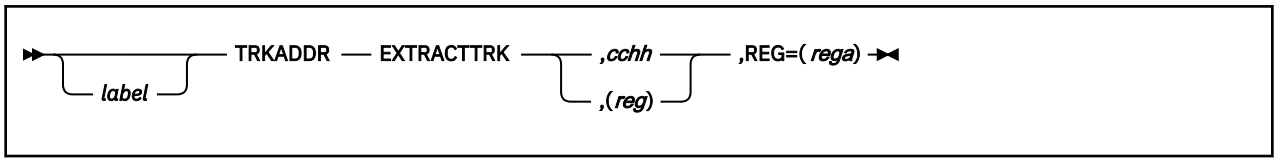
cchh
Input: Track address in absolute format

(*reg*)
This is a register from 1 to 15 containing the address of the cchh.

rega
Output: Cylinder number from the input track address

Extract 4-bit track number (TRKADDR EXTRACTTRK)

The format of the execute form of the TRKADDR EXTRACTTRK macro is:



Extracts the 4-bit track number to a register (*CCCCcccH* to 0000000H). The input field is returned unchanged.

Parameters

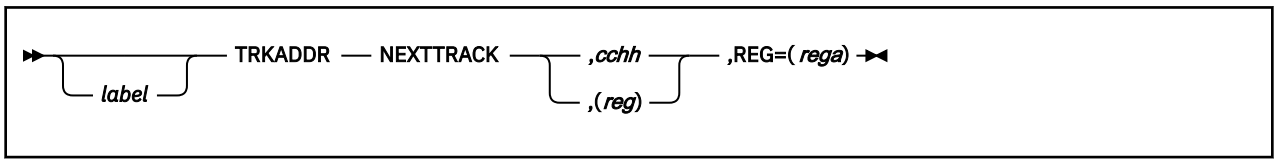
cchh
Input: Track address in absolute format

(*reg*)
This is a register from 1 to 15 containing the address of the cchh.

rega
Output: Track number from the input track address

Increment track address (TRKADDR NEXTTRACK)

The format of the execute form of the TRKADDR NEXTTRACK macro is:



Increments the track address by one track and increments the cylinder number if necessary. The modified value is returned in the input *cchh* field. The register is used as a work register.

Parameters

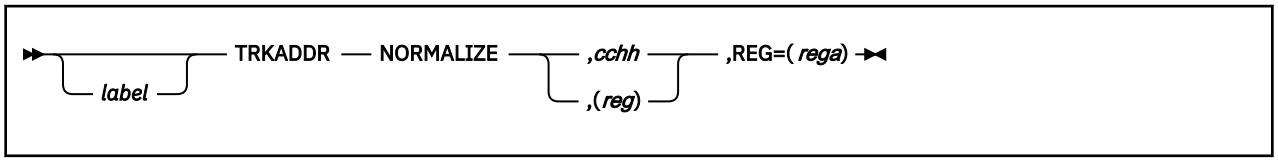
cchh
Input/Output: Track address in absolute format (*CCCCcccH*). Upon completion of the operation, this parameter contains the incremented track address in absolute format.

(*reg*)
This is a register from 1 to 15 containing the address of the cchh.

rega
Work register

Normalize cylinder number (TRKADDR NORMALIZE)

The format of the execute form of the TRKADDR NORMALIZE macro is:



Reverses the 16-bit and 12-bit portions of the cylinder number and stores the result in the 32-bit register with the H digit so you can use a simple unsigned comparison. The *CCCCcccH* becomes cccCCCCH. Use this when comparing one *cchh* against another. Normalize each and do an unsigned comparison.

Parameters

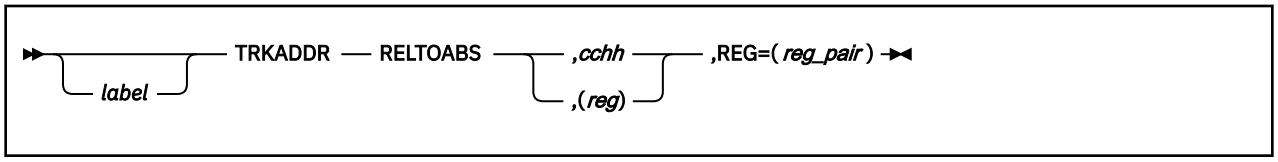
cchh
Input: Track address in absolute format.

(*reg*)
This is a register from 1 to 15 containing the address of the cchh.

rega
Output: Normalized track address

Convert a relative track number to a 28-bit cylinder address (TRKADDR RELTOABS)

The format of the execute form of the TRKADDR RELTOABS macro is:



Converts relative track number to absolute format (*CCCCcccH*). RELTOABS converts a relative track number to a 28-bit cylinder address form in the passed *cchh* field. The register must be the first in an even/odd pair. The odd register must contain the relative track number on the volume. The macro modifies both registers. In 24-bit and 31-bit addressing modes these are four-byte registers. In 64-bit mode, they are eight-byte registers.

Parameters

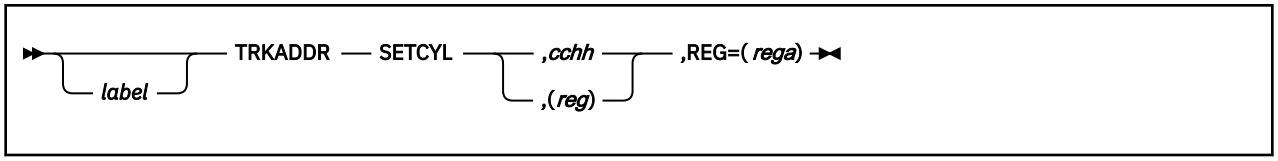
cchh
Output: Converted track address in absolute format.

(*reg*)
This is a register from 1 to 15 containing the address of the cchh.

reg_pair
Input: The first register of an even/odd pair where the odd register contains the track address to be converted.

Set cylinder number from register (TRKADDR SETCYL)

The format of the execute form of the TRKADDR SETCYL macro is:



Stores the cylinder number from the register to the 28-bits in the *cchh* and sets H to 0 (*0cccCCCC* to *CCCCccc0*). Destroys the register.

Parameters

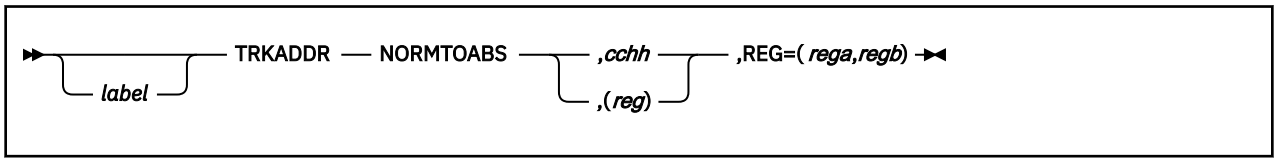
cchh
Output: Contains the cylinder number

(*reg*)
This is a register from 1 to 15 containing the address of the cchh.

rega
Input: Contains the cylinder number to be converted

Convert normalized track address into an absolute 28-bit track address (TRKADDR NORMTOABS)

The format of the execute form of the TRKADDR NORMTOABS macro is:



Reverses the 12-bit and 16-bit portions of the cylinder number and stores the result in the 32-bit register with the H digit. The cccCCCCH becomes *CCCCcccH*. Use this to convert a normalized track address to an absolute 28-bit track address.

Parameters

cchh
Input: Cylinder address to be converted

(*reg*)
This is a register from 1 to 15 containing the address of the cchh.

rega
Output: Contains the converted value

regb
Work register

Determining Level and Name of DFSMS

You can use the IHADFA mapping macro to determine the level and name of DFSMS. It maps the data facilities area. Use the CVT mapping macro to define symbol CVTDFA, which points to the DFA. The DFARELS field in the DFA is four bytes that designate the product level.

Determining Version, Release, and Modification Level of DFSMS

The first byte of DFARELS contains a binary value that indicates the level of DFSMS on which your program is running:

Value
Meaning

0
Your program is not executing on DFSMS; it is executing on MVS/XA DFP Version 2 or MVS/DFP Version 3 and the following three bytes also contain zeroes. On those two products you can determine the release level by examining the two-byte field DFAREL. DFAREL is described in the comments in IHADFA.

1
Your program is running on DFSMS/MVS and the following three bytes designate the version, release and modification level of DFSMS/MVS. A value of X'01010200' in DFARELS designates DFSMS/MVS Version 1, Release 2, Modification level 0.

2
Your program is running on the level of DFSMS that is exclusive to OS/390® or one of the first two releases of z/OS. A value of X'02020A00' in DFARELS designates DFSMS for OS/390 Version 2 Release 10, Modification level 0. DFSMS was not modified in the first two releases of z/OS, so these releases also have a value of X'02020A00'.

>3
Your program is running on a level of DFSMS that is part of a hypothetical replacement product after all versions and releases of z/OS. The system never returns this value. This represents IBM's intent in case there is such a product. The following three bytes designate the version, release, and modification level of that product. The value in the other three bytes is X'010100' or higher. It may differ from the level of installed z/OS.

3
Your program is running on a level of DFSMS that is exclusive to z/OS Version 1 Release 3 or higher. The following three bytes designate the version, release, and modification level of z/OS for which that DFSMS was designed. The value in the other three bytes is X'010300' or higher. It may differ from the level of installed z/OS.

IBM intends that for any future level of the DFA, the 4-byte DFARELS will not contain a value smaller than any previous value. If your purpose in testing DFARELS is to determine whether a particular feature of DFSMS is available, then we suggest that your program test all four bytes of DFARELS. IBM intends that if one of the low-order three bytes of DFARELS contains a value that is smaller than the corresponding byte in the prior release, then a higher order byte will contain a larger value.

For compatibility with programs that were designed to run on MVS/XA DFP Version 2 or MVS/DFP Version 3, DFSMS sets DFAREL to the value X'3321', which designates MVS/DFP Version 3, Release 3, modification level 2. The last digit indicates that the system actually is at a higher level than DFP 3.3.2.

See also “Call for DFSMS Level Determination” on page 322 for an alternative method of determining the level of DFSMS.

See “Data Facilities Area (DFA) Fields” on page 438 for a layout of the fields of the Data Facilities Area (DFA) control block.

Determining Name of DFSMS

If the value of DFARELS is '03010300' or greater, it means the system is z/OS Version 1 Release 3.0 or later. This means that field DFAELNMP points to a structure that contains the name of DFSMS. See DFAELNM in “Data Facilities Area (DFA) Fields” on page 438.

Determining DFARELS During Assembler Macro Phase

Your program can test the DFARELS field during execution as described earlier. This does not allow you to assemble a program that optionally uses a new macro parameter that is available only on a certain level of the system. Your program receives syntax error messages if assembled on an older level of the system.

A solution is to test a macro variable symbol set by the IHADFA macro. The name of the symbol is &IHADFARELS and it is a character type of global variable symbol. Your program's test of its value must follow the IHADFA invocation.

The other system facilities determine whether your program can run on a different release than the one on which it was assembled. For some new functions the older release will ignore the new function. Other new functions will fail on an older release.

The IHADFA macro sets the variable symbol &IHADFARELS to an eight-character value. Each pair of characters in the value represents the decimal value of one byte in DFARELS. They are not hexadecimal digits because the EBCDIC values of "A" to "F" are not in proper collating sequence with the numeric digits. For example the value for z/OS Version 1 Release 10 is ' 03011000', '03' represent the name z/OS, ' 01' represents Version 1, ' 10' represents release 10, ' 00' represents modification level 0.

This is an example of a program using &IHADFARELS:

```
 xxxx    CSECT
           .
           .
           .
         GBLC  &IHADFARELS    Set by IHADFA macro to be system level
         IHADFA ,             Set &IHADFARELS and define DFARELS
 xxxx    CSECT                Reset CSECT
           .
           .
           .
* Expand one of two macro invocations.  Either works on any DFSMS
* release.  If in 31-bit mode on 1.3 or later, then ANY means a UCB
* may be above the line.  Neither works on DFP Version 3 when assembled
* on DFSMS.
         AIF   ('&IHADFARELS' LT '01010300').OLD
* If executing in 31-bit mode on 1.3 or later, this requires that each
* UCB address be 31-bit.  They may point below the line.  On an older
* level of DFSMS, the ANY has no effect.  DFP 3.x will reject it.
         DEVTYPE UCBLIST=(MYLIST,1,ANY),MF=(E,DEVTLIST)
         AGO   .CONT
.OLD     DEVTYPE UCBLIST=(MYLIST,1),MF=(E,DEVTLIST)
.CONT    ANOP
           .
MYLIST   DC    A(0)
DEVTLIST DEVTYPE ,(DEVINFO,24),MF=L
```

Figure 34. Sample &IHADFARELS Program

The IHADFA macro as shipped prior to DFSMS/MVS V1R3 did not set &IHADFARELS. You can use the technique in the example even if IHADFA does not set &IHADFARELS.

This technique of using IHADFA to decide on another macro invocation assumes that IHADFA resides in a complete macro library for the same release as the other macro. It might not work properly with a macro from a different release or product.

Following is an example of determining whether a mapping macro has defined a symbol that is needed during the assembly. During execution, the program tests DFARELS to determine how to execute.

```
          GBLC  &IHADFARELS   Set by IHADFA macro to be system level
          IHADFA ,            Learn release of assembly & execution
 TRKLIST  DSECT
          TRKCALC MF=D        DSECT for TRKCALC parameter list
          SPACE 2
 * If global symbol &IHADFARELS has a null value or is less than
 * 01010300, then TRKCALC did not define a certain symbol.  Since
 * other parts of this program use it, it must be defined.
          AIF   ('&IHADFARELS' GE '01010300').GOTBIT   Go if newer
 STARLOC  EQU   X'01'     LOC=ANY.  DEVTAB or UCB may be above line
 .GOTBIT  ANOP
```

Figure 35. Example of Determining Symbol Definition

Chapter 8. Displaying Messages on Cartridge Magnetic Tape Subsystems (MSGDISP macro)

This information covers using the MSGDISP macro to display messages on magnetic tape devices that have displays. With MSGDISP, you can specify the message to be displayed and how to display it (for example, steady or flashing). The standard, executes, and list forms of the macro are described here. The six main parameters of the macro and their functions are:

Value	Meaning
MOUNT	Displays an 'M' in position 1 of the display area during a mount request until a volume is loaded and made ready. The 'M' is followed by the volume serial number and label type.
VERIFY	Shows that a volume has been accepted by displaying its serial number and label type in positions 2 through 8.
RDY	Displays text in positions 2 through 7 while a data set is open.
DEMOUNT	Places a volume disposition indicator in position 1 of the display until a volume is demounted.
RESET	Clears the display area.
GEN	Provides the full range of display options, including the option to alternate two messages.

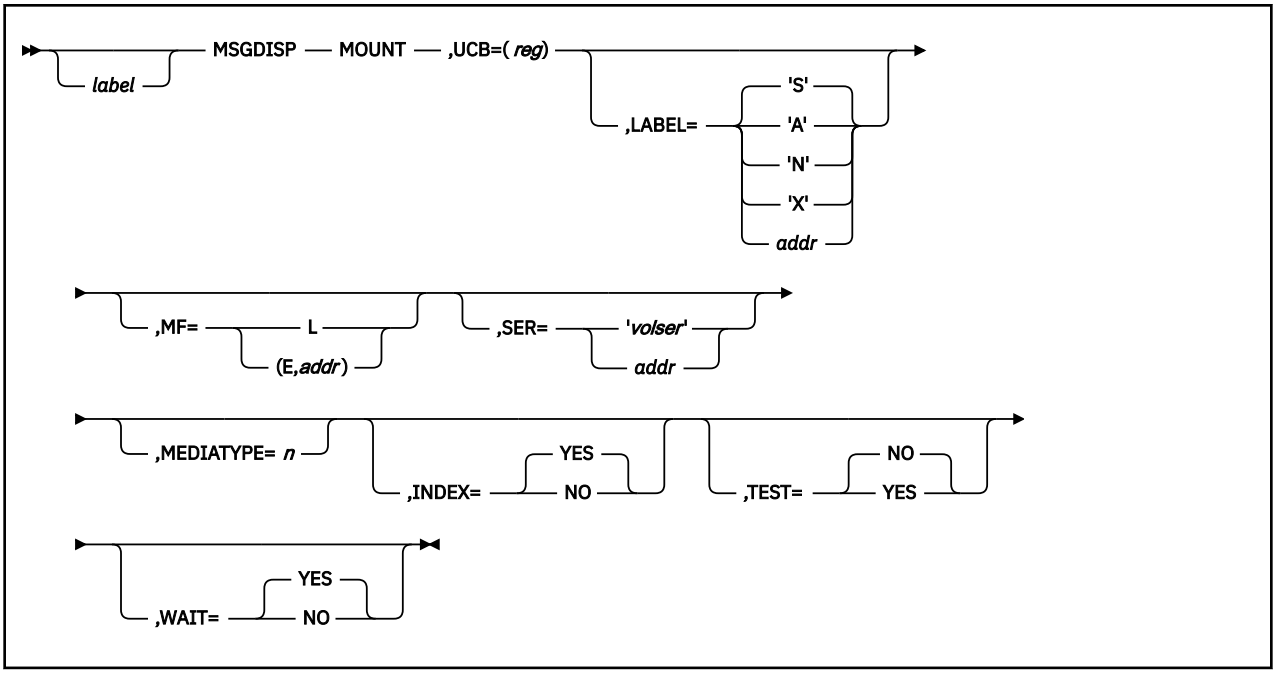
All except the RDY parameter require that the caller be in supervisor state, have a storage protect key of 0 through 7, or be authorized by the authorized program facility.

You can issue the MSGDISP macro in 24- or 31-bit addressing mode. When you use 31-bit addressing mode, all addresses must be valid 31-bit addresses.

The MSGDISP macro generates a parameter list as input to the message display service routine. You can code an installation exit routine named IGXMSGEX,which gains control when MSGDISP is processing MOUNT, DEMOUNT, VERIFY, or GEN requests. The exit can change the message text displayed (two 8-byte strings) and 1 bit of the format control byte. See *z/OS DFSMS Installation Exits* for details.

MSGDISP—Displaying a Mount Message

The format of the MSGDISP macro with the MOUNT parameter is:



MOUNT
Displays an 'M' in position 1 of the display area during a mount request. The 'M' is followed by a volume serial number and label type. The display flashes on and off until a volume is loaded and ready. If the device is ready at the time a mount request is issued, the 'M' is not displayed.

UCB=(*reg*)—(2-12)
Specifies a register containing the UCB address for the device. Use the address of a UCB, not a UCB copy.

LABEL='A' or 'N' or 'S' or 'X' or *addr*
Displays the label type of the mounted volume in position 8. If you specify an unknown label type other than a blank, a "?" is displayed.

'A'
Specifies ISO/ANSI (AL) or ISO/ANSI with user labels (AUL). Apostrophes are required.

'N'
Specifies no labels (NL), LTM (VSE), or bypass label processing (BLP) Apostrophes are required.

'S'
Specifies IBM Standard (SL) or IBM Standard with user labels (SUL). Apostrophes are required.

'X'
Specifies nonstandard labels (NSL). Apostrophes are required.

***addr*—RX-type address, A-type address, or (2-12)**
Specifies the address of an area containing an “A”, “N”, “S”, or “X”. (See the following explanations of these characters.) For MF=L, you can only specify an A-type address.

MF=L or (E,*addr*)
Specifies either the execute or the list form of MSGDISP. If you do not specify this parameter, the standard form of the macro is used.

L
Specifies the list form of MSGDISP. This generates a parameter list that can be used as input to the execute form. The execute form can modify the parameter list.

(E,*addr*)
Specifies that the execute form of the macro and an existing parameter list are used.

***addr*—RX-type address, (1), or (2-12)**
Specifies the address of the parameter list.

SER='*volser*' or *addr*
Specifies the serial number of the volume to be mounted. The serial number is displayed in positions 2 through 7. If you do not specify SER, the system supplies the volume serial number. If the serial number is not available, a scratch volume is used, unless the volume use attribute indicates a default of "PRIVAT".

'*volser*'
Specifies the volume serial number as a literal. Specify in apostrophes.

***addr*—RX-type address, A-type address, or (2-12)**
Specifies the address of the volume serial number. For MF=L, you can only specify an A-type address.

MEDIATYPE=*n*
Specifies what media type to mount for SCRTCH or PRIVAT mounts. The MEDIATYPE keyword applies only when volumes are to be mounted on devices that reside in a Manual Tape Library (MTL). If MEDIATYPE is specified for devices outside of a Manual Tape Library, it is ignored. The value n can be specified as a literal, the address of a 1 byte field containing the value, or the name of the addressable field containing the value. Valid values for MEDIATYPE are the numbers 1 through 8.

TEST=NO or YES
Specifies whether the macro expansion is to include code that tests the UCB to determine whether message display is supported. If the result of the test is that the message display is not supported, an SVC is not invoked.

NO
Specifies that the macro expansion is not to include code that tests the UCB to determine whether the device supports message display.

YES
Specifies testing the UCB by the MSGDISP macro before attempting to invoke the message display service routine.

Requirement: TEST=YES requires you to include the UCB mapping macro (IEFUCBOB) in the source code.

Restriction: There is a restriction when using TEST=YES. Programs running in AMODE 24 and invoking the MSGDISP macro with the TEST=YES parameter cannot pass the actual address of a UCB that resides above the 16 MB line. These programs must pass the captured UCB address or, if an actual address is passed, the UCB must reside below the 16 MB line.

INDEX=NO or YES
Specifies whether the automatic cartridge loader (ACL) should be indexed to satisfy a scratch mount request.

NO
Specifies that indexing should not be done regardless of the state of the ACL.

YES
Specifies that indexing should be done if:

- The ACL is present and loaded, and
- The request is for SCRTCH or PRIVAT.

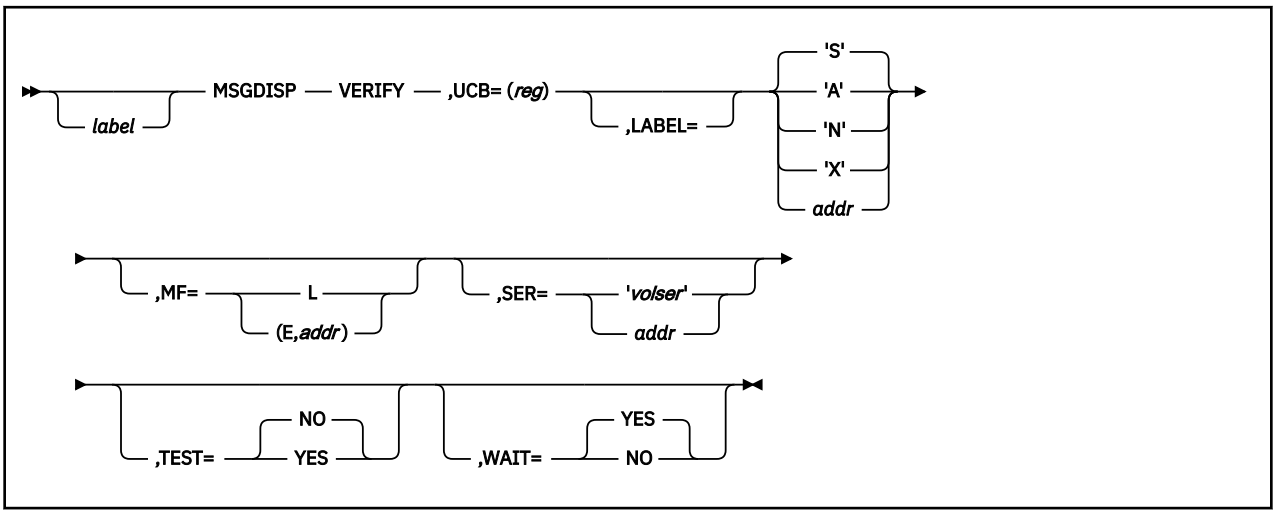
WAIT=NO or YES
Specifies when control is returned to you.

NO
Specifies that the MSGDISP function is not to wait for completion of I/O initiated on the caller's behalf. When MSGDISP returns, the I/O request might still be running. I/O return codes are not returned, and I/O errors are recorded in the same manner as any permanent error by the error recovery procedure.

YES
Specifies that control is to be returned after I/O is complete.

MSGDISP—Displaying a Verify Message

The format of the MSGDISP macro with the VERIFY parameter is:



VERIFY
Displays the serial number and label type of a volume that has been accepted for processing. The serial number is displayed on the pod in positions 2 through 7, and the 1 character label type in position 8. Position 1 remains blank. The display lasts until the next display request is executed.

UCB=(*reg*)—(2-12)
Specifies a register containing the UCB address for the device. Use the address of a UCB, not a UCB copy.

LABEL='A' or 'N' or 'S' or 'X' or *addr*
Specifies label type of the mounted volume in position 8 of the display. If an unknown label type other than a blank is specified, a "?" is displayed.

'A'
Specifies ISO/ANSI (AL) or ISO/ANSI with user (AUL) labels. Apostrophes are required.

'N'
Specifies no labels (NL), LTM (VSE), or bypass label processing (BLP). Apostrophes are required.

'S'
Specifies IBM Standard (SL) or IBM Standard with user (SUL) labels. Apostrophes are required.

'X'
Specifies nonstandard (NSL) labels. Apostrophes are required.

***addr*—RX-type address, A-type address, or (2-12)**
Specifies the address of an area containing an “A”, “N”, “S”, or “X” (see explanations below for these characters). For MF=L, you can only specify an A-type address.

MF=L or (E,*addr*)
Specifies either the execute or list form of MSGDISP. If you do not specify this parameter, the standard form of the macro is used.

L
Specifies the list form of MSGDISP. This generates a parameter list that can be used as input to the execute form. The execute form can modify the parameter list.

(E,*addr*)
Specifies that the execute form of the macro and an existing parameter list is to be used.

***addr*—RX-type address, (1), or (2-12)**
Specifies the address of the parameter list.

SER='*volser*' or *addr*
Specifies the serial number of the volume that has been verified. The serial number displays in positions 2 through 7. If you do not specify SER, the system supplies the volume serial number. If the serial number is not available, a scratch volume is used, unless the volume use attribute indicates a default of “PRIVAT”.

'*volser*'
Specifies the volume serial number as a literal. Express® in apostrophes.

***addr*—RX-type address, A-type address, or (2-12)**
Specifies the address of the volume serial number. For MF=L, you can only specify an A-type address.

TEST=NO or YES
Specifies whether the macro expansion is to include code that will test the UCB to determine whether message display is supported. If the result of the test is that the message display is not supported, an SVC is not invoked.

NO
Specifies that the macro expansion is not to include code that tests the UCB to determine whether the device supports message display.

YES
Specifies testing the UCB by the MSGDISP macro before attempting to invoke the message display service routine.

Requirement: TEST=YES requires you to include the UCB mapping macro (IEFUCBOB) in the source code. If this provision is not followed, a program check in expansion code might result. Programs running in AMODE 24 and invoking the MSGDISP macro with the TEST=YES parameter cannot pass the actual address of a UCB that resides above the 16 MB line. These programs must pass the captured UCB address or, if an actual address is passed, the UCB must reside below the 16 MB line.

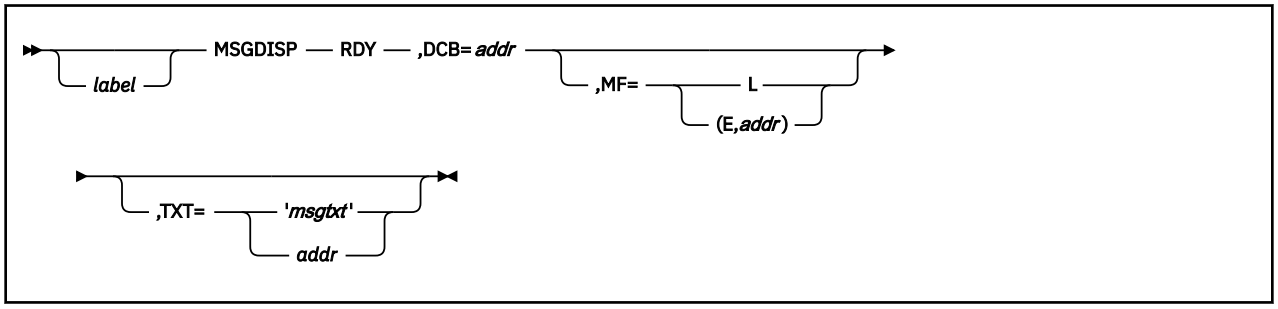
WAIT=NO or YES
Specifies when control is to be returned to you and that the MSGDISP function is not to wait for completion of I/O initiated on the caller's behalf. When MSGDISP returns, the I/O request might still be running.

NO
Specifies that the MSGDISP function is not to wait for completion of I/O that is initiated on the caller's behalf. When MSGDISP returns, the I/O request might still be running. I/O return codes are not returned, and I/O errors are recorded in the same manner as any permanent error by the error recovery procedure.

YES
Specifies that control is to be returned after I/O is complete.

MSGDISP—Displaying a Ready Message

The format of the MSGDISP macro with the RDY parameter is:



RDY
Displays the text supplied in the TXT parameter in positions 2 through 7 while the data set is open. The display is steady (not flashing) and is enclosed in parentheses. The display is also written to the tape pool console (routing code 3, descriptor code 7).

DCB=*addr*
Specifies the address of a DCB opened to a data set on the mounted volume. If multiple devices are allocated, the message display is directed to the one containing the volume currently in use.

Tip: If multiple devices or multiple volumes are allocated, you can update a message display after an end-of-volume condition by using the EOV user exit specified in a DCB exit list. In the case of a concatenated data set with unlike characteristics, the DCB OPEN exit can also be used to update the display.

***addr*—RX-type address, A-type address, or (2-12)**
Specifies the address of the opened DCB. For MF=L, you can only specify an A-type address.

MF=L or (E,*addr*)
Specifies either the execute or list form of MSGDISP. If this parameter is not specified, the standard form of the macro is used.

L
Specifies the list form of MSGDISP. This generates a parameter list that can be used as input to the execute form. The execute form can modify the parameter list.

(E,*addr*)
Specifies that the execute form of the macro and an existing parameter list is to be used.

***addr*—RX-type address, (1), or (2-12)**
Specifies the address of the parameter list.

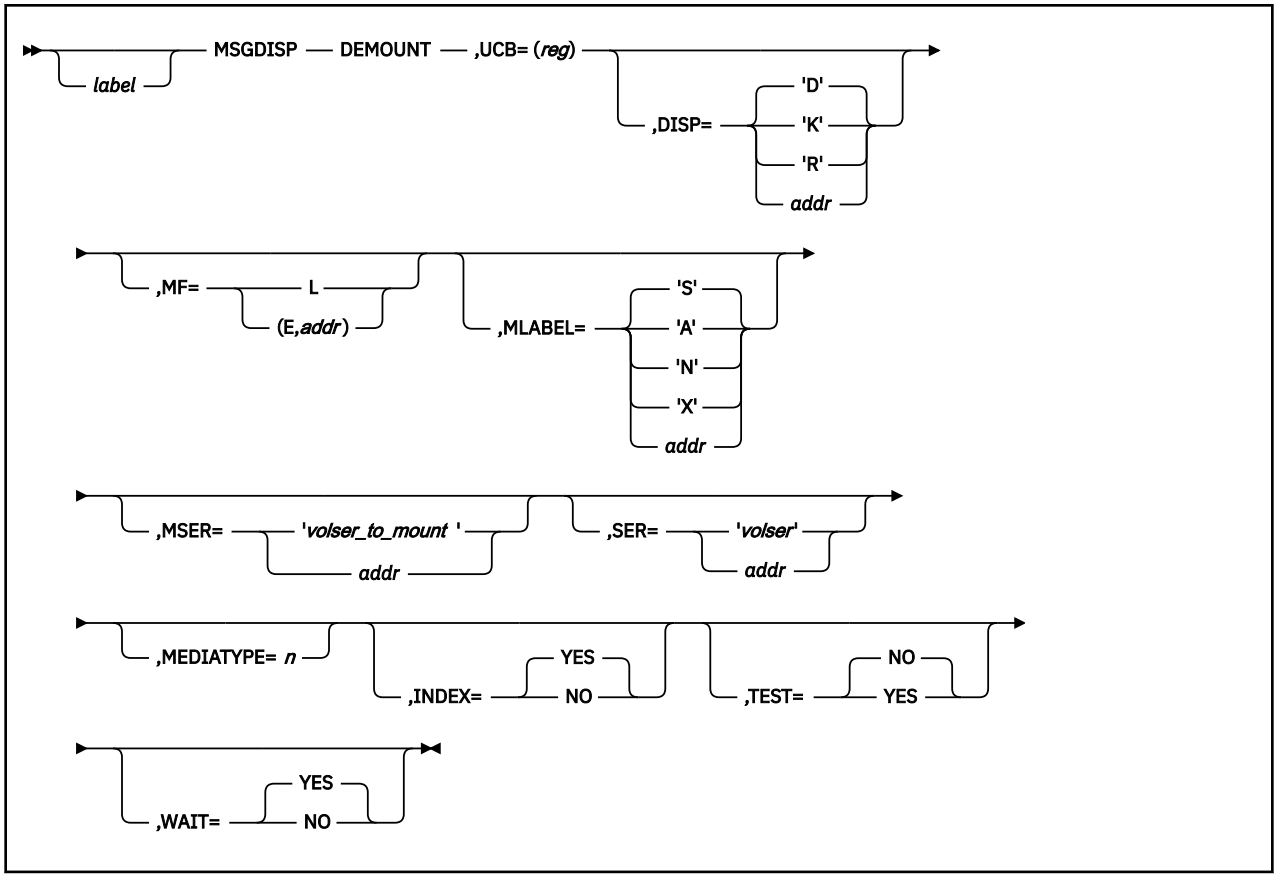
TXT='*msgtxt*' or *addr*
Specifies up to six characters to display in positions 2 through 7 of the display. If you do not specify TXT, blanks are displayed.

'*msgtxt*'
Specifies the text as a literal. Express in apostrophes.

***addr*—RX-type address, A-type address, or (2-12)**
Specifies the address of an area containing the text to be displayed. For MF=L, you can only specify an A-type address.

MSGDISP—Displaying a Demount Message

The format of the MSGDISP macro with the DEMOUNT parameter is:



DEMOUNT
Displays a volume disposition indicator in position 1 until the volume is demounted. Optionally, you can display the serial number of the volume to be demounted at the same time. The display flashes on and off. If a volume is not mounted on the device when the display request is executed, blanks are displayed.

The demount message can be displayed alternately (flashing) with a mount message for the next volume by specifying the MSER parameter.

UCB=(*reg*)—(2-12)
Specifies a register containing the UCB address for the device. Use the address of a UCB, not a UCB copy.

DISP='D' or 'K' or 'R' or *addr*
Specifies the character to display in position 1 of the pod, representing the volume disposition.

'D'
Specifies demount a public volume. Apostrophes are required. "D" also displays when you specify an invalid character or when the volume use attribute is unknown (as in an automatic volume recognition (AVR) error when reading a label).

'K'
Specifies keep a private volume and return it to the library. Apostrophes are required.

'R'
Specifies retain a private volume near the device for further use. Apostrophes are required.

***addr*—RX-type address, A-type address, or (2-12)**
Specifies the address of an area containing a “D”, “K”, or “R”. For MF=L, you can only specify an A-type address.

MF=L or (E,*addr*)
Specifies either the execute or list form of MSGDISP. If you do not specify this parameter, the standard form of the macro is used.

L
Specifies the list form of MSGDISP. This generates a parameter list that can be used as input to the execute form. The execute form can modify the parameter list.

(E,*addr*)
Specifies that the execute form of the macro and an existing parameter list is to be used.

***addr*—RX-type address, (1), or (2-12)**
Specifies the address of the parameter list.

MLABEL='A' or 'N' or 'S' or 'X' or *addr*
Displays the label type of the volume to be loaded and made ready following a demount, in position 8. If you specify an unknown label type other than a blank, a "?" is displayed. You can only specify this parameter if you also specify the MSER parameter.

'A'
Specifies ISO/ANSI (AL) or ISO/ANSI with user (AUL) labels. Apostrophes are required.

'N'
Specifies no labels (NL), LTM (leading tape mark, created by VSE), or bypass label processing (BLP). Apostrophes are required.

'S'
Specifies IBM Standard (SL) or IBM Standard with user (SUL) labels. Apostrophes are required.

'X'
Specifies nonstandard (NSL) labels. Apostrophes are required.

***addr*—RX-type address, A-type address, or (2-12)**
Specifies the address of an area containing an “A”, “N”, “S”, or “X” (see the following explanations of these characters). For MF=L, you can only specify an A-type address.

MSER='*volser-to-mount*' or *addr*
Displays the mount message for the next volume alternately (flashing) with the demount message. The display continues until you demount the current volume. At that time, the mount message will display (flashing) until you load the volume and make the device ready. If no volume is mounted at the time the demount and mount messages are executed, only the mount message will display (flashing) until the volume is loaded and ready.

'*volser-to-mount*'
Specifies the volume serial number of the volume to be mounted, as a literal. Apostrophes are required.

***addr*—RX-type address, A-type address, or (2-12)**
Specifies the address of the volume serial number of the volume to be mounted. For MF=L, you can only specify an A-type address.

SER='*volser*' or *addr*
Specifies the serial number of the volume to be demounted. The serial number is displayed in positions 2 through 7. If you do not specify SER, the system supplies the volume serial number. If the serial number is not available, a scratch volume is used, unless the volume use attribute indicates a default of “PRIVAT”.

'*volser*'
Specifies the volume serial number as a literal. Specify with apostrophes.

***addr*—RX-type address, A-type address, or (2-12)**
Specifies the address of the volume serial number. This parameter is not valid for the MF=L form. For MF=L, you can only specify an A-type address.

MEDIATYPE=*n*
Specifies what media type to demount for SCRTCH or PRIVAT demounts. The MEDIATYPE keyword applies only when volumes are to be demounted on devices that reside in a Manual Tape Library (MTL). If MEDIATYPE is specified for devices outside of a Manual Tape Library, it is ignored. The value n can be specified as a literal, the address of a 1 byte field containing the value, or the name of the addressable field containing the value. Valid values for MEDIATYPE are the numbers 1 through 8.

INDEX=NO or YES
Specifies whether the ACL should be indexed to satisfy a scratch mount request.

NO
Specifies that indexing should not be done regardless of the state of the ACL.

YES
Specifies that indexing should be done if:

- The ACL is present and loaded, and
- The request is for SCRTCH or PRIVAT.

TEST=NO or YES
Specifies whether the macro expansion is to include code that tests the UCB to determine whether message display is supported. If the result of the test is that the message display is not supported, an SVC is not invoked.

NO
Specifies that the macro expansion is not to include code that tests the UCB to determine whether the device supports message display.

YES
Specifies testing the UCB by the MSGDISP macro before attempting to invoke the message display service routine.

Requirement: TEST=YES requires you to include the UCB mapping macro (IEFUCBOB) in the source code. If this provision is not followed, a program check in expansion code might result. Programs running in AMODE 24 and invoking the MSGDISP macro with the TEST=YES parameter cannot pass the actual address of a UCB that resides above the 16 MB line. These programs must pass the captured UCB address or, if an actual address is passed, the UCB must reside below the 16 MB line.

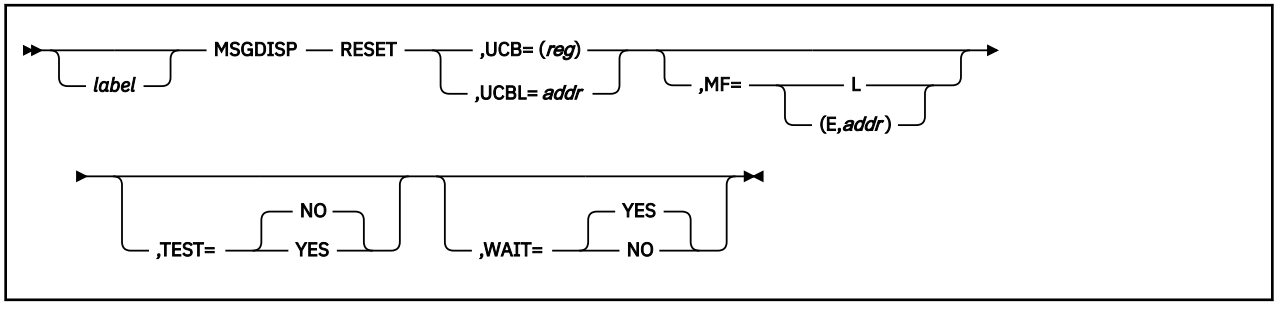
WAIT=NO or YES
Specifies when control is to be returned to you.

NO
Specifies that the MSGDISP function is not to wait for completion of I/O initiated on the caller's behalf. When MSGDISP returns, the I/O request might still be running. I/O return codes are not returned, and I/O errors are recorded in the same manner as any permanent error by the error recovery procedure.

YES
Specifies that control is to be returned after I/O is complete.

MSGDISP—Resetting the Message Display

The format of the MSGDISP macro with the RESET parameter is:



RESET
Clears all existing data on the display. If you specify WAIT=NO and the last service requested was a demount, the display is not cleared.

After being cleared, the display shows the internal status of the device (for example, a message indicating that the device is ready).

UCB=(*reg*)—(2-12)
Specifies a register containing the UCB address for the device. Use the address of a UCB, not a UCB copy.

UCBL=*addr*—RX-type address, A-type address, (0), or (2-12)
Specifies the address of a list containing a maximum of 64 words. Each word in the list contains the address of a UCB representing a device whose display is to be reset. The end of the list is indicated by a '1' in the high-order bit of the last address in the list. If an error is encountered while processing the list, register 1 points to the associated UCB when you regain control.

You cannot specify UCBL with TEST=YES and WAIT=NO.

MF=L or (E,*addr*)
Specifies either the execute or the list form of MSGDISP. If you do not specify this parameter, the standard form of the macro is used.

L
Specifies the list form of MSGDISP. This generates a parameter list that can be used as input to the execute form. The execute form can modify the parameter list.

(E,*addr*)
Specifies that the execute form of the macro and an existing parameter list is to be used.

***addr*—RX-type address, (1), or (2-12)**
Specifies the address of the parameter list.

TEST=NO or YES
Specifies whether the macro expansion is to include code that tests the UCB to determine whether message display is supported. If the result of the test is that the message display is not supported, an SVC is not invoked.

NO
Specifies that the macro expansion is not to include code that tests the UCB to determine whether the device supports message display. NO is the default?

YES
Specifies testing the UCB by the MSGDISP macro before attempting to invoke the message display service routine. You cannot specify TEST=YES if you also specify the UCBL parameter.

Requirement: TEST=YES requires you to include the UCB mapping macro (IEFUCBOB) in the source code. If this provision is not followed, a program check in expansion code might result. Programs running in AMODE 24 and invoking the MSGDISP macro with the TEST=YES parameter cannot pass the actual address of a UCB that resides above the 16 MB line. These programs must pass the captured UCB address or, if an actual address is passed, the UCB must reside below the 16 MB line.

WAIT=NO or YES
Specifies when control is to be returned to you.

NO
Specifies that the MSGDISP function is not to wait for completion of I/O initiated on the caller's behalf. When MSGDISP returns, the I/O request might still be running. I/O return codes are not returned, and I/O errors are recorded in the same manner as any permanent error by the error recovery procedure.

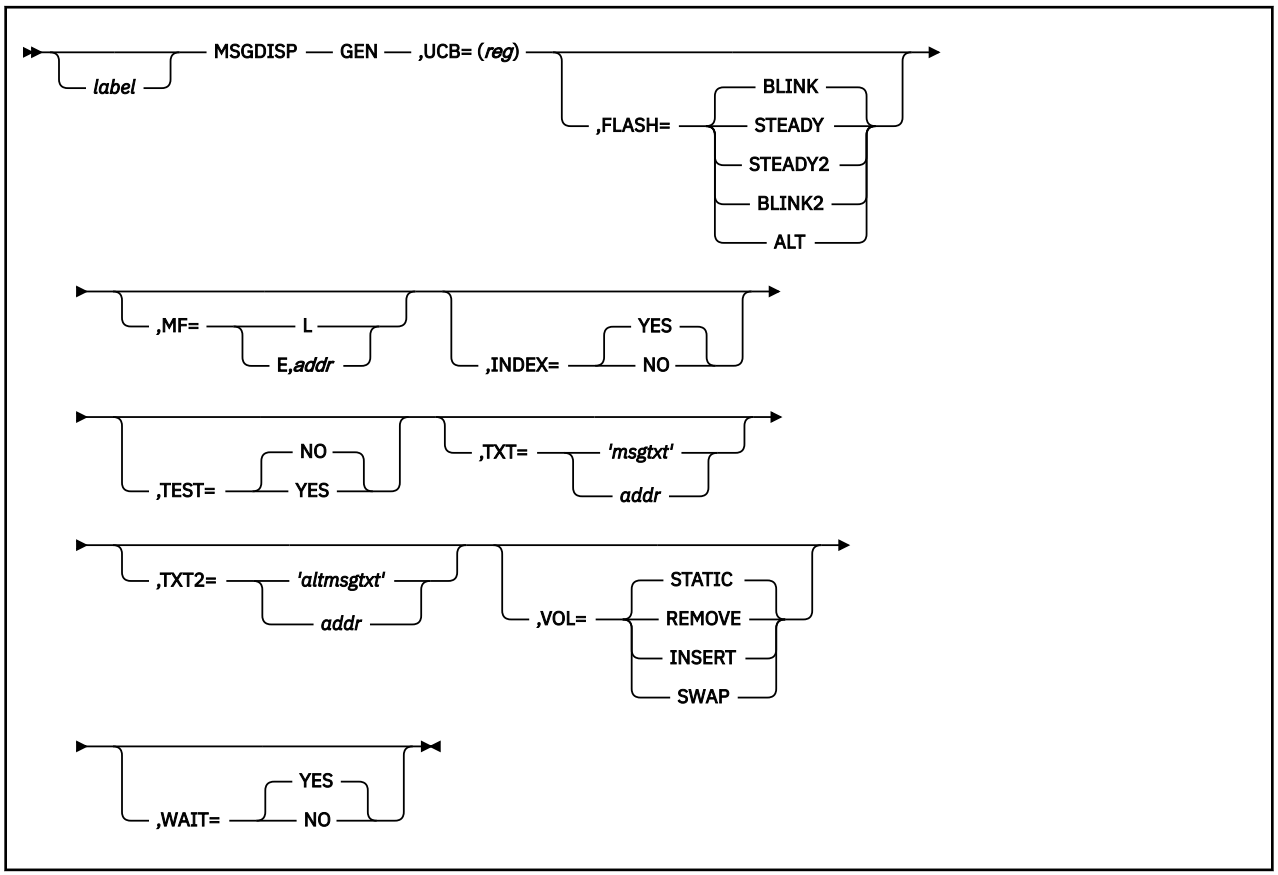
You cannot specify WAIT=NO if you also specify the UCBL parameter.

YES
Specifies that control is to be returned after I/O is complete.

Demount messages can be reset only if WAIT=YES is specified.

MSGDISP—Providing the Full Range of Display Options

The format of the MSGDISP macro with the GEN parameter is:



GEN
Specifies the full range of display options.

UCB=(*reg*)—(2-12)
Specifies a register containing the UCB address for the device. Use the address of a UCB, not a UCB copy.

FLASH=STEADY or STEADY2 or BLINK or BLINK2 or ALT
Specifies message display mode.

Hint: If you specify VOL=SWAP, messages will always be displayed as if you had specified FLASH=ALT.

STEADY
Specifies that the primary message (TXT) is to be displayed without flashing.

STEADY2
Specifies that the alternate message (TXT2) is to be displayed without flashing.

BLINK
Specifies that the primary message (TXT) flash on and off at a rate of approximately two seconds on and one-half second off.

BLINK2
Specifies that the alternate message (TXT2) flash on and off at a rate of approximately two seconds on and one-half second off.

ALT
Specifies that the primary and alternate messages (TXT and TXT2) flash on and off alternately, at a rate of approximately two seconds on and one-half second off.

MF=L or (E,*addr*)
Specifies either the execute or the list form of MSGDISP. If you do not specify this parameter, the standard form of the macro is used.

L
Specifies the list form of MSGDISP. This generates a parameter list that can be used as input to the execute form. The execute form can modify the parameter list.

(E,*addr*)
Specifies that the execute form of the macro and an existing parameter list is to be used.

addr
Specifies the address of the parameter list. Specify either an RX-type address or a register in the range of 2 through 12.

INDEX=NO or YES
Specifies whether the ACL should be indexed to satisfy a scratch mount request.

NO
Specifies that indexing should not be done regardless of the state of the ACL.

YES
Specifies that indexing should be done if:

- The ACL is present and loaded, and
- The request is for SCRTCH or PRIVAT.

TEST=NO or YES
Specifies whether to test the UCB to determine if the device is capable of displaying messages.

NO
Specifies that the macro expansion is not to include code that tests the UCB to determine whether the device supports message display.

YES
Specifies testing the UCB by the MSGDISP macro before attempting to invoke the message display service routine.

Requirement: TEST=YES requires you to include the UCB mapping macro (IEFUCBOB) in the source code. If this provision is not followed, a program check in expansion code might result. Programs running in AMODE 24 and invoking the MSGDISP macro with the TEST=YES parameter cannot pass the actual address of a UCB that resides above the 16 MB line. These programs must pass the captured UCB address or, if an actual address is passed, the UCB must reside below the 16 MB line.

TXT='*msgtxt*' or *addr*
Specifies 8 characters to be shown in positions 1 through 8 of the display. If you do not specify TXT, blanks are displayed.

'*msgtxt*'
Specifies the 8 characters as literals. Apostrophes are required.

***addr*—RX-type address, A-type address, or (2-12)**
Specifies the address of an area containing the 8 characters. For MF=L, you can only specify an A-type address.

TXT2='*altmsgtxt*' or *addr*
Specifies 8 alternate characters to display in positions 1 through 8 of the display. If you do not specify TXT2, blanks are displayed.

'*altmsgtxt*'
Specifies the 8 characters as literals. Apostrophes are required.

***addr*—RX-type address, A-type address, or (2-12)**
Specifies the address of an area containing the 8 characters. For MF=L, you can only specify an A-type address.

VOL=STATIC or REMOVE or INSERT or SWAP
Specifies message display mode, based on volume status.

STATIC
Specifies that messages display without regard to volume status until the next message request is executed, or until the next command initiates volume movement.

REMOVE
Specifies that messages display until the current volume is demounted. This parameter is ignored if a volume is not mounted when the request is executed.

INSERT
Specifies that messages display until a volume is present, the tape is threaded, and the active/inactive switch is in the active position. This parameter is ignored if a volume is loaded and ready when the request is executed.

SWAP
Specifies that messages always display as if FLASH=ALT were specified. The data from TXT and TXT2 displays alternately (flashing) until the current volume has been demounted. Then only TXT2 displays (flashing) until a new volume is loaded and ready. If no volume is mounted when this parameter is specified, only TXT2 data displays (flashing) until a new volume is loaded and ready.

WAIT=NO or YES
Specifies when control is to be returned to you.

NO
Specifies that the MSGDISP function is not to wait for completion of I/O initiated on the caller's behalf. When MSGDISP returns, the I/O request might still be running. I/O return codes are not returned, and I/O errors are recorded in the same manner as any permanent error by the error recovery procedure.

YES
Specifies that control is to be returned after I/O is complete. This is the default.

Return Codes from MSGDISP

When the system returns control to the problem program, the low-order byte of register 15 contains a return code. The low-order byte of register 0 can contain a reason code. These codes are described in the following table:

Return Code	Reason Code	Meaning
0 (X'00')		Successful completion.
4 (X'04')		Device does not support MSGDISP.
8 (X'08')	1 (X'01')	Invalid input parameter.
	2 (X'02')	Invalid DCB or DEBCHK error.
	3 (X'03'	Environmental error.
	4 (X'04')	Authorization (TESTAUTH) violation.
	5 (X'05')	Invalid UCB. Requires the address of a UCB, not a UCB copy.
	6 (X'06')	Invalid request.
	11 (X'0B')	Unsuccessful ESTAE macro call.
	12 (X'0C')	Unsuccessful GETMAIN request.
12 (X'0C')		I/O error (The system posted the request for an error).

An I/O error occurs for load display if the drive display has a hardware failure.

If you get return code X'04' or X'0C' on a RESET UCBL operation, when you regain control, register 1 points to the UCB associated with the error.

Chapter 9. Using DFSMSdfp Callable Services

This information describes the DFSMSdfp-related callable services. Callable services reside in SYS1.CSSLIB, the callable system service library. They are invoked from a user program by issuing a CALL statement, accompanied by a parameter identifying the desired service and a list of service-specific arguments and storage areas. The intent is that if you link edit any of the routines described here with your program, the routine continues to execute correctly on future levels of the operating system.

The DFSMSdfp callable services provide 15 callable system services. They can be invoked by the high-level languages supported by z/OS Language Environment® and by assembler language callers.

Note: Nine of these callable services are related to using the character data representation architecture (CDRA) identifiers.They are application program interfaces (APIs) that are needed to consistently and correctly process graphic character data. For detailed information on their use, refer to *Character Data Representation Architecture Reference and Registry.*

These services enable programs written in assembler language or high-level languages to use:

- IECTRKAD to perform conversions and compares of DASD track addresses that contain 16-bit or 28-bit cylinder numbers.
- IGWARLS to query the information in the catalog for record-level sharing (RLS), including the values of the LOG, BWO, and LOGSTREAMID parameters, the VSAM_QUIESCED indicator, the RLS_RECOVERY_TIMESTAMP fields, and whether the sphere requires forward recovery.
- IGWABWO to communicate with DFSMSdfp to retrieve or set various data set–related indicators. Through the use of these indicators, your program can determine if a data set is eligible for backup while it is open for update and, if eligible, what action can or should be taken. See Table 77 on page 330 for an explanation of the indicators.
- IGWASMS to return certain data set attributes for SMS managed data sets. These data set attributes are:
  - SMSDATA—SMS class names, that is, storage class, management class, and data class.
  - DSTYPE—Currently indicates whether the data set is a PDSE-type data set, HFS, or neither.

    Note: If IGWASMS is called for a non-SMS managed data set, zeros are returned for the DSTYPE attribute.
- IGWASYS to determine the version, release, and modification level of DFSMSdfp on your system, and the status of the SMS subsystem.
- IGWLSHR to determine the DFSMSdfp share attributes in use on the current system.

This information describes calling the service from a nonreentrant program written in assembler language. See *z/OS MVS Programming: Assembler Services Reference ABE-HSP* for information on using CALL in a reentrant program. For information on using CALL in programs written in high-level languages, see the applicable language documentation.

Your program can call the DFSMSdfp callable services in either 24- or 31-bit AMODE. The program can be executed in any protection key and in either supervisor or problem state. When you invoke any of the callable services, your program must provide the address of a standard 18 word save area in register 13. The syntax diagrams here show a CALL statement in assembler language. These callable services can be invoked in the following two ways:

- A CALL statement is coded in the invoking application. The callable services IGWASYS, IGWASMS, IGWABWO, IGWLSHR or IGWARLS, among others, are in SYS1.CSSLIB. When link-editing the invoking application, specify SYS1.CSSLIB in the library concatenation.
- The invoking application can issue a LINK or LOAD/CALL to the desired service, IGWASYS, IGWASMS, IGWABWO, IGWLSHR, or IGWARLS. For an example of using a LINK macro, see “Example” on page 325. For an example of using CALL and LOAD/CALL macros, see “Example” on page 324 and “Example” on page 328.

To invoke the callable services write a set of arguments in a specific order on the invocation. The number of arguments associated with each callable service is fixed, and the types of arguments are restricted to 32-bit binary integers (hereafter referred to as integers) and fixed-length EBCDIC character strings. The CALL statement format is described in the following information:

- "Call for DFSMS Level Determination" on page 322
- "Call for Data Set Attribute Retrieval" on page 324
- "Call for Data Set Backup-While-Open Support" on page 326
- "Call for DFSMSdfp Share Attributes" on page 331
- " Call for Record-Level Sharing Query (IGWARLS)" on page 333
- "Call for converting and comparing 28-bit cylinder addresses (IECTRKAD)" on page 336

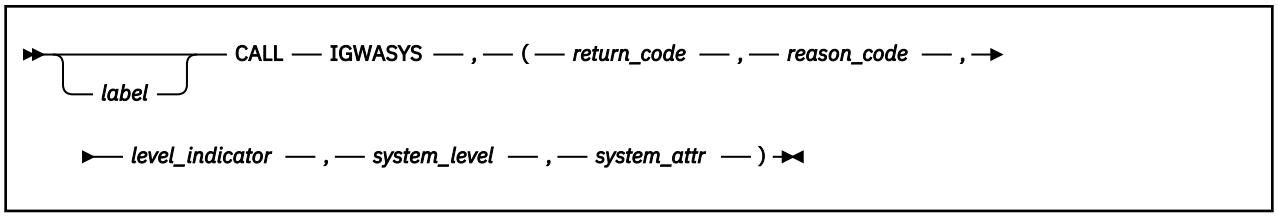
Call for DFSMS Level Determination

The DFSMS level determination call (IGWASYS) returns the version, release and modification levels of DFSMS. It also returns a code number to represent the name of the product that contains DFSMS. These four numbers represent the environment that DFSMS was designed to run in. That level might be earlier than the release for the rest of the operating system.

An alternative technique to determine the level of the system is described in "Determining Level and Name of DFSMS" on page 304.

Format

The format of the system attribute IGWASYS call statement is:



Parameters

return_code
Return code from IGWASYS. The return code is also returned in register 15. For an explanation, see "IGWASYS, IGWASMS, IGWABWO, IGWLSHR Return and Reason Codes" on page 332. This is an output argument that must be defined as an integer.

reason_code
Reason code from the IGWASYS service. The reason code is also returned in register 0. For an explanation, see "IGWASYS, IGWASMS, IGWABWO, IGWLSHR Return and Reason Codes" on page 332. This is an output argument that must be defined as an integer.

level_indicator
The product whose level information is requested. This is an input argument. Define level_indicator as an integer. Code a value of 1 to request the level of MVS/XA DFP Version 2 or MVS/DFP Version 3 or code a 2 to request the level of DFSMS/MVS or later. This value affects what the system returns in system_level, which is described below.

system_level
The product level that is installed on the system invoking the service. This is an output argument and is a four-element array of integers. The array elements consist of the following integers:

- Version number
- Release number
- Modification level

- Special indicator

If you pass a value of 1 for *level_indicator*, DFSMS returns the *system_level* values as 3, 3, 2 and 1. The first three values reflect the fact that DFSMS contains the functions of MVS/DFP Version 3 Release 3 modification level 2. The system sets the fourth value to 1 to indicate that the system is actually at a level higher than 3.3.2. If your program were executing on MVS/DFP 3.3.2, then these four values would have been returned as 3, 3, 2 and 0.

If you pass a value of 2 for *level_indicator*, IGWASYS sets the fourth *system_level* word to a code that represents the product name that IGWASYS is part of. On no level of the system will this word contain a value that is smaller than on a prior release. One of the following values is returned:

1
"DFSMS/MVS". With MVS/ESA SP Version 5, it constituted an MVS/ESA system. With OS/390 MVS™, it was part of OS/390 Version 1 or Version 2. The first three *system_level* words represent DFSMS/MVS.

2
DFSMS is part of OS/390 and not a separate product. The first three *system_level* words show OS/390 Version 2, Release 10, Modification Level 0, or later. Those words represent the OS/390 level that DFSMS was designed to support. You will see these values also on z/OS DFSMS Version 1 Release 1 and 2 because DFSMS was not changed from OS/390 2.10.

3
DFSMS is part of z/OS and not a separate product. The first three system-level words show z/OS Version 1, Release 3, Modification Level 0, or later. Those words represent the z/OS level that the DFSMS was designed to support.

>3
DFSMS is part of hypothetical replacement product after all versions and releases of z/OS. The system never returns this value. It represents IBM's intent in case there is such a product. The other words represent the operating system level that the DFSMS was designed to support.

If the next release of the system does not contain a new level of DFSMS, IGWASYS will continue to return the original DFSMS release. This is to help with diagnosis of configuration problems and service.

The intent is that in any future level of the system, your program can determine whether it is running on a particular level or it is running on some later level. If that is what you wish your program to do, then it is suggested that your program test all four words of system level. First test the fourth word to ensure that it has a value of 1 or greater. Then test the other three words to see whether they designate the appropriate version, release and modification level. It is IBM's intent that if one of the values in the first column of the figure below is smaller than was returned for the corresponding value in the prior release, then one of the values in the second column contains a larger value than was returned in the prior release:

```
Third word (modification level)   Fourth, first or second word
Second word (release)             Fourth or first word
First word (version)              Fourth word
```

system_attr
System attributes are returned in the *system_attr* array. This is an output argument. The array elements are defined as follows:

1
Status of the Storage Management Subsystem. A value of 0 indicates inactive; 1 indicates active.

2–4
Reserved elements; 0 is returned.

Define as a four-element array of integers.

Return Codes

See Table 78 on page 332 for the IGWASYS return and reason codes.

Example

The following example shows a system attribute call using a CALL statement.

```
            .
            .
         CALL  IGWASYS,(RC1,RS1,CODE1,LEVEL,ATTR)   Test pre-DFSMS/MVS
         LTR   R15,R15        Test return code
         BNZ   BADSYS
         CLC   LEVEL,=F'2'      Test for MVS/XA DFP Version 2
         BE    OLDSYS
         MVC   SYSNAME,UNKNAME  Assume name is unknown
         BL    BADSYS           Branch if unknown system
         MVC   SYSNAME,=CL12'MVS/DFP'   Show we are on MVS/DFP
         CLC   LEVEL+12(4),=F'1'        See if after MVS/DFP
         BL    SHOWSYS           Branch if before DFSMS/MVS
         CALL  IGWASYS,(RC1,RS1,CODE2,LEVEL,ATTR)
         LTR   R15,R15          Branch if environment or
         BNZ   BADSYS             system error
         MVC   SYSNAME,UNKNAME      Assume unknown product name
         CLC   LEVEL+12(4),=F'1'    Test for DFSMS/MVS code
         BL    OLDSYS           Branch if unexpected code
         MVC   SYSNAME,=CL12'DFSMS/MVS'  Show product name
         BE    SHOWSYS         Branch if DFSMS/MVS
         MVC   SysName,=CL12'OS/390 DFSMS'  Set assumed new name
         CLC   LEVEL+12(4),=F'3'    Branch if
         BL    SHOWSYS                OS/390
         BH    COPYSYS              Branch if after z/OS name
         MVC   SysName,=CL12'z/OS DFSMS'  Assume early z/OS
*  On z/OS.  Test for early releases.
         CLC   Level(8),=F'1,3'     Branch if
         BL    ShowSys                before z/OS 1.3
*  On z/OS 1.3 or later.  Use the name provided by the system.
CopySys  L     R14,16               Point to CVT
         L     R15,CVTDFA-CVT(,R14) Point to DFA
         L     R14,DFAELNMP-DFA(,R15) Point to element name
         USING DFAELNM,R14
         MVI   SysName,C' '         Blank out name
         MVC   SysName+1(L'SysName-1),SysName
         LH    R15,DFAELNML         Get name length
         CH    R15,MaxLen           Skip one instruction if
         BNH   *+8                    name not too long
         LH    R15,MaxLen           Truncate to our field length
         BCTR  R15,0                Decrement for EX instruction
         EX    R15,MVCName          Copy name from DFA
         DROP  R14
ShowSys  EQU   *     Handle version, release and modification level
         .
         .
RC1      DC    F'0'    Return code
RS1      DC    F'0'    Reason code
CODE1    DC    F'1'    Ask for pre-DFSMS DFP level
CODE2    DC    F'2'    Ask for level of DFSMS, or later
LEVEL    DC    4F'0'   Version, release, Modification Level and code
ATTR     DC    4F'0'   SMS attributes
SYSNAME  DC    CL12'MVS/XA DFP'  Name of product
UNKNAME  DC    (L'SYSNAME)C'?'   Constant for unknown name
MaxLen   DC    Y(L'SysName)      Our maximum allowed length of name
MVCName  MVC   SysName(0),DFAELTXT-DFAELNM(R14)
```

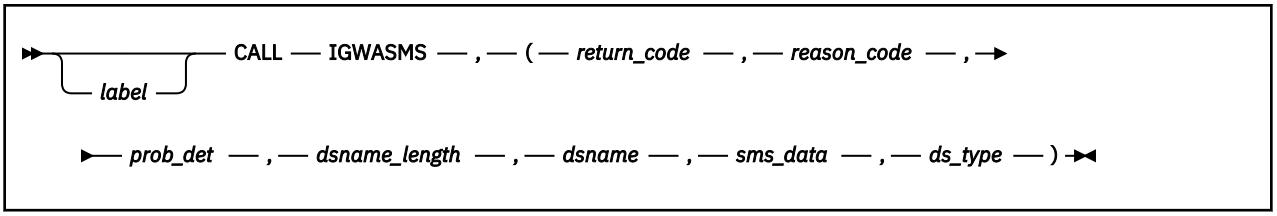
Figure 36. Example of an IGWASYS Call Statement

Call for Data Set Attribute Retrieval

The data set attribute retrieval call (IGWASMS), returns the names of the data set's related SMS classes and whether it is a PDSE, or an HFS data set, or neither.

Format

The format of the data set attribute IGWASMS call statement is:



Parameters

return_code
Return code from IGWASMS. The return code is also returned in register 15. Return codes are explained in "IGWASYS, IGWASMS, IGWABWO, IGWLSHR Return and Reason Codes" on page 332. This is an output argument that must be defined as an integer.

reason_code
Reason code from IGWASMS. The reason code is also returned in register 0. Reason codes are explained in "IGWASYS, IGWASMS, IGWABWO, IGWLSHR Return and Reason Codes" on page 332. This is an output argument that must be defined as an integer.

prob_det
Problem determination data. See "IGWASYS, IGWASMS, IGWABWO, IGWLSHR Return and Reason Codes" on page 332 for more information about problem determination data. This is an output argument that must be defined as a two-element array of integers.

dsname_length
Length, in bytes, of the data set name provided by the caller in *dsname*. The value can be a number from 1 to 44. This is a required input argument that must be defined as an integer.

dsname
Name of the data set on which the IGWASMS service. For VSAM data sets, the cluster name must be specified. This is a required input argument that must be defined as EBCDIC character data of length *dsname_length*.

sms_data
The SMS class names associated with the specified data set returned, left-justified with blanks padded on the right. The array elements are returned in the following circumstances:

- Storage class name, or blanks if the data set is not an SMS data set.
- Management class name, or blanks if the data set has no associated management class.
- Data class name, or blanks if the data set has no associated data class.

This is an output argument that must be defined as a three-element array, where each entry is a 30 (byte) character EBCDIC string.

ds_type
The type of data set, *dsname*, is returned. A value of 1 indicates the data set is a PDSE-type data set. A value of 2 indicates the data set is an HFS-type data set. An HFS data set defines a file system and is not a file within the file system.A value of 0 indicates that it is neither. No other values are currently defined. This is an output argument that must be defined as an integer.

Return Codes

See Table 78 on page 332 for the IGWASMS return and reason codes.

Example

The following example shows sample coding for a data set attribute call using a LINK statement.

```
            .
            .
          LINK EP=IGWASMS,MF=(E,ASMSLIST)
            .
            .
RC2       DC   F'0'
RS2       DC   F'0'
PROB1     DC   2F'0'
DSNLEN1   DC   A(L'DSN1)
DSN1      DC   CL12'THIS.DATASET'
SMSDATA   DC   3CL30' '
DSTYPE    DC   F'0'
ASMSLIST  DC   A(RC2)
          DC   A(RS2)
          DC   A(PROB1)
          DC   A(DSNLEN1)
          DC   A(DSN1)
          DC   A(SMSDATA)
          DC   A(DSTYPE)
```

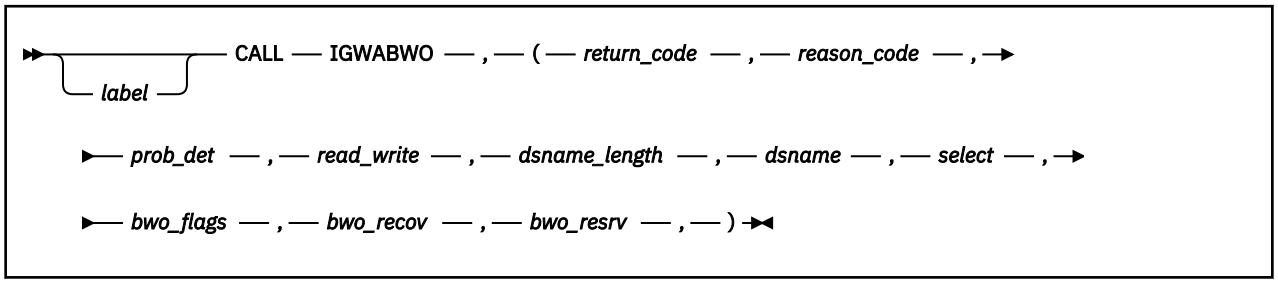
Figure 37. Example of an IGWASMS Call LINK Statement

Call for Data Set Backup-While-Open Support

The data set backup-while-open support call (IGWABWO) communicates with DFSMSdfp to retrieve or set indicators related to taking data set backups while they are open for update.

Format

The format of the IGWABWO callable service is:



Parameters

return_code
Return code from IGWABWO. The return code is also returned in register 15. Return codes are explained in “IGWASYS, IGWASMS, IGWABWO, IGWLSHR Return and Reason Codes” on page 332. This is an output argument that must be defined as an integer.

reason_code
Reason code from IGWABWO. The reason code is also returned in register 0. Reason codes are explained in “IGWASYS, IGWASMS, IGWABWO, IGWLSHR Return and Reason Codes” on page 332. This is an output argument that must be defined as an integer.

prob_det
Problem determination data. See “IGWASYS, IGWASMS, IGWABWO, IGWLSHR Return and Reason Codes” on page 332 for more information about problem determination data. This is an output argument that must be defined as a two-element array of integers.

read_write
Function indicator for this service. A caller-supplied value of 0 indicates that this is a READ-type request for the backup-while-open (BWO) data of the specified data set. A value of 1 indicates a WRITE-type request, that is, an initialization or update of the specified data set's BWO data with the supplied arguments *bwo_flags, bwo_recov,* and *bwo_resrv*. The select argument indicates which arguments are to be processed. This is a required input argument that must be defined as an integer.

dsname_length
Length, in bytes, of the data set name provided by the caller in *dsname*. The value can be a number from 1 to 44. This is a required input argument that must be defined as an integer.

dsname
Name of the data set that the IGWABWO service operates on. Only system-managed VSAM-type data sets are eligible to be backed up while they are open for update. The *dsname* specified must be the base cluster name of a VSAM data set. This is a required input argument that must be defined as EBCDIC character data of length *dsname_length*.

select
Indicates which of the following arguments will be processed. Arguments are specified by selecting the appropriate value. This is a required input argument.

1 to process *bwo_flags*
2 to process *bwo_recov*
3 to process *bwo_flags* and *bwo_recov*

Regardless of how many arguments are to be processed, all three fields (*bwo_flags, bwo_recov*, and the reserved *bwo_resrv* field) must be defined in your program and included in the invocation. Those not selected will receive no value in a READ-type request. The values of those not selected will be ignored in a WRITE-type request.

bwo_flags
This argument is a three-element array, whose elements correspond to the three BWO flags associated with an SMS data set. *bwo_flags* is an output argument for *read_write=0* type requests, and a required input argument for *read_write=1* type requests.

1. The first element is associated with flag, BWO1. 1 is on, 0 is off.
2. This element corresponds to flag, BWO2. 1 is on, 0 is off.
3. This element corresponds to flag, BWO3. 1 is on, 0 is off.

Define as a three-element array of integers.

bwo_recov
This argument is an 8-byte storage area containing the recovery timestamp associated with a data set that is eligible for BWO. *bwo_recov* is an output argument for read_write=0 type requests, and a required input argument for read_write=1 type requests. The format of the timestamp for CICS® VSAM data sets is as follows:

- The first word contains the date in packed decimal format, 0CYYDDDF, where:

  0C
  is the century - 00 represents 19YY, 01 represents 20YY

  YY
  is the last two digits of the year

  DDD
  is the day of the year (Julian date)

  F
  is the sign (F for positive number)

- The second word contains the time in packed decimal format, HHMMSSTF, where:

  HH
  Hours, based on a 24-hour clock

  MM
  Minutes

  SS
  Seconds

  T
  Tenths of a second

F
is the sign (F for positive number)

bwo_resrv
This argument is reserved for future use. While the *bwo_resrv* argument cannot be written or read, it must be defined in your program and included in the invocation. Define as EBCDIC character data of length 16 bytes.

Return Codes

See Table 78 on page 332 for the IGWABWO return and reason codes.

Example

The following example shows sample coding for WRITE and READ-type backup-while-open calls using LOAD and CALL statements.

```
               .
               .

          LOAD EP=IGWABWO
          LR   R2,R0
          CALL (R2),(RC1,RS1,PROB1,RW1,DSNLEN1,DSN1,SEL1,BWOF1,BWOR1,BWRE)
          CALL (R2),(RC2,RS2,PROB2,RW2,DSNLEN2,DSN2,SEL2,BWOF2,BWOR2,BWRE)
               .
               .

*  ARGUMENTS FOR WRITING
RC1       DC   F'0'
RS1       DC   F'0'
PROB1     DC   2F'0'
RW1       DC   F'1'            WRITE
DSNLEN1   DC   F'11'
DSN1      DC   CL44'THAT.VSAM01'
SEL1      DC   F'3'            WRITE BWO_FLAGS AND BWO_RECOV
BWOF1     DC   F'0'
          DC   F'1'
          DC   F'0'
BWOR1     DS   0F
          DC   X'0096137F'     INPUT DATE IN 0CYYDDDF FORMAT
          DC   X'1045301F'     INPUT TIME IN HHMMSSTF FORMAT
BWRE1     DC   CL16' '
*  ARGUMENTS FOR READING
RC2       DC   F'0'
RS2       DC   F'0'
PROB2     DC   2F'0'
RW2       DC   F'0'            READ
DSNLEN2   DC   F'11'
DSN2      DC   CL44'THAT.VSAM01'
SEL2      DC   F'3'            READ BWO_FLAGS AND BWO_RECOV
BWOF2     DC   3F'0'
BWOR2     DC   CL8' '
BWRE2     DC   CL16' '
*
```

Figure 38. Example of IGWABWO Using LOAD and CALL Statements

Using the Backup-While-Open Facility

The following information describes the usage of the backup-while-open (BWO) facility. BWO flags and the BWO recovery field can be retrieved or updated using the IGWABWO service described in “Call for Data Set Backup-While-Open Support” on page 326. The BWO indicators are described in Table 77 on page 330.

For environments that require high-availability, it might not be possible or desirable to stop or quiesce an application to produce consistent backup copies of the application's data sets.

For these environments DFSMSdfp provides support to allow SMS-managed VSAM data sets that are open for output to be backed up. The support is only useful for applications (such as database systems) that can recover a restored database to a point of consistency. This is typically done from a log (forward

recovery log) maintained by the application that contains record images of all changed (added, deleted, or updated) records. These images can then be reapplied to a backup copy of the database, logically recreating the status of the database at a particular point in time.

The support provided by BWO might not be necessary for online applications that can quiesce the database data sets to ensure no output or update activity against the data set while the backup is in progress. Quiescing a data set in this context means the data set is closed and unallocated.

The following discussion of the operation of this support uses these terms:

- Database manager-the application that controls access to the data sets to be processed. In order for BWO support to be effective, the database manager must have some logging facility to allow point-in-time reconstruction of a database.
- Backup manager-the applications or products that perform the backup and restore functions, such as DFSMShsm and DFSMSdss.
- Recovery manager-the component that manages the inventory of recovery logs and applies the changes from the appropriate log(s) to the restored data set.

The following paragraphs describe the relationships between the BWO support and a user of BWO. Refer to Table 77 on page 330 for the various states of the BWO flags in the following discussion. The BWO indicators are retained in the catalog. The BWO flag states are set or reset using the DFSMSdfp callable system service IGWABWO.

- At initial allocation (IDCAMS DEFINE, IDCAMS or TSO ALLOCATE, JCL and dynamic allocation), the data set is not enabled for BWO (default, BWO flag state is 000).
- The database manager should check the BWO flags prior to opening the data set to ensure it is not downlevel (BWO flag state is 101 or 001). If the data set is downlevel, the recovery manager must be used to apply log changes to the data set.
- The database manager must set flag BWO1 (BWO flag state 100) on for each data set that is allowed to be backed up while open for output. This authorizes the backup manager to initiate backups without serializing the data set, whether or not it is being accessed by the application.
- The backup manager must retrieve the BWO flags prior to the start of the backup. If BWO1 is on, then a backup can be taken without any serialization; otherwise, normal data set serialization must be performed by the backup manager.

  When the backup completes, the backup manager must retrieve the BWO flags again. If the BWO flag state has changed, then at some point during the backup an action occurred that prevented creation of a valid backup. The backup manager should discard the backup just created.
- When the data set needs to be recovered, it is first restored using the backup manager. Data sets are serialized during restore to prevent applications from accessing them. The backup manager must set the BWO flags at the completion of the restore to indicate whether the restore was done using a backup copy that was created with or without serialization.
  - If the backup was taken without serialization, the BWO flags must be set to 101.
  - If the backup was taken with serialization, the BWO flags must be set to 000.

  In either case, the application administrator should decide whether or not to apply recovery logs.

  The database manager should not allow access to the data set until the recovery manager has completed processing.
- The recovery manager should change the BWO flags to 001 before opening the data set, apply the logs, and then set the flags to 000 to indicate that the data set has been processed and is consistent. The database manager can then reestablish normal access to the data set (thus opening the data set for use).
- For a data set that is enabled for BWO, in certain instances the system prevents starting a backup copy without serialization (for example, during a CI/CA split) by setting the BWO flags to 010. This indicates a backup should not be started without serialization (BWO1 off), and that a backup that is currently in process should be considered invalid. When the condition that prevented the starting of a backup is

ended, the system resets the BWO flags to 110. This indicates that a backup can now be started without serialization, and that any backup in progress should be discarded.

- If the database data sets are accessed by batch programs (when the database manager is not accessing the data sets) that do not create forward recovery logs, the database manager should clear the BWO1 flag and set the BWO3 flag at close (that is, OX1). The setting of the BWO2 flag should not be changed. If the backup manager discovers this BWO state at the end of backup without serialization, the backup is not valid and should be discarded. The backup manager can start a backup with serialization if the BWO flag state is 011, but the flags should be reset to 000.

Note:

1. Since backups can result in heavy I/O activity, you might want to take backups during the time of least activity against the data set to avoid affecting the application response time.
2. The BWO flags are not locked between reading and setting them with IGWABWO. The application is responsible for providing serialization when the settings of the flags are changed.
3. KSDS data sets require special consideration. Inserts and updates can result in control interval (CI) and control area (CA) splits. Backups taken without serialization during CI and CA splits are discarded by the IBM products to prevent missing or duplicate records in the backup copies.

   The frequency of CI and CA splits depends on the insert activity, the update activity that increases the lengths of records, and the amount of free space in CI and CA. For KSDS data sets that are BWO enabled, run backups during periods of low inserts and updates or ensure adequate free space in control intervals and control areas.
4. Backups without serialization can be taken (on data sets defined with share option 1 or 2) when:
   - The database contains alternate indices in the upgrade set.
   - The database is accessed by pathnames, or with *ddname* or *dsname* sharing.
5. Data sets can be opened regardless of the setting of the BWO flags. It is the database manager's responsibility to determine whether the contents of the data set are consistent.
6. The database manager can use the BWO recovery field to store information (such as the log RBA or log timestamp) for the recovery manager to use in locating the appropriate recovery logs.
7. BWO concept only applies to logical data set backup/restore. It does not apply to physical data set backup/restore or full volume dump/restore.

Table 77 on page 330 describes the meaning of each of the BWO (BWO1, BWO2, and BWO3) indicators.

Table 77. Backup-While-Open Indicators

BWO Setting	Description
000	Data set does not support backup without serialization.
001	Forward recovery in progress. Reset to 000 after recovery.
010	A CI/CA split for a BWO data set is in progress. Do not start backup. If backup in progress, discard at end. This state can exist at open if the database manager abended during split. Database action depends on database manager. The data set might need restore and forward recovery or backout of changes if AIXs are present.
011	The database manager closed a BWO data set and a CI/CA split had occurred when it was previously open. Backup manager should reset it to 000 and serialize to back up, not a BWO candidate. The database manager should change this state to 110 at open.
100	A BWO data set has been opened by the database manager. Back up without serialization.
101	Data set has been restored and requires forward recovery before it can be used. Reset it to 001 before forward recovery.

Table 77. Backup-While-Open Indicators (continued)

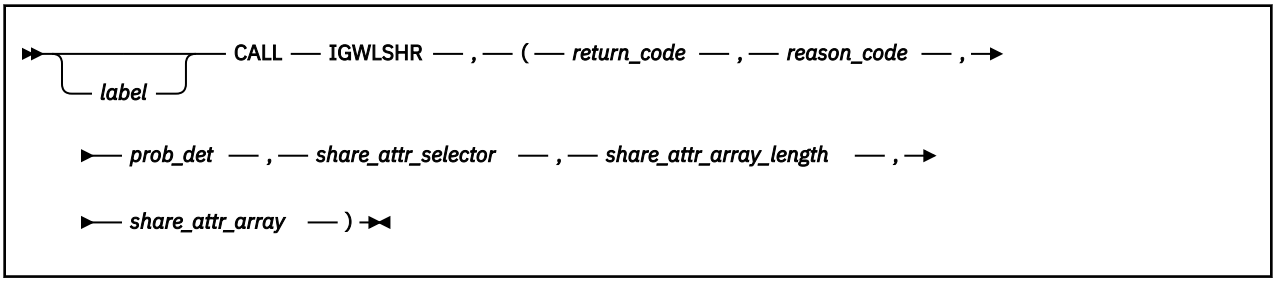
BWO Setting	Description
110	A CI/CA split has occurred and completed on a BWO data set. This state can exist at open if the database manager abended. Back up without serialization. Reset it to 100 before backup.
111	An invalid state.

Call for DFSMSdfp Share Attributes

The DFSMSdfp share attributes call (IGWLSHR) returns the DFSMSdfp share attributes currently in use.

Format

The format of the IGWLSHR callable service is:



Parameters

return_code
Return code from IGWLSHR. The return code is also returned in register 15. Return codes are explained in "IGWASYS, IGWASMS, IGWABWO, IGWLSHR Return and Reason Codes" on page 332. This is an output argument that must be defined as an integer.

reason_code
Reason code from IGWLSHR. The reason code is also returned in register 0. Reason codes are explained in "IGWASYS, IGWASMS, IGWABWO, IGWLSHR Return and Reason Codes" on page 332. This is an output argument that must be defined as an integer.

prob_det
Problem determination data. See "IGWASYS, IGWASMS, IGWABWO, IGWLSHR Return and Reason Codes" on page 332 for more information about problem determination data. This is an output argument that must be defined as a two-element array of integers.

share_attr_selector
Use to specify which DFSMSdfp share attributes are requested. Code a value of 1 to request the PDSE sharing protocol attributes. This is a required input argument that must be defined as an integer.

share_attr_array_length
Use to specify the size of the share attributes array. This length represents the number of array elements in the share_attr_array. Code the value required for the chosen *share_attr_selector* value. The *share_attr_array_length* must minimally be 1 when the share_attr_selector is 1. This is a required input argument that must be defined as an integer.

share_attr_array
Returns DFSMSdfp share attributes. This is an output argument. Define as an array of integers of length *share_attr_array_length*, where *share_attr_array_length* is the length required for the chosen *share_attr_selector* value. The array elements for the *share_attr_selector* value are returned as follows:

1
Status of PDSE sharing protocol. A value of 0 indicates PDSE support is unavailable; that is, the system supports the call, but SMS PDSE support is not active. If the system does not support this call (as with a previous release), then a return code of 36 is returned.

A value of 1 indicates normal PDSE sharing protocol in use. A value of 2 indicates extended PDSE sharing protocol in use.

2–4
Reserved elements; 0 is returned.

Return Codes

See Table 78 on page 332 for the IGWLSHR return and reason codes.

IGWASYS, IGWASMS, IGWABWO, IGWLSHR Return and Reason Codes

When IGWASYS, IGWASMS, IGWABWO, or IGWLSHR returns control to the calling program, it provides both a return code and a reason code. IGWASMS and IGWABWO can return additional data useful for problem determination in the *prob_det* array. IGWLSHR can return additional data regarding *share_attr_selector* and *share_attr_array_length* arguments. The following table identifies return code and reason code combinations, tells what each means, explains what and when additional problem determination data is returned, and recommends what action should be taken.

Table 78. IGWASYS, IGWASMS, IGWABWO, IGWLSHR Return and Reason Codes

Return Code Dec (Hex)	Reason Code Dec (Hex)	Description
0 (0)	0 (0)	The operation was successful.
4 (4)	4 (4)	The operation was successful, but the *bwo_recov* argument has no valid value for the data set specified in *dsname*. This is because it was created under DFP 3.1.0, and no *bwo_recov* has been added to the data set. Add *bwo_recov* to the data set as appropriate.
8 (8)	4 (4)	An invalid *dsname_length* or *share_attr_selector* was specified. Correct the argument and retry the request.
8 (8)	8 (8)	An invalid *dsname* of blanks or invalid *share_attr_array_length* was specified. Correct the argument and retry the request.
8 (8)	12 (C)	An invalid *read_write* was specified. A value of 0 or 1 must be supplied. Correct the *read_write* argument and retry the request.
8 (8)	16 (10)	The values supplied for *bwo_flags* are not valid. BWO1, BWO2, and BWO3 must have a value of either 0 or 1. Correct the *bwo_flags* argument and retry the request.
8 (8)	20 (14)	BWO is only supported for VSAM-type data sets. The name specified was not a VSAM cluster name. Specify the name of a VSAM cluster in the *dsname* argument and retry the request.
8 (8)	24 (18)	An invalid *level_indicator* was specified. Correct the *level_indicator* argument and retry the request.
8 (8)	28 (1C)	An invalid *select* argument was specified. A value between 1 and 3 must be specified. Correct the *select* argument and retry the request.
8 (8)	32 (20)	The data set specified in *dsname* is not an SMS-managed data set. Correct the *dsname* argument and retry the request.

Table 78. IGWASYS, IGWASMS, IGWABWO, IGWLSHR Return and Reason Codes (continued)

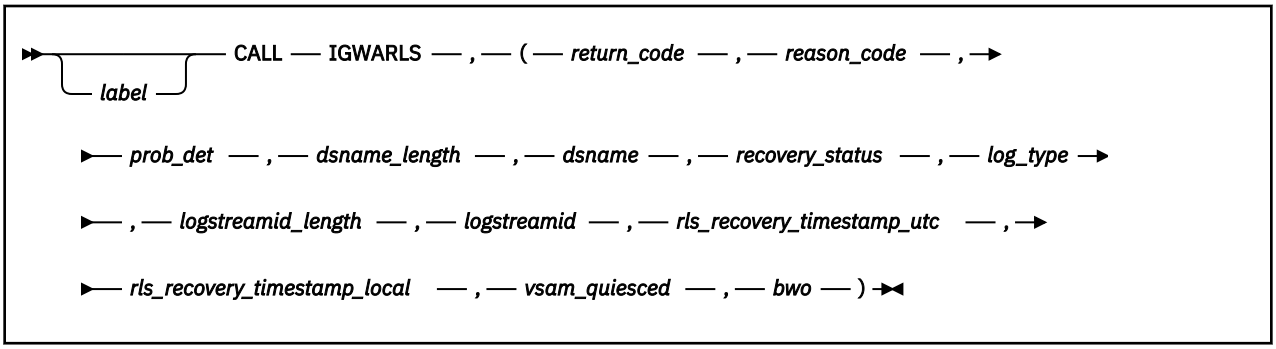
Return Code Dec (Hex)	Reason Code Dec (Hex)	Description
12 (C)	8 (8)	There is insufficient virtual storage to process the request. Free some virtual storage and retry the request. If the condition persists, contact IBM for programming assistance.
12 (C)	12 (C)	The data set specified in *dsname* could not be found. Verify that the data set exists and has been correctly specified in *dsname*.
12 (C)	16 (10)	The data set specified in *dsname* is currently in MIGRATE status.
16 (10)	4 (4)	An error occurred on a call to catalog management. The catalog return code is in the first element of *prob_det* and the catalog reason code is in the second element of *prob_det*. See message IDC3009I for an explanation of the catalog return code and reason code. A catalog management return code of 8 indicates that the specified data set was not found. If you get this return code, correct *dsname* and retry the request.
20 (14)	4 (4)	A system error occurred during IGWASYS/SMS/BWO or IGWLSHR processing. The elements of *prob_det* contain additional diagnostic data. Contact IBM for programming assistance and provide them with the IGWASYS/SMS/BWO *return_code, reason_code,* and *prob_det* values.
36 (24)	4 (4)	Linkage cannot be established to the IGWLSHR service module or to IGWASYS/SMS/BWO service modules, IGWAMCS1 and IGWAMCS2. Either the wrong level of the operating system is being used, or the callable system service library, SYS1.CSSLIB, is missing the required services. Contact your installation system programmer for assistance.

Call for Record-Level Sharing Query (IGWARLS)

IGWARLS is used to query the information in the catalog for record-level sharing.

Format

The format of the IGWARLS call statement is:



Parameters

return_code
Return code from IGWARLS. The return code is also returned in register 15. Return codes are explained in “Return Codes” on page 335. This is an output argument. Define *return_code* as an integer.

reason_code
Reason code from IGWARLS. The reason code is also returned in register 0. Reason codes are explained in "Return Codes" on page 335. This is an output argument. Define *reason_code* as an integer.

prob_det
Problem determination data. See "Return Codes" on page 335 for more information about problem determination data. This is an output argument. Define *prob_det* as a two element array of integers.

dsname_length
Length, in bytes, of the data set name provided by the caller in *dsname*. The value can be a number from 1 to 44. This is a required input argument. Define *dsname_length* as an integer.

dsname
Name of the base cluster that the IGWARLS service will operate on. This is a required input argument. Define dsname as EBCDIC character data of length *dsname_length*.

recovery_status
Returns an indication as to whether the sphere is marked as requiring forward recovery.

- 0 - RLS recovery required is not pending for the VSAM sphere.
- 1 - RLS recovery required is pending.

recovery_status is an output argument that is defined as an integer.

log_type
The specification of the LOG= parameter on DEFINE CLUSTER is returned.

- 1 - LOG parameter undefined
- 2 - LOG=NONE
- 3 - LOG=UNDO
- 4 - LOG=ALL

log_type is an output argument that is defined as an integer.

logstreamid_length
Length, in bytes, of the LOGSTREAMID field. This is a required input parameter. Define logstreamid_length as an integer. The value of logstreamid_length should be at least 26 bytes.

logstreamid
The specification of the LOGSTREAMID on DEFINE CLUSTER is returned. If the parameter is undefined, blanks are returned. The caller can determine the size of the returned LOGSTREAMID field by scanning from right to left looking for a non blank character or until the entire field has been scanned. *logstreamid* is an output argument that is defined as an EBCDIC character data field of length logstreamid_length.

rls_recovery_timestamp_utc
An output argument which represents the UTC time (formally known as GMT) of the dump/copy in STCK format. The field is defined as an eight-byte unsigned integer. Your program might regard this field as either an integer or a character string.

rls_recovery_timestamp_local
An output argument which represents the local time of the dump/copy in STCK format. The field is defined as an eight-byte unsigned integer. Your program might regard this field as either an integer or a character string.

vsam_quiesced
An output argument which indicates whether the sphere is marked as VSAM_QUIESCED.

- 0 - The sphere is not marked VSAM_QUIESCED.
- 1 - The sphere is marked VSAM_QUIESCED.

vsam_quiesced is an output argument, defined as an integer.

bwo
An output argument which indicates whether the value of the BWO parameter on define cluster.

- 1 - BWO parameter is undefined.
- 2 - BWO = TYPECICS processing allowed.
- 3 - BWO = NO. BWO processing is not allowed.
- 4 - BWO = TYPEIMS processing allowed.
- 5 - BWO = TYPEOTHER processing allowed.

Return Codes

When IGWARLS returns control to the calling program, it provides both a return code and a reason code. IGWARLS can return additional data useful for problem determination in the *prob_det* array. Table 79 on page 335 identifies return code and reason code combinations, tells what each means, explains what and when additional problem determination data is returned, and recommends what action should be taken.

Table 79. IGWARLS Return and Reason Codes

Return Code Dec (Hex)	Reason Code Dec (Hex)	Description
0 (0)	0 (0)	The operation was successful.
8 (8)	4 (4)	An invalid *dsname_length* was specified. Correct the *dsname_length* argument and retry the request.
8 (8)	8 (8)	An invalid *dsname* of blanks was specified. Correct the *dsname* argument and retry the request.
8 (8)	20 (14)	IGWARLS is only supported for VSAM data sets. The name specified was not the name of the base cluster. Specify the name of the base cluster in the dsname argument and retry the request.
8 (8)	32 (20)	The data set specified in *dsname* is not an SMS managed data set. Correct the *dsname* argument and retry the request.
8 (8)	40 (28)	For IGWARLS, the logstreamid_length specified was invalid (<=0) or was not large enough to return the requested logstreamid. Correct the *logstreamid_length* argument and retry the request.
12 (C)	8 (8)	There is insufficient virtual storage to process the request. Retry the request. If the condition persists, contact IBM for programming assistance.
12 (C)	12 (C)	The data set specified in *dsname* could not be found. Verify that the data set exists and has been correctly specified in *dsname*.
12 (C)	14 (E)	The data set specified in *dsname* was found in the catalog but its attributes were not available. Verify that the data set has been correctly specified in *dsname*.
12 (C)	16 (10)	Cannot access the data set that is specified in *dsname*. The data set has been HSM migrated. HRECALL the data set and retry the request.
16 (10)	4 (4)	An error occurred on a call to catalog management. The catalog return code is in the first element of *prob_det* and the catalog reason code is in the second element of *prob_det*. See message IDC3009I for an explanation of the catalog return code and reason code. A catalog management return code of 8 indicates that the specified data set was not found. If you get this return code, correct *dsname* and retry the request.
20 (14)	4 (4)	A system error occurred during IGWARLS processing. The elements of *prob_det* contain additional diagnostic data. Contact IBM for programming assistance and provide them with the IGWARLS *return_code, reason_code,* and *prob_det* values.

Table 79. IGWARLS Return and Reason Codes (continued)

Return Code Dec (Hex)	Reason Code Dec (Hex)	Description
36 (24)	4 (4)	Linkage cannot be established to the IGWRLS service module, IGWAMCS4. Either the wrong level of the operating system is being used, or the callable system service library, SYS1.CSSLIB, is missing the required services. Contact your installation system programmer for assistance.

Example

The following example shows the RLS query call using LOAD and CALL statements:

```
            .
            .

               LOAD  EP=IGWARLS
               LR    R9,R0
               CALL  (R9),(RC1,RS1,PROB1,DSNLEN1,DSN1,RECSTAT,LOGTYPE,      X
                          LOGSTRML,LOGSTRM,RCVTMG,RCVTML,VSAMQUIS,BWO)

         RC1       DC    F'0'
         RS1       DC    F'0'
         PROB1     DC    2F'0'
         DSNLEN1   DC    A(L'DSN1)
         DSN1      DC    CL12'BASE.CLUSTER'
         RECSTAT   DC    F'0'
         LOGTYPE   DC    F'0'
         LOGSTRML  DC    A(L'LOGSTRM)
         LOGSTRM   DC    CL26' '
         RCVTMG    DC    XL8'00'
         RCVTML    DC    XL8'00'
         VSAMQUIS  DC    F'0'
         BWO       DC    F'0'
```

Figure 39. Example of the IGWARLS Query Call Using LOAD and CALL Statements

Call for converting and comparing 28-bit cylinder addresses (IECTRKAD)

IECTRKAD is a callable service to perform conversions and compares of 28-bit cylinder addresses. The track addresses are in the form *CCCCcccH*, where *CCCC* is the 16 low order bits of the cylinder number and *ccc* is the 12 high order bits of it.

Cobol, PL/I, and C programs can call IECTRKAD without having to write assembler routines to invoke TRKADDR.

The caller requirements of IECTRKAD are:

- Register 1 contains an address to a parameter list. Register 0 is not used. Register 13 points to a standard register 18-word save area and registers 14 and 15 have their standard usage.
- Calling program can be in either 24-or 32-bit addressing mode
- Calling program can be executing in any protection key and in either supervisor or problem state

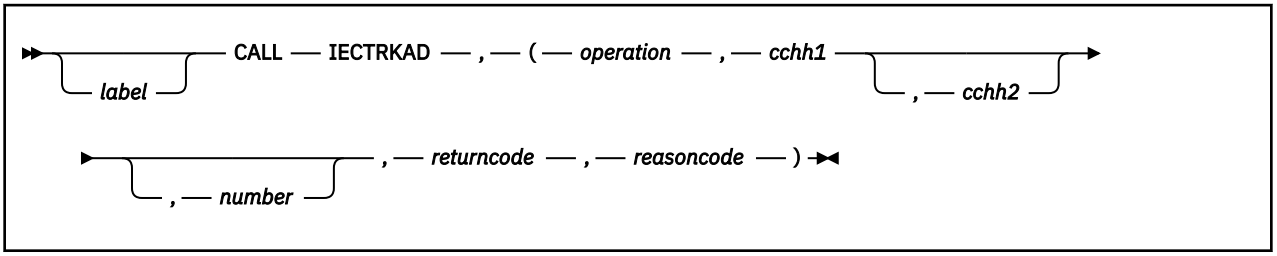
The called routine, IECTRKAD, has the following characteristics:

- No executable macro other than CALL and LINK is provided to call IECTRKAD.
- The called routine resides in SYS1.CSSLIB and is shipped in distribution library ACSSLIB.
- User program can link edit with the called routine to invoke IECTRKAD.
- User program can use the LINK macro or the LOAD and CALL macros to invoke IECTRKAD.
- IECTRKAD processing will use the equivalent TRKADDR function and pass the result back accordingly.

- IECTRKAD is release independent; if you link or bind IECTRKAD with your program, it will run on earlier or later releases of z/OS. This includes releases before the first availability of IECTRKAD.

Format

You can adapt the assembler language syntax shown here to your computer language such as C, COBOL or PL/I:



Parameters

Where the parameters are:

operation
Specifies the operation to perform. This parameter is the name of a field that contains character data of length 10. The allowable values are:

ABSTOREL
calculates the relative track number on the volume from the passed *cchh1* track address.

COMPARE
compares the two track addresses passed in *cchh1* and *cchh2*.

EXTRACTCYL
extracts the 28-bit cylinder number from the passed *cchh1* track address.

EXTRACTTRK
extracts the 4-bit track number from the passed *cchh1* track address.

NEXTTRACK
increments the track address by one track and increments the cylinder number if necessary from the passed *cchh1* track address.

NORMALIZE
reverses the 16-bit and 12-bit portions of the cylinder number from the passed *cchh1* track address. The *CCCCcccH* becomes *cccCCCCH*. This could be used to subsequently perform unsigned comparisons of track addresses.

NORMTOABS
reverses the 12-bit and 16-bit portions of the cylinder from the track address passed in number. The *cccCCCCH* becomes *CCCCcccH*. Use this to convert a normalized track address to an absolute 28-bit cylinder address.

RELTOABS
converts a relative track number, passed in *number*, to a 28-bit cylinder address.

SETCYL
converts a relative cylinder number, passed in *number*, to a 28-bit cylinder address and sets the head portion to zero.

If you request an operation that is less than 10 characters, it must be padded on the right with blanks.

cchh1
Required parameter, *cchh1*, is a four-byte area containing a track address that nominally is in the form of CCHH. For all functions except RELTOABS and SETCYL this is input to the called routine. For RELTOABS, SETCYL, and NORMTOABS this is output from the called routine.

cchh2
Optional positional parameter, *cchh2*, is a four-byte area whose meaning depends on the operation specified by the first parameter. For all functions except COMPARE and NEXTTRACK this parameter is ignored.

For COMPARE processing, *cchh2* contains the track address that is to be compared to the first *cchh1* parameter.

For NEXTTRACK processing, *cchh2*, is the four-byte output area to contain the track address (CCHH) of the next logical track on the volume. If the input track number is 0 to 13, the output cylinder number will be the same and the output track number will be one greater than the input track number. If the input track number is 14, the output cylinder will be one higher than the cylinder number in *cchh1* and the output track number will be 0. The called routine does not check for numeric overflow.

number
Optional positional parameter, *number*, is a four-byte integer whose meaning depends on the operation specified by the first parameter.

For ABSTOREL processing, *number* is the output area that will contain the relative track number on the volume.

For COMPARE processing, *number* is the output area that will contain the result of the comparison. Zero means the inputs are equal. Negative one means the first one is lower. Positive one means the first one is higher.

For EXTRACTCYL processing, *number* is the output area that will contain the 28-bit cylinder number with the four high order bits set to zero.

For EXTRACTTRK processing, *number* is the output area that will contain the four-bit track number with the 28 high order bits set to zero.

For NEXTTRACK processing, *number* is an ignored parameter. It can be 0 or any valid virtual address. Not checked by the called routine.

For NORMALIZE processing, *number* is the output area that will contain the normalized version of the input *CCHH*. The *CCCCcccH* becomes *cccCCCCH*. The high order 28-bits are the cylinder number and the low order four bits are the track number. This allows your code to do a more efficient comparison of one track address with many track addresses. Normalize each and do unsigned comparisons.

For NORMTOABS processing, *number* is the input area that contains the normalized track address to be converted. The *cccCCCCH* becomes *CCCCcccH*. Use this to convert a normalized track address to an absolute 28-bit cylinder address.

For RELTOABS processing, *number* is the input area that contains the relative track number on the volume. For example the first two tracks on the second cylinder (cylinder 1) have relative track numbers of 15 and 16. The called routine converts the relative track number to a 28-bit nonlinear cylinder address with a 4-bit head value in the low order four bits in the *cchh1* output area.

For SETCYL processing, *number* is the input area that contains the cylinder number on the volume in the low order 28-bits with the four high order bits set to zero. The called routine splits the 28-bits into the high order 12-bits and low order 16-bits, reverses them in the output field, *cchh1*, with the cylinder address in the high order 28-bits and the low order four bits to zero

returncode
returncode is a 4-byte integer that contains the return code from IECTRKAD processing. *returncode* is also returned in register 15. The return codes that could be set include the following:

0
Successful.

4
Successful, but with exceptions

8
Request was unsuccessful because of invalid or incorrect input. Refer to the*reasoncode* for more detailed information

reasoncode
reasoncode is a 4-byte integer associated with a specific return code. *reasoncode* is returned in Register 0.

No reason codes are supplied for return code 4:

The following reason codes are supplied for return code 8:

4
The H portion of a track address is not valid on a NEXTTRACK operation

8
The caller specified an invalid operation (the first parameter is not one of the supported operations)

12
The address of a required parameter is zero.

Character Data Representation Architecture (CDRA) APIs

The following CDRA APIs are included in the DFSMS product library. For more detailed description of both the APIs and their use, see Character Data Representation Architecture Reference (www.ibm.com/downloads/cas/G01BQVRV).

API
Description

CDRGESP
Get Encoding Scheme, Character Set, and Code Page Elements

CDRSCSP
Get Short Form (CCSID) from Specified ES (CS, CP)

CDRGESE
Get Encoding Scheme Element and its Subelements

CDRGCTL
Get Control Function Definition

CDRSMXC
Get Short Form (CCSID) with maximal CS for Specified ES, CP

CDRMSCI
Multiple-Step Convert Initialize

CDRMSCP
Multiple-Step Convert Perform

CDRMSCC
Multiple-Step Convert Clean Up

CDRXSRF
Extract Status and Reason Codes from Feedback Code

Chapter 10. Using the DESERV Exit

The DESERV exit is designed to support those programs which, before the binder, used SVC screening or replacement of the SVC table to trap SVC 21 (STOW) to monitor STOWs. DESERV provides an equivalent function to that obtained by SVC Screening or replacing the SVC table entry for SVC 21 (or SVC 12, BLDL). The DESERV exit can be used along with the SVC screening and SVCUPDTE facilities to monitor accesses and updates to PDS and PDSE directories.

The system calls the DESERV GET function when the following occurs:

- The binder is used to bind a program object or a load module and it is searching for member names to be included. (Note the linkage editor does not use DESERV).
- The system is searching for modules to load into storage while processing the ATTACH, LINK, LOAD, or XCTL functions.

In these situations, the system uses DESERV GET rather than issuing BLDL. However, in some situations DESERV GET issues BLDL to perform the directory search (the BLDL function does not issue DESERV calls). The system calls the DESERV PUT function when the following occurs:

- The binder is creating a program object in a PDSE (note the linkage editor does not use DESERV, nor does the binder use DESERV PUT when creating a load module in a PDS).
- An IEBCOPY job is loading a program object from an IEBCOPY unloaded data set.
- An IEBCOPY job is copying a member to a PDSE where one of the member‘s names is greater than 63 bytes long.

In these situations, the system uses DESERV PUT rather than issuing STOW. The DESERV PUT and STOW code do not interact. STOW does not issue DESERV PUT nor does DESERV PUT issue STOW.

Currently, the system does not use the RENAME or UPDATE functions. SMP/E is the only known user of the DELETE function. The rename, update, and delete functions and the STOW code do not interact.

With SVC screening, an SVC screen table is associated with a task control block (TCB). The table marks specific SVC numbers as not valid. The table also defines the address of a routine that gets control when an SVC that is not valid is issued. Then, it is possible for the screen routine to inhibit the function, perform the function itself, or temporarily disable the SVC screening and reissue the SVC. This technique provides a front and back end mechanism for SVC routines.

With SVCUPDTE, an application can dynamically replace or delete SVC table entries for the system or obtain the SVC number of a routine at a specified entry point. One specific use of the replace function of SVCUPDTE would use a scenario like the following to replace an IBM supplied SVC routine.

1. Extract the SVC entry for SVC 18 (BLDL) from the SVC table.
2. Issue SVCUPDTE to install the vendor's version of the BLDL function.
3. When an SVC 18 is issued, the vendor's BLDL module gets control.
4. The vendor's BLDL either performs the function and returns to the caller, or branch enters the IBM supplied BLDL code whose address was obtained earlier from the SVC table.

For more detail and explanation of the DESERV functions, see *z/OS DFSMS Using Data Sets*; for their macros, see *z/OS DFSMS Macro Instructions for Data Sets*.

DESERV provides a task level exit for an interface that is similar to SVC screening for the SVC routines BLDL and STOW. DESERV also provides a global exit for an interface that is similar to the SVCUPDTE replace option. For more information on using SVC Screening and SVCUPDTE, refer to *z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO* and *z/OS MVS Programming: Authorized Assembler Services Guide*.

Note: The DESERV EXIT has two calls. The information returned from the function requested is available when the second call to the exit is executed.

Task Level Exit

The task level DESERV exit can be established for any TCB following the LPA's initialization. Once established for a task, the DESERV exit is given control for the DESERV functions that are issued under the TCB. The exit entry point is given control twice for each GET, PUT, RENAME, DELETE, or UPDATE invocation; once before DESERV executes and once immediately before the return from the DESERV function. The parameters passed to the exit indicate whether this call happens before or after the DESERV function executes.

If a DESERV FUNC=GET call is made for which there are no PDSEs in the concatenation, the DESERV code issues BLDL to search for the requested names. The task level exit is called in this case. However, if SVC screening is also active, the exit might perform one of the following tasks:

- Not process this DESERV call (that is, lets the SVC screen routine process the BLDL request) or
- Process this DESERV call with the pre- and post-processing exits, but also disable the SVC screening for BLDL in the pre-processing exit and enable the BLDL screening in the post-processing exit. A parameter passed to the exit indicates whether a BLDL will be or has been issued.

Just as with the SVC screening facility, the DESERV task level exit function enables the user to indicate that the exit should be propagated to subsequently attached tasks.

The first DESERV call for a task searches the TCB chain for a DESERV task level exit routine with propagate specified. DESERV searches the TCB chain following the originating TCB pointer (TCBOTC). If none exists, the task is marked to indicate that no DESERV task level exit exists. Therefore, for the propagate option to work, the exit routine must be established before issuing the ATTACH macro. This is roughly consistent with implementing SVC screening. The difference is that the SVC screening table is propagated at the time of the ATTACH, while the DESERV exit might not be propagated at ATTACH (for example, if the attached program is found in the job pack queue, no directory search (that is, DESERV GET) is done to find the module).

Global Exit

The global DESERV exit can be established for the system following the initialization of LPA. When establishing a global exit, obtain the DST (DESERV Screen Table) storage in common storage. The DST is used to identify a DESERV exit. Once established, the DESERV exit gets control anytime DESERV GET, PUT, RENAME, DELETE, or UPDATE functions are called. The exit entry point is given control twice for each invocation, once before DESERV is executed, and once immediately after the return from the DESERV function. The call to the exit indicates whether this call is before or after DESERV executes. The global exit routine must reside in commonly addressable storage.

If a DESERV FUNC=GET call is made for which there are no PDSEs in the concatenation, the DESERV code issues BLDL to search for the requested names. The global exit is called in this case. If both the SVCUPDTE facility and the DESERV global exit are used, before implementing the global DESERV exit consider the interactions of the DESERV global exit and the routine that is given control when the SVC is issued.

Interactions Between the Task Level and Global Exits

If both task level and global DESERV exits have been defined, there is a prescribed calling sequence. The task level exit is called first. If the task level exit indicates that the DESERV function should be terminated (via a return code 4 from the exit), DESERV returns immediately to its caller. However, the global exit is given control when the task level exit returned with return code 0. The global exit can indicate (via return code 4) that control should pass back immediately to the DESERV caller. In this case before returning to the DESERV caller, the task level exit is given control indicating that the DESERV function is complete. After returning from the task level exit, DESERV returns to the caller.

If the global exit returns with return code 0, the DESERV function executes, making the post-processing exit calls. The post processing exit calls are made first to the global exit and second to the task level processing exit. This sequence (the reverse of the pre-processing exit sequence) is chosen to simulate the

return sequence that would have been seen if both SVC screen routine and updated SVC routine were in place. The following diagram illustrates the exit routine call sequence:

```
DESERV GET, PUT, RENAME, DELETE or UPDATE is issued
  enter DESERV GET, PUT, RENAME, DELETE, or UPDATE
    call task exit for pre-processing
    if return_code = 0 then
      call global exit for preprocessing
      if return_code = 0 then
        process GET, PUT, RENAME, DELETE, or UPDATE
        GOTO post_process_global
      else if return_code = 4 then
        GOTO post_process_task
    else if return_code = 4 then
      return to DESERV caller

  Post_process_global:
    call global exit for post-processing

  Post_process_task:
    call task exit for post-processing
    return to DESERV caller
```

Figure 40. Exit Routine Call Sequence

Establishing Multiple Task level or Multiple Global Exits

The system identifies a DESERV exit by a DST (DESERV Screen Table). The system maintains at most one task level DST for each task, and at most one DST for the system (which represents the global exit). However, multiple DESERV exits can be established. To support multiple exits of a given type (task or global), when a DESERV FUNC=EXIT is issued with EXIT_OPTION=REPLACE, DESERV returns the address of the DST that was replaced (or zero if no DST was replaced). Then it is the responsibility of the newly defined exit to pass control to the previously defined exit.

Issuing DESERV FUNC=EXIT (invocation environment)

INTERRUPTS:
Enabled

STATE and KEY:
Supervisor state, or system key (0-7)

ASC Mode:
P=H=S

AMODE, RMODE:
No restrictions

LOCKS:
None held

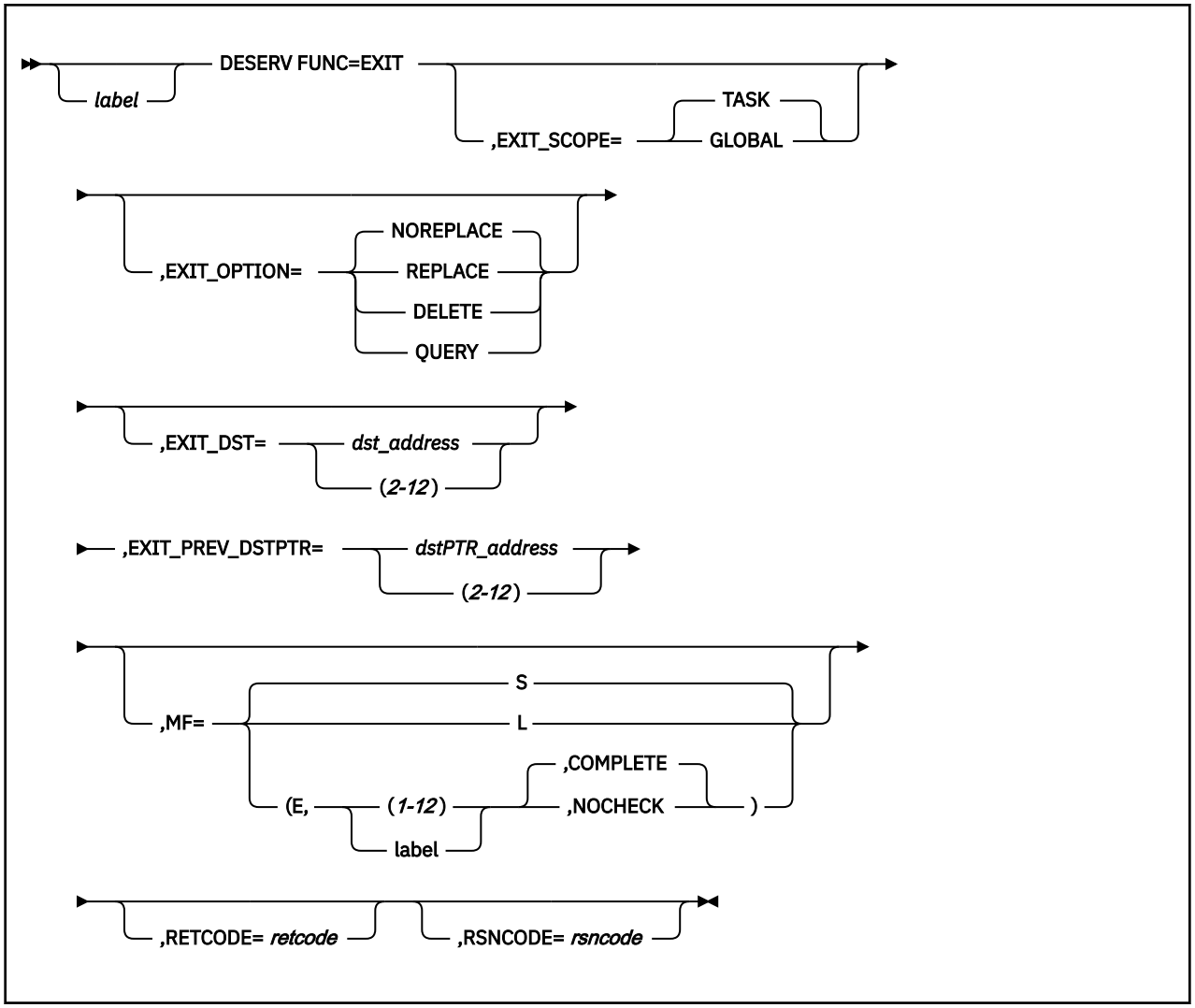
REGISTERS:

- All register contents except registers 15 and 0 are restored on return.
- Register 15 contains the return code and register 0 contains the reason code.
- No save area is required.

SERIALIZATION REQUIREMENTS:
None. DESERV maintains serialization. Therefore the caller does not need to provide any ENQ-like serialization.

Invocation Syntax

The following figure illustrates the syntax for the DESERV EXIT function:



FUNC=EXIT

Requests the DESERV function which operates on a DESERV exit. This keyword is always required except when MF=E is coded with "NOCHECK" and no other keywords are coded or MF=L is coded with no other keywords. For the MF=E case, the FUNC keyword and other keywords specified on the MF=L DESERV macro invocation are assumed to have been coded completely.

EXIT_SCOPE=GLOBAL or TASK

Specifies whether the exit specified will be of a TASK level or of a GLOBAL level.

EXIT_OPTION=REPLACE or NOREPLACE or DELETE or QUERY

If EXIT_OPTION=REPLACE or NOREPLACE is coded, this specifies whether this invocation of the DESERV FUNC=EXIT should replace an existing DESERV EXIT (TASK or GLOBAL as specified by the EXIT_SCOPE parameter). EXIT_PREV_DST will return the existing exit or be set to zero if one does not exist.

If EXIT_OPTION=DELETE is coded, this indicates that the current exit is to be deleted (EXIT_OPTION=DELETE). In this case the EXIT_DST parameter specifies the address of the DST which is to be deleted. This address will be used as the compare value in a compare and swap operation, and only the currently active exit can be deleted. The DST address specified with the EXIT_PREV_DSTPTR parameter will be used as the swap value.

If EXIT_OPTION=QUERY is coded, this indicates that the current exit DST address is to be returned via the EXIT_PREV_DSTPTR parameter.

EXIT_DST=*deserv_exit_screen_table* RX-Type Address or (2-12)
Specifies the address of the DESERV Screen Table (DST). The screen table is mapped by DST DSECT of the IGWDES mapping macro. For an EXIT_OPTION of either NOREPLACE or REPLACE, this parameter defines the DST and defines the exit routine address which becomes the currently active exit if the operation is successful. For an EXIT_OPTION of DELETE, this parameter defines the DST whose address is used as a compare value in a compare and swap operation when deleting the current DST (the address of the input DST is used as the compare value). If the compare fails, DESERV returns an error return and reason code. If EXIT_OPTION=QUERY is coded, this parameter is not required.

EXIT_PREV_DSTPTR=*addr_of_deserv_exit_screen_table* RX-Type Address or (2-12)
The EXIT_PREV_DSTPTR is an output parameter when an EXIT_OPTION of NOREPLACE, REPLACE or QUERY is specified, and an input parameter if an EXIT_OPTION of DELETE is specified.

For EXIT_OPTION=NOREPLACE or QUERY this parameter specifies a four byte field into which DESERV will return the address of the current DST (or zero if no DST exists).

For EXIT_OPTION=REPLACE, this parameter specifies a four byte field into which DESERV will return the address of the DST which was successfully replaced (or zero if no previous DST existed).

For EXIT_OPTION=DELETE, this parameter specifies a four byte field that points to the DST that DESERV restore as the current DST. This address is the swap value for the compare and swap operation.

MF=S or L or {(E,{(1-12) or label}{,COMPLETE or NOCHECK})} RX-Type Address or (1-12) - for MF=E second argument Default=S - if the MF keyword is not specified Default=COMPLETE - if MF=E is specified without the third argument (COMPLETE or NOCHECK).
Specifies the format of the macro expansion.

The Standard form, S, checks all required keywords and keywords that are not valid. This form generates a complete inline expansion of the parameter list and code to call the Directory Entry Services routine. The standard form is for programs that are not reenterable, or for programs that do not change values in the parameter list.

L specifies the List form of the macro. This form generates a remote parameter list. Only keywords of argument type KEY or SYM can be coded. Registers are not valid because code generation does not occur, adcons are generated. Invalid keyword checking is done.

Keywords with defaults that are set by MF=L invocation are not reset to their default during MF=E invocation.

E specifies the Execute form of the macro. This form updates the remote parameter list (MF=L) and transfers control to the DESERV routine.

The second parameter for MF=E format is the address of the parameter list created by the MF=L DESERV invocation. This parameter must be specified as either an RX type of address (possibly the label from MF=L macro invocation) or a register enclosed in parentheses.

The third parameter, COMPLETE or NOCHECK, is optional. Default is COMPLETE. This argument specifies whether required keyword checking will be done. If MF=E is coded with the NOCHECK argument then no, some, or all allowed keywords can be specified, assuming that any missing keywords were coded on the MF=L macro invocation. If MF=E is coded with the COMPLETE argument or allowed to default, the parameter list will be zeroed out (except for the parameter list header) This sets all defaults because the defaults for the DESERV macro are 0. All required keywords must be specified.

RETCODE=retcode RX-Type Address or (2-12)
Specifies the address where the return code returned by DESERV will be stored. Can not be specified on MF=L macro format. The default is not to store the return code in virtual storage. The return code is always returned in register 15 without regard to whether RETCODE is coded.

RSNCODE=rsncode RX-Type Address or (2-12)
Specifies the address where the reason code returned by DESERV will be stored. Can not be specified on MF=L macro format. The default is not to store the return code in virtual storage. The return code is always returned in register 0 without regard to whether RETCODE is coded.

DESERV code is available and used by the system starting with DFSMSdfp Version 1.1. However, invoking the EXIT function requires that the appropriate PTF be applied to the system to enable the support. Your program can test to determine if the appropriate level of DFSMS™ or PTF is installed.

If the DESERV FUNC=EXIT interface is called on DFSMS without the support code being available, DESERV returns an error return and reason code.

Installing or Replacing the DESERV Exit

Your program, while operating in task mode, can establish DESERV exits by issuing the DESERV macro with FUNC=EXIT and appropriate parameters.

Table 80. Installing or Replacing the DESERV Exit

If EXIT_OPTION =	then:	and these parameters...
REPLACE	The new exit definition replaces the most recently defined exit. The DST (DESERV Screen Table) address of the previously defined exit's DST is returned to the caller.	EXIT_DST and EXIT_PREV_DSTPTR must be specified.
NOREPLACE (with no exit currently existing)	The exit definition is activated. A value of zero is returned as the previous DST address.	See REPLACE.
NOREPLACE (with an exit currently existing)	The currently defined exit remains the active exit. The address of the currently defined DST is returned as the previous DST address.	See REPLACE.

Note: If EXIT_OPTION=REPLACE is specified the caller must expect that a previous DST address is returned. It is the responsibility of the caller to replace the old DST address when eventually disabling the new exit. Also the exit routine might need to be aware that there was a previous exit defined, and might choose to invoke the previously defined exit.

EXIT_DST defines the DST address. The DESERV FUNC=EXIT caller owns and manages storage for the DST. The DST can be encapsulated inside other application managed control blocks. This can help the exit routine determine the context of the application in which it was called. When the DST is used with a task level exit, the DST_FLAGS_PROP bit can be set on to indicate that this DST should be propagated to tasks that are attached by this task. The propagate function is not supported for the global exit. The format of the DST is shown in Table 81 on page 346.

Table 81. DESERV Screen Table Structure

Offset	Length or Bit Pattern	Name	Description
0 (X'0')	20	DST	(structure)
0 (X'0')	16	DST_HEADER	(character)
0 (X'0')	8	DST_ID	Eyecatcher 'IGWDST' (character)
08 (X'08')	4	DST_LEN	Length of DST
	X'14'	DST_LEN_IV	Constant to be used with DST_LEN
12 (X'0C')	1	DST_LEV	Control block level (unsigned)
	X'01'	DST_LEV_IV	Constant to be used with DST_LEV

Table 81. DESERV Screen Table Structure (continued)

Offset	Length or Bit Pattern	Name	Description
13 (X'0D')	1	DST_FLAGS	DST flags (unsigned)
	xxxx xxx.	-	Reserved
	1	DST_FLAGS_PROP	Propagate this DST to lower level tasks
14 (X'0E')	2	DST_RES	Reserved
16 (X'10')	4	DST_EXIT	Address of exit routine screen table (address)

EXIT_PREV_DSTPTR returns the address of the DST that was defined prior to this DESERV FUNC=EXIT call.

Deleting the DESERV Exit

An application that has established a DESERV exit can delete the currently active exit by issuing the DESERV macro with FUNC=EXIT. You can specify the following:

Table 82. Deleting the DESERV Exit

For this parameter	specify this:
EXIT_SCOPE	(application dependent)
EXIT_OPTION	DELETE
EXIT_DST	Currently active DST to be deleted
EXIT_PREV_DSTPTR	The DST to become active

Note: The address of the DST must match that of the currently defined DST. If the addresses do not match, DESERV returns error return and reason codes to indicate the DELETE operation has failed and the active DST remains unaffected.

For a task related exit, the DESERV exit is implicitly deleted when the task ends. A global exit can only be explicitly deleted by issuing a DESERV FUNC=EXIT call.

The system does not serialize the deletion of a DESERV exit with the use of that exit by any other task or application. It is up to the application that establishes and deletes the exit to ensure that the storage occupied by the DST and exit code is not freed until all other users of the exit are no longer using the exit. For a task related exit, you can accomplish this by putting the DST and exit into task-related storage and not explicitly freeing the storage. For a global exit, you can accomplish this by putting the DST and exit into system related common storage and never freeing the storage. For practical purposes, waiting for several minutes after the exit is deleted before freeing the storage should be safe as no new invocations of DESERV will use the exit once it is deleted.

Determining If a DESERV Exit Is Active

To determine if a DESERV exit is active issue the DESERV macro with FUNC=EXIT and choose the following options:

Table 83. Determining If a DESERV Exit Is Active

For this parameter	choose this:
EXIT_OPTION	QUERY
EXIT_SCOPE	(application dependant)

Table 83. Determining If a DESERV Exit Is Active (continued)

For this parameter	choose this:
EXIT_PREV_DSTPTR	The address of a 4 byte area into which the currently active DST address is returned (or zero if there is no currently defined DST address.)

Writing the DESERV Exit

A DESERV exit gets control once prior to any DESERV GET, PUT, RENAME, UPDATE or DELETE function processing, and once immediately prior to DESERV's return to the caller. A DESERV exit receives control in key 0 and supervisor state. Register 13 points to an 18-word key 0 register save area.

The DESX DSECT maps the input to the exit routines and is defined in the IGWDES macro. Register 1 points to the DESX on entry to the exit routine. The DESX structure is shown in Table 84 on page 348.

Table 84. DESX Structure Mapping DESERV Exit Parameter List

Offset	Length or Bit Pattern	Name	Description
0 (X'0')	36	DESX	(structure)
0 (X'0')	16	DESX_HEADER	(character)
0 (X'0')	8	DESX_ID	Eyecatcher - IGWDESX (character)
08 (X'08')	4	DESX_LEN	Length of DESX (signed)
	X'24'	DESX_LEN_IV	Constant to be used with DESX_LEN
12 (X'0C')	1	DESX_LEV	Control block level (character)
	X'01'	DESX_LEV_IV	Constant to be used with DESX_LEV
13 (X'0D')	3	-	Reserved
16 (X'10')	4	DESX_DESP_PTR	Address of caller's DESP (address)
20 (X'14')	4	DESX_DST	Address of DESERV screen table (address)
24 (X'18')	1	DESX_CALLER_KEY	Key of DESERV caller in bits 0-3 (unsigned)
25 (X'19')	1	DESX_FLAGS	(bitstring)
	1...	DESX_BLDL_BIT	DESERV issues BLDL to process this GET request
	.1..	DESX_PREV_BIT	EXIT called before DESERV PUT or GET function
	..1.	DESX_POST_BIT	EXIT called after DESERV PUT or GET function
26 (X'1A')	2	-	Reserved
28 (X'1C')	4	DESX_RETURN_CODE	Return code to be returned to DESERV caller (unsigned)
32 (X'20')	4	DESX_REASON_CODE	Reason code to be returned to DESERV caller (unsigned)

Note that the DESERV return and reason codes, with the exception of the PUT codes, can be found in *z/OS DFSMS Using Data Sets*. See Figure 41 on page 363 for the PUT return and reason codes.

Parameters Related to the GET Function

If the DESERV exit gets control for a DESERV GET function invocation, DESX_DESP_PTR points to the DESERV parameter list. If the DESP field DESP_FUNC=X'01' (DESP_FUNC_GET), this indicates a GET function parameter list. See Table 85 on page 349 for the DESP structure for fields pertaining to a DESERV GET invocation.

DESERV GET will return information on selected members. This information is returned in a DESB structure. The DESB is mapped by the DESB DSECT in the IGWDES macro. If the storage for the DESB is provided by the DESERV GET, the DESP_AREA_PTR field contains the address of this storage. Alternatively the caller may request that DESERV GET **not** obtain the storage for the DESB. In this case the DESP_AREAPTR_PTR field contains the address of a 4 byte area into which DESERV GET will return the address of a DESB. The DESB will be obtained in the subpool identified by DESP_SUBPOOL (or default to subpool zero). The flag DESP_SUBPOOL_FLG indicates whether the subpool was specified explicitly by the DESERV GET caller.

A DESERV GET invocation identifies the members to be searched for by a name list, a PDS format directory entry, or an SMDE. The DESP field DESP_GETTYPE defines the get type. If the get type is a PDSDE (the member to be searched for is defined by a PDS format directory entry), the DESP_PDSDE_PTR points to a directory entry as returned by the BLDL macro. The PDS2 DSECT in the IHAPDS macro maps this structure. The function of DESERV GET for a PDSDE get type depends on the type of library identified by the concatenation number in the PDS style directory entry. If the concatenation number identifies a PDS, the GET function is simply to convert the PDS style directory entry into a SMDE. If the concatenation number identifies a PDSE, the GET function is to connect to the member identified by the PDS2TTRP field, and to return the appropriate SMDE. In either case the SMDE is retuned (if in the PDSE case the member actually exists) in the data portion of the output buffer (DESB, mapped below).

If the get type is name list, the DESL area points to the names to be searched for. The DESP_NAME_LIST_PTR points to the DESL and the DESL DSECT in the IGWDES macro maps it. A DESL is an array consisting of the number of entries the DESP field DESP_NAME_LIST2 defines. The DESL parameter list is shown in Table 86 on page 351.

If the get type is SMDE, the DESP_SMDE_PTR points to a system-managed directory entry (SMDE) as returned by DESERV GET. DESERV GET will cause a connection to the member identified by the SMDE. DESERV GET will return a copy of the SMDE in the output DESB. The SMDE returned will of course have updated connect token and connect id fields.

There are two input flags which control DESERV GET's view of the PDS or PDSE to be searched. If the DESP_C370LIB flag is on, a PDS may be viewed as a C370LIB. This means that if the PDS has a special member named @@DC370$, this member is treated as the "real" directory for the PDS. If the DESP_SYSTEM_DCB is on, this indicates that the caller (who must be authorized) has indicated that this DCB is "owned by the system" and is not on any DEB chain, therefore DEBCHK should not be done.

Hint: The name of the special member @@DC370$ might not display correctly on your screen or printer. The first two characters are X'7C' and the last character is X'5B'.

Table 85. Structure of DESP for DESERV GET Invocations

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	104	DESP	DE Services parameter list (structure)
00 (X'00')	16	DESP_HEADER	(character) 'IGWDESP'
00 (X'00')	8	DESP_ID	Eyecatcher IGWDESP (character)
08 (X'08')	24	DESP_LEN	Length of DESP (signed)
	X'04'	DESP_LEN_IV	Constant to be used with DESP_LEN
12 (X'0C')	1	DESP_LEV	Control block level (character)
	X'01'	DESP_LEV_IV	Constant to be used with DESP_LEV
13 (X'0D')	3	-	Reserved

Table 85. Structure of DESP for DESERV GET Invocations (continued)

Offset	Length or Bit Pattern	Name	Description
16 (X'10')	1	DESP_FUNC	Function type (unsigned)
	X'07'	DESP_FUNC_DELETE	Function is DELETE
	X'08'	DESP_FUNC_RENAME	Function is RENAME
	X'09'	DESP_FUNC_UPDATE	Function is UPDATE
	X'04'	DESP_FUNC_PUT	Function is PUT
	X'01'	DESP_FUNC_GET	Function is GET
	X'00'	DESP_FUNC_OMITTED	Function is omitted
17 (X'11')	3	-	Reserved
20 (X'14')	4	-	Reserved
24 (X'18')	12	DESP_DATA	Function data (character)
24 (X'18')	2	DESP_FLAGS	Flags (bitstring)
	1...	DESP_BYPASS_LLA	0=USE LLA, 1=BYPASS LLA
	.x..	-	Reserved
	..1.	DESP_SUBPOOL_FLG	0=SUBPOOL not specified, 1=SUBPOOL specified
	...1	DESP_C370LIB	1=treat PDSs as C370LIB if @@DC370$ member exists
	.... xx..	-	Reserved
	.... ..1.	DESP_SYSTEM_DCB	1=treat DCB as a system DCB
26 (X'1A')	1	-	Reserved
27 (X'1B')	1	-	Reserved
28 (X'1C')	1	DESP_LIBTYPE	Indicates whether a DCB or DEB is input =X'02', DEB input=X'01') (unsigned)
	X'02'	DESP_LIBTYPE_DCB	Constant to be used with DESP_LIBTYPE
	X'01'	DESP_LIBTYPE_DEB	Constant to be used with DESP_LIBTYPE
	X'00'	DESP_LIBTYPE_OMITTED	Constant to be used with DESP_LIBTYPE
29 (X'1D')	1	DESP_GETTYPE	Indicates whether Name List or PDSDE is input. (NAME_LIST input=(X'01', PDSDE input=X'02') (unsigned)
	X'03'	DESP_GETTYPE_SMDE	Constant to be used with DESP_GETTYPE
	X'02'	DESP_GETTYPE_PDSDE	Constant to be used with DESP_GETTYPE
	X'01'	DESP_GETTYPE_NAME_LIST	Constant to be used with DESP_GETTYPE
	X'00'	DESP_GETTYPE_OMITTED	Constant to be used with DESP_GETTYPE
30 (X'1E')	1	-	Reserved
31 (X'1F')	1	-	Reserved
32 (X'20')	1	-	Reserved
33 (X'21')	1	DESP_SUBPOOL	Subpool number for getting DESB.

Table 85. Structure of DESP for DESERV GET Invocations (continued)

Offset	Length or Bit Pattern	Name	Description
34 (X'22')	1	DESP_CONN_INTENT	Connect intent (unsigned)
	X'03'	DESP_CONN_INTENT_INPUT	INPUT
	X'02'	DESP_CONN_INTENT_EXEC	EXEC
	X'01'	DESP_CONN_INTENT_HOLD	HOLD
	X'00'	DESP_CONN_INTENT_NONE	None
35 (X'23')	1	-	Reserved
36 (X'24')	4	DESP_DCB_PTR	DCB address, valid if DESP_LIBTYPE=X'02' (address)
40 (X'28')	4	DESP_DEB_PTR	DEB address, valid if DESP_LIBTYPE=X'01' (address)
44 (X'2C')	4	DESP_CONN_ID_PTR	Connect identifier address
48 (X'30')	4	DESP_AREAPTR_PTR	Address of output field for buffer address
52 (X'34')	4	DESP_AREA_PTR	Buffer address
56 (X'38')	4	DESP_AREA2	Buffer length (unsigned)
60 (X'3C')	4	-	Reserved
64 (X'40')	4	-	Reserved
68 (X'44')	4	DESP_ENTRY_GAP	Entry gap size (signed)
72 (X'48')	4	-	Reserved
76 (X'4C')	4	-	Reserved
80 (X'50')	4	DESP_NAME_LIST_PTR	Name list address, valid if DESP_GETTYPE=X'01' (address)
84 (X'54')	4	DESP_NAME_LIST2	Input list number of entries, valid if DESP_GETTYPE=X'01' (unsigned)
88 (X'58')	4	-	Reserved
92 (X'5C')	4	DESP_PDSDE_PTR	BLDL directory entry address, valid if DESP_GETTYPE=X'02' (address)
92 (X'5C')	4	DESP_SMDE_PTR	SMDE directory entry address, valid if DESP_GETTYPE=X'03' (address)
96 (X'60')	4	-	Reserved
100 (X'64')	4	-	Reserved

Table 86. DESL Structure

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	16	DESL	Name list (structure)
00 (X'00')	16	DESL_ENTRY	Name list entry (character)
00 (X'00')	1	DESL_FLAGS	Flags (unsigned)
	1...	DESL_MODULE_BUFFERED_ LLA	Module is staged by LLA

Table 86. DESL Structure (continued)

Offset	Length or Bit Pattern	Name	Description
1 (X'1')	1	DESL_CODE	Result code (New name exists=X'03', Error=X'02', Not found or not processed=X'01', Found=X'00') (unsigned)
	X'03'	DESL_CODE_NEWNAME_ EXISTS	For func=rename, indicates a new name already existed in the PDSE.
	X'02'	DESL_CODE_ERROR	An unexpected error has occurred. The DESL_ERRCODE field is set to a DESRF value
	X'01'	DESL_CODE_NOTFOUND	Entry not found or entry not processed. If func=rename, old name was not found.
	X'00'	DESL_CODE_SUCC	Entry successfully processed
2 (X'2')	2	DESL_ERRCODE	Error reason code (low order halfword of DESERV reason code if error) (unsigned)
4 (X'4')	4	-	Reserved
08 (X'08')	4	DESL_SMDE_PTR	Pointer to SMDE within DESB. Output for GET function, input for UPDATE function (address)
08 (X'08')	4	DESL_NEW_NAME_PTR	Pointer to new name (DESN) descriptor for RENAME function (address)
12 (X'0C')	4	DESL_NAME_PTR	Pointer to name (DESN) descriptor for GET and DELETE functions (address)
12 (X'0C')	4	DESL_OLD_NAME_PTR	Pointer to old name (DESN) descriptor for RENAME function (address)

The DESERV GET caller will have built the DESL to point to variable length names. The DESN DSECT maps these names in the IGWDES macro. See Table 87 on page 352 for the DESN parameter list.

Table 87. DESN Parameter List

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	variable	DESN	Name record (structure)
00 (X'00')	2	DESN_LEN	Length of name that follows (unsigned)
2 (X'02')	variable	DESN_VAL	Name data (character)

The DESP_CONN_INTENT field of the DESP indicates the connection intent requested by the caller. The connection intent only has an effect if the name is found in a PDSE. If the connection intent is DESP_CONN_INTENT_HOLD (X'01'), the effect is similar to a BLDL invocation (because the member is connected for HOLD which is not sufficient to read the member). If the connection intent is DESP_CONN_EXEC (X'02') or DESP_CONN_INTENT_INPUT (X'03'), the effect is similar to a FIND invocation (because the member is connected and sufficient control blocks are built so that the member can be read). The GET function does not currently support a connect intent of NONE.

The output from DESERV GET consists of flags and error codes in the DESL (if the get type is name list) as well as an SMDE (system managed directory entry) pointer. For a gettype of name list, the SMDE is pointed to by the DESL_SMDE_PTR field. For a gettype of PDSDE, the SMDE is in the DESB at the label DESB_DATA. The SMDE is mapped by the SMDE DSECT in the IGWSMDE macro and the PMAR DSECT in the IEWPMAR macro. The SMDE resides in the output buffer as provided by the caller of DESERV GET. The output buffer is mapped by the DESP DSECT of the IGWDES macro. The DESB structure is shown in Table 88 on page 353.

The basic SMDE format is shown in Table 89 on page 353.

Table 88. DESB Parameter List

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	variable	DESB	DEServ buffer header (structure)
00 (X'00')	40	DESB_FIXED	(character)
00 (X'00')	16	DESB_HEADER	(character)
00 (X'00')	8	DESB_ID	Eyecatcher - IGWDESB (character)
08 (X'08')	4	DESB_LEN	Length of buffer (signed)
12 (X'0C')	1	DESB_LEV	Control block level (character)
	X'01'	DESB_LEV_IV	Constant to be used with DESB_LEV
13 (X'0D')	3	-	Reserved
16 (X'10')	4	DESB_NEXT	Next buffer pointer (address)
20 (X'14')	4	-	Reserved
24 (X'18')	4	DESB_COUNT	Count of entries in this buffer (unsigned)
28 (X'1C')	4	DESB_AVAIL	Start of free space in buffer (address)
32 (X'20')	1	-	Reserved
33 (X'21')	1	DESB_SUBPOOL	Subpool number (unsigned)
34 (X'22')	2	DESB_GAP_LEN	Length of user-requested gap (unsigned)
36 (X'24')	4	-	Reserved
40 (X'28')	variable	DESB_DATA	Start of data area (character)

Table 89. SMDE Format

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	variable	SMDE	Member directory entry (structure)
00 (X'00')	44	SMDE_BASIC	Start of basic section (character)
00 (X'00')	16	SMDE_HDR	Header (character)
00 (X'00')	8	SMDE_ID	Eyecatcher (character)
08 (X'08')	4	SMDE_LEN	Length of control block. This is the sum of the sizes of the SMDE sections and the size of the user data. (unsigned)
12 (X'0C')	1	SMDE_LVL	SMDE version number (unsigned)
	X'01'	SMDE_LVL_VAL	Constant to be used with SMDE_LVL
13 (X'0D')	3	-	Reserved
16 (X'10')	1	SMDE_LIBTYPE	Source library type. Possible values are declared below with names like SMDE_LIBTYPE_XXX. (unsigned)
	X'03'	SMDE_C370LIB	Constant to be used with SMDE_LIBTYPE
	X'02'	SMDE_LIBTYPE_HFS	Constant to be used with SMDE_LIBTYPE
	X'01'	SMDE_LIBTYPE_PDSE	Constant to be used with SMDE_LIBTYPE
	X'00'	SMDE_LIBTYPE_PDS	Constant to be used with SMDE_LIBTYPE

Table 89. SMDE Format (continued)

Offset	Length or Bit Pattern	Name	Description
17 (X'11')	1	SMDE_FLAG	Flag byte (bitstring)
	1...	SMDE_FLAG_ALIAS	Entry is an alias
	.1..	SMDE_FLAG LMOD	Member is a program
	..xx xxxx	*	Reserved
18 (X'12')	2	-	Reserved, must be zero
20 (X'14')	5	-	Extended MLTK (character)
20 (X'14')	1	-	Reserved, must be zero
21 (X'15')	4	SMDE_MLTK	MLT and concatenation number (character)
21 (X'15')	3	SMDE_MLT	MLT of member - zero if HFS (character)
24 (X'18')	1	SMDE_CNCT	Concatenation number (unsigned)
25 (X'19')	1	SMDE_LIBF	Library flag - Z-byte (unsigned)
	X'02'	SMDE_LIBF_TASKLIB	Constant to be used with SMDE_LIBF
	X'01'	SMDE_LIBF_LINKLIB	Constant to be used with SMDE_LIBF
	X'00'	SMDE_LIBF_PRIVATE	Constant to be used with SMDE_LIBF
26 (X'1A')	2	SMDE_NAME_OFF	Name offset (signed)
28 (X'1C')	2	SMDE_USRD_LEN	User data length (signed)
28 (X'1C')	2	SMDE_PMAR_LEN	Sum of lengths of program management attribute record sections (PMAR, PMARR, PMARL) (signed)
30 (X'1E')	2	SMDE_USERD_OFF	User data offset (signed)
30 (X'1E')	2	SMDE_PMAR_OFF	Program management attribute record offset (signed)
32 (X'20')	2	SMDE_TOKEN_LEN	Token length (signed)
34 (X'22')	2	SMDE_TOKEN_OFF	Token data offset (signed)
36 (X'24')	2	SMDE_PNAME_OFF	Primary name offset, zero for non-alias SMDES or if library type is a PDS and this is not a program. (signed)
38 (X'26')	2	SMDE_NLST_CNT	Number of note list entries that exist at beginning of user data field. Always zero for non-PDS members. (signed)
40 (X'28')	4	-	Reserved
44 (X'2C')	variable	SMDE_SECTIONS	Start of entry sections (character)

Table 90 on page 354 through Table 93 on page 355 shows the optional SMDE_SECTIONS, or extensions to the SMDE.

Table 90. Directory Entry Name Section

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	variable	SMDE_NAME	Name descriptor (structure)
00 (X'00')	2	SMDE_NAME_LEN	Length of entry name (signed)
2 (X'02')	variable	SMDE_NAME_VAL	Entry name (character)

Table 91. Directory Entry Notelist Section (PDS Only)

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	variable	SMDE_NLST	Note list extension (structure)
00 (X'00')	4	SMDE_NLST_ENTRY	Note list entries (character)
00 (X'00')	3	SMDE_NLST_RLT	Note list record location token (character)
3 (X'03')	1	SMDE_NLST_NUM	Number of RLT described by this note list block. If 0 this is not a notelist but a data block. (unsigned)

Table 92. Directory Entry Token Section

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	32	SMDE_TOKEN	(structure)
00 (X'00')	4	SMDE_TOKEN_CONNID	CONNECT_IDENTIFIER (unsigned)
4 (X'04')	4	SMDE_TOKEN_ITEMNO	Item number (unsigned)
08 (X'08')	24	SMDE_TOKEN_FT	File token (character)

Table 93. Directory Entry Primary Name Section

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	variable	SMDE_PNAME	Primary name descriptor (structure)
00 (X'00')	2	SMDE_PNAME_LEN	Length of primary name (signed)
2 (X'02')	variable	SMDE_PNAME_VAL	Primary name (character)

If the SMDE represents a directory entry for a program (either a load module or a program object) the program's attributes are defined by the PMAR structure. The PMAR is a subfield of the SMDE and its offset is defined by the field SMDE_PMAR_OFF. Table 94 on page 355 shows the basic PMAR definition. Table 95 on page 357 and Table 96 on page 359 show the PMAR extensions for program objects (PMARL) and load modules (PMARR), respectively.

If the SMDE represents a data member of a PDS or a PDSE, the SMDE_USRD_OFF field indicates the offset into the SMDE for the user data of the directory entry.

Table 94. Directory Entry Name Section. Data is always present at offset SMDE_PMAR_OFF in an SMDE.

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	30	PMAR	Basic section of program user data (structure)
00 (X'00')	30	PMAR_ENTRY	Alternative name for the PMAR section (character)
00 (X'00')	2	PMAR_SLEN	Section length (unsigned)
2 (X'02')	1	PMAR_LVL	PMAR format level (unsigned)
	X'02'	PMAR_LVL_VAL	Constant to be used with PMAR
	X'01'	PMAR_PM1_VAL	Constant to be used with PMAR
	X'02'	PMAR_PM2_VAL	Constant to be used with PMAR

Table 94. Directory Entry Name Section. Data is always present at offset SMDE_PMAR_OFF in an SMDE. *(continued)*

Offset	Length or Bit Pattern	Name	Description
3 (X'03')	1	PMAR_PLVL	Bind processor creating object 1 - E-level linkage editor 2 - F-level linkage editor 3 - (VS1/VS2) linkage editor 4 - XA linkage editor 5 - binder version 1. (unsigned)
	X'01'	PMAR_PLVL_E_VAL	Constant to be used with PMAR_PLVL
	X'02'	PMAR_PLVL_F_VAL	Constant to be used with PMAR_PLVL
	X'03'	PMAR_PLVL_AOS_VAL	Constant to be used with PMAR_PLVL
	X'04'	PMAR_PLVL_XA_VAL	Constant to be used with PMAR_PLVL
	X'05'	PMAR_PLVL_B1_VAL	Constant to be used with PMAR_PLVL
	X'06'	PMAR_PLVL_B2_VAL	Constant to be used with PMAR_PLVL
4 (X'04')	4	PMAR_ATR	Attribute bytes (character)
4 (X'04')	1	PMAR_ATR1	First attribute byte. These flags must be at the same offsets as the corresponding flags in PDS2ATR1 declared by macro IHAPDS. (bitstring)
	1...	PMAR_RENT	Reenterable
	.1..	PMAR_REUS	Reusable
	..1.	PMAR_OVLY	Overlay structure
	...1	PMAR_TEST	Module to be tested - TSO/E TEST
	.... 1...	PMAR LOAD	Only loadable
	.... .1..	PMAR_SCTR	Scatter format
	.... ..1.	PMAR_EXEC	Executable
	1	PMAR_1BLK	Load module contains only one block of text data and has no RLD data.
5 (X'05')	1	PMAR_ATR2	Second attribute byte. These flags must be at the same offsets as the corresponding flags in PDS2ATR2 declared by macro IHAPDS. (bitstring)
	1...	PMAR_FLVL	If on, the program cannot be processed by the E level linkage editor. If off, the program can be processed by any level of the linkage editor or the binder.
	.1..	PMAR_ORGO	Linkage editor assigned origin of first block of text is zero.
	..x.	-	Reserved
	...1	PMAR_NRLD	Program contains no RLD items
	.... 1...	PMAR_NREP	Module cannot be reprocessed by the linkage editor
	.... .1..	PMAR_TSTN	Module contains TSO/E TEST symbol records
	.... ..x.	-	Reserved
	1	PMAR REFR	Refreshable program
6 (X'06')	1	PMAR_ATR3	Third attribute byte. (bitstring)

Table 94. Directory Entry Name Section. Data is always present at offset SMDE_PMAR_OFF in an SMDE. *(continued)*

Offset	Length or Bit Pattern	Name	Description
6 (X'06')	1	PMAR_FTB1	Alternative name for flags byte. These flags must be at the same offsets as the corresponding flags in PDS2FTB1 declared by macro IHAPDS. (bitstring)
	x...	-	Reserved
	.1..	PMAR_BIG	This program requires 16MB or more of virtual storage.
	..1.	PMAR_PAGA	Page alignment is required
	...1	PMAR_XSSI	SSI information present
	.... 1...	PMAR_XAPF	APF information present
	.... .1..	PMAR_LFMT	PMARL follows PMAR.
	xx	-	Reserved
7 (X'07')	1	PMAR_ATR4	Fourth attribute byte (bitstring)
7 (X'07')	1	PMAR_FTB2	Alternative name for flags byte. These flags must be at the same offsets as the corresponding flags in PDS2FTB2 declared by macro IHAPDS. (bitstring)
	1...	PMAR_ALTP	Alternate primary flag. If on for a primary name, indicates primary name was generated by the binder. If on for an alias, indicates the long alias name was specified as the primary name on the bind.
	.xx.	-	Reserved
	...1	PMAR_RMOD	RMODE is ANY.
	.... xx..	PMAR_AAMD	Alias entry point addressing mode. If B'00', AMODE is 24. If B'10', AMODE is 31. If B'11', AMODE is ANY.
	xx	PMAR_MAMD	Main entry point addressing mode. If B'00' AMODE is 24. If B'10', AMODE is 31. If B'11', AMODE is ANY.
08 (X'08')	1	-	Reserved
9 (X'09')	1	PMAR_AC	APF authorization code (unsigned)
10 (X'0A')	4	PMAR_STOR	Virtual storage required (unsigned)
14 (X'0E')	4	PMAR_EPM	Main entry point offset (unsigned)
18 (X'12')	4	PMAR_EPA	This entry point offset (unsigned)
22 (X'16')	4	PMAR_SSI	SSI information (bitstring)
22 (X'16')	1	PMAR_CHLV	Change level of member (unsigned)
23 (X'17')	1	PMAR_SSFB	SSI flag byte (bitstring)
24 (X'18')	2	PMAR_MSER	Member serial number (Reserved)
26 (X'1A')	4	-	Reserved
30 (X'1E')	variable	PMAR_END	End of basic section (character)

Table 95. LSLoader Attributes Unique to Program Objects. If PMAR_LFMT=ON this section follows the PMAR basic section.

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	50	PMARL	LSLoader section for program objects (structure)

Table 95. LSLoader Attributes Unique to Program Objects. If PMAR_LFMT=ON this section follows the PMAR basic section. *(continued)*

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	2	PMARL_SLEN	Section length (unsigned)
2 (X'02')	48	PMARL_DATA	Section data (character)
2 (X'02')	4	PMARL_ATR	Attribute bytes (character)
2 (X'02')	1	PMARL_ATR1	Fifth attribute byte (bitstring)
	1...	PMARL_NMIG	This program object cannot be converted directly to PDS load module format.
	.1..	PMARL_PRIM	FETCHOPT PRIME option
	..1.	PMARL_PACK	FETCHOPT PACK option
	...x xxxx	-	Reserved
3 (X'03')	1	PMARL_ATR2	Sixth attribute byte (bitstring)
	1...	PMARL_CMPR	Compressed format module
	.1..	PMARL_1RMOD	1st segment is RMODE Any, set for PM2-level PO only
	..1.	PMARL_2RMOD	2nd segment is RMODE Any, set for PM2-level PO if there are at least two segments.
	...1	PMARL_SEGM	Loader data includes a Segment Table with >1 loadable entry or a Gas Table, set for PM2-level PO only.
	.... 1...	PMARL_1ALIN	1st segment is page-aligned, set for PM2-level PO only
	.... .1..	PMARL_2ALIN	2nd segment is page-aligned, set for PM2-level PO if there are at least 2 segments.
	.... ..1.	PMARL_FILL	FILL option specified set for PM2-level PO only
	x	-	Reserved
4 (X'04')	1	PMARL_FILLVAL	FILL character value set for PM2-level PO only
5 (X'05')	1	-	Reserved
THE FOLLOWING NOTED FIELDS ARE NOT INTENDED FOR USE. INCLUDED HERE FOR INFORMATION PURPOSES ONLY.			
6 (X'06')	4	PMARL_MPGS	Total length of program on DASD in pages (unsigned)
10 (X'0A')	40	PMARL_MDAT	DASD program descriptors (character)
10 (X'0A')	4	PMARL_TXTL	Length of text (unsigned)
14 (X'0E')	4	PMARL_TXTO	Offset to text (address)
18 (X'12')	4	PMARL_BDRL	Length of binder index (unsigned)
22 (X'16')	4	PMARL_BDRO	Offset to binder index (address)
26 (X'1A')	4	PMARL_RDTL	Length of PRDT (unsigned)
30 (X'1E')	4	PMARL_RDTO	Offset to PRDT (address)
34 (X'22')	4	PMARL_RATL	Length of PRAT (unsigned)
38 (X'26')	4	PMARL_RATO	Offset to PRAT (address)
42 (X'2A')	4	PMARL_NVSPGS	Number of virtual storage pages to contain program object, for PM2-level PO

Table 95. LSLoader Attributes Unique to Program Objects. If PMAR_LFMT=ON this section follows the PMAR basic section. *(continued)*

Offset	Length or Bit Pattern	Name	Description
42 (X'2A')	4	PMARL_LMDL	Length of LSLoader data (unsigned) for PM1-level PO
46 (X'2E')	4	PMARL_LMDO	Offset to LSLoader data (address)
50 (X'32')	2	PMARL_NSEG	Number of loadable segments
52 (X'34')	2	PMARL_NGAS	Count of entries in Gas Table
54 (X'36')	4	PMARL_1STOR	Virtual storage required for first loadable segment, valid when PMARL_NSEG > 1.
58 (X'3A')	4	PMARL_2STOR	Virtual storage required for second loadable segment, valid when PMARL_NSEG > 1.
62 (X'3E')	4	PMARL_2TXTO	Offset to second txt segment including gas, valid when PMARL_NSEG > 1.
END INFORMATION ONLY FIELDS.			
66 (X'42')	16	PMARL_TRACE	AUDIT trace data
66 (X'42')	4	PMARL_DATE	Date saved
70 (X'46')	4	PMARL_TIME	Time saved
74 (X'4A')	8	PMARL_USER	User or job identification
82 (X'52')	variable	PMARL_END	End of LSLoader section (character)

Table 96. Attributes Unique to Load Modules (PDS only). If PMAR_LFMT=OFF then this section follows the PMAR basic section.

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	23	PMARR	Load module (PDS) attributes section (structure)
00 (X'00')	2	PMARR_SLEN	Section length (unsigned)
2 (X'02')	21	PMARR_DATA	Section data (character)
2 (X'02')	8	PMARR_TTRS	TTR fields (character)
2 (X'02')	3	PMARR_TTRT	TTR of first block of text (character)
5 (X'05')	1	PMARR_ZERO	Zero (character)
6 (X'06')	3	PMARR_TTRN	TTR of note list or scatter translation table. Used for modules in scatter load format or overlay structure only. (character)
9 (X'09')	1	PMARR_NL	Number of entries in note list for scatter format modules and modules in overlay structure, otherwise zero. (address)
10 (X'0A')	2	PMARR_FTBL	Length of first block of text (signed)
12 (X'0C')	3	PMARR_ORG	Load module origin if 0 (unsigned)
12 (X'0C')	2	-	Reserved
14 (X'0E')	1	PMARR_RLDS	Number of RLD/CTL records that follow the first text record
15 (X'F')	8	PMARR_SCAT	Scatter load information (character)
15 (X'F')	2	PMARR_SLSZ	Scatter list length (unsigned)

Table 96. Attributes Unique to Load Modules (PDS only). If PMAR_LFMT=OFF then this section follows the PMAR basic section. *(continued)*

Offset	Length or Bit Pattern	Name	Description
17 (X'11')	2	PMARR_TTSZ	Translation table length (unsigned)
19 (X'13')	2	PMARR_ESDT	ESDID of first text block (character)
21 (X'15')	2	PMARR_ESDC	ESDID of EP control section (character)
23 (X'17')	variable	PMARR_END	End of load module attributes (character)

Table 97. Alias in Unformatted Form. Used only as input to the PUT function.

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	7	PMARA	PMAR alias entry section (structure)
00 (X'00')	2	PMARA_LEN	Section length (unsigned)
2 (X'02')	5	PMARA_DATA	Section data (character)
2 (X'02')	4	PMARA_EPA	Entry point offset (unsigned)
6 (X'06')	1	PMARA_ATR	Attribute bytes (character)
6 (X'06')	1	PMARA_ATR1	First attribute byte (bitstring)
6 (X'06')	1	PMARA_FTB2	Alternative name for flags byte. These flags must be at the same offsets as the corresponding flags in PDS2FTB2 declared by macro IHAPDS. (bitstring)
	xxxx	-	Reserved
	.... 11..	PMARA_AMD	Alias entry addressing mode. If B'00', AMODE is 24. If B'10', AMODE is 31. If B'11', AMODE is ANY.
	xx	-	Reserved
7 (X'07')	variable	PMARA_END	End of alias entry section (character)

Parameters Related to the PUT Function

If the DESERV exit gets control for a DESERV put function invocation, DESX_DESP_PTR points to the DESERV parameter list. If the DESP field DESP_FUNC=X'04' (DESP_FUNC_PUT), this indicates a PUT function parameter list. Table 98 on page 360 shows the fields of the DESP that pertain to the DESERV PUT invocation.

Table 98. DESERV PUT DESP fields

Offset	Length or bit pattern	Name	Description
00 (X'00')	104	DESP	DE Services parameter list (structure)
00 (X'00')	16	DESP_HEADER	Standard header (character)
00 (X'00')	8	DESP_ID	Eyecatcher 'IGWDESP' (character)
08 (X'08')	4	DESP_LEN	Length of DESP (signed)
	4	DESP_LEN_IV	Constant to be used for DESP_LEN
12 (X'0C')	1	DESP_LEV	Control block level (character)
	4	DESP_LEV_IV	Constant to be used for DESP_LEV
13 (X'0D')	3	-	Reserved

Table 98. DESERV PUT DESP fields (continued)

Offset	Length or bit pattern	Name	Description
16 (X'10')	1	DESP_FUNC	Function type (GET=X'01', PUT=X'04', DELETE=X'07', RENAME=X'08', UPDATE=X'09') (unsigned)
	X'07'	DESP_FUNC_DELETE	Constant to be used for DESP_FUNC
	X'08'	DESP_FUNC_RENAME	Constant to be used for DESP_FUNC
	X'09'	DESP_FUNC_UPDATE	Constant to be used for DESP_FUNC
	X'04'	DESP_FUNC_PUT	Constant to be used for DESP_FUNC
	X'01'	DESP_FUNC_GET	Constant to be used for DESP_FUNC
	X'00'	DESP_FUNC_OMITTED	Constant to be used for DESP_FUNC
17 (X'11')	3	-	Reserved
20 (X'14')	4	-	Reserved
24 (X'18')	12	DESP_DATA	Function data (character)
24 (X'18')	2	-	Reserved
26 (X'1A')	1	-	Reserved
27 (X'1B')	1	-	Reserved
28 (X'1C')	1	DESP_LIBTYPE	Indicates whether a DCB or a DEB is input. (unsigned)
	X'02'	DESP_LIBTYPE_DCB	DCB input
	X'01'	DESP_LIBTYPE_DEB	DEB input
	X'00'	DESP_LIBTYPE_OMITTED	Omitted
29 (X'1D')	1	-	Reserved
30 (X'1E')	1	-	Reserved
31 (X'1F')	1	-	Reserved
32 (X'20')	1	DESP_OPTION	REPLACE option (REPLACE=X'01', NOREPLACE=X'00') (unsigned)
	X'02'	DESP_OPTION_REPLACE_ALIAS	Constant to be used with DESP_OPTION
	X'01'	DESP_OPTION_REPLACE	Constant to be used with DESP_OPTION
	X'00'	DESP_OPTION_NOREPLACE	Constant to be used with DESP_OPTION
33 (X'21')	1	-	Reserved
34 (X'22')	1	-	Reserved
35 (X'23')	1	-	Reserved
36 (X'24')	4	DESP_DCB_PTR	DCB address
40 (X'28')	4	DESP_DEB_PTR	DEB address
44 (X'2C')	4	-	Reserved
48 (X'30')	4	-	Reserved
52 (X'34')	4	-	Reserved
56 (X'38')	4	-	Reserved
60 (X'3C')	4	-	Reserved

Table 98. DESERV PUT DESP fields (continued)

Offset	Length or bit pattern	Name	Description
64 (X'40')	4	-	Reserved
68 (X'44')	4	-	Reserved
72 (X'48')	4	DESP_MEM_DATA_PTR	MEM_DATA_ADDRESS
76 (X'4C')	4	DESP_MEM_DATA2	MEM_DATA entry count (unsigned)
80 (X'50')	4	-	Reserved
84 (X'54')	4	-	Reserved
88 (X'58')	4	-	Reserved
92 (X'5C')	4	-	Reserved
96 (X'60')	4	-	Reserved
100 (X'64')	4	-	Reserved

PUT Return and Reason Codes

Figure 41 on page 363 describes the return and reason codes for the PUT function.

Return Code	Description	
DESRC_SUCC	X'00' Successful processing	
	Reason Code	**Description**
	DESRS_SUCC	X'00' Successful processing
	Description	
DESRC_INFO 4	Not completely successful	
DESRC_WARN 8	Results questionable	
	Reason Code	**Description**
	DESRS_DST_ALREADY_EXISTS	X'44B' An EXIT exists and DESERV FUNC EXIT with NOREPLACE specified was issued. The current exit is not replaced
	DESRS_DST_COMP_SWAP_FAILED	X'451' An EXIT_OPTION=DELETE specified a DST address that was not current. The compare and swap failed.
DESRC_PARM 12	Missing/invalid parameters	
	Reason Code	**Description**
	DESRS_EXIT_OPTION_INVALID	X'44F' The EXIT_OPTION specified is not supported
	DESRS_EXIT_SCOPT_INVALID	X'45' The EXIT_SCOPE specified is not supported
	DESRS_EXIT_DST_PTR_ZERO	X'44C' The caller of DESERV FUNC=EXIT supplied a DST address of zero via the EXIT_DST parameter
	DESRS_INVAL_DST_HEADER	X'44D' The DST header is not correct
	DESRS_INVAL_PREVDST_HEADER	X'453' The DST header is not valid for the DST pointed to by DESP_PREV_DSTPTR_PTR. This is checked for EXIT_OPTION=DELETE
	DESRS_PREV_DSTPTR_PTR_ZERO	X'452' The pointer to the previous DST is zero. This is checked for EXIT_OPTION=DELETE
	DESRS_INVALID_PARM_LIST_HEADER	X'411' The id, length, or level of the DESP is not valid
	DESRS_UNSUPPORTED_FUNC	X'424' The FUNC value is incorrect
	DESRS_DEB_REQUIRES_AUTH	X'423' To pass the DEB the caller must be in supervisor state or a privileged key
	DESRS_INVALID_DCB_PTR	X'422' The address of the DCB is 0
	DESRS_DCB_NOT_OPEN	X'421' The passed DCB is not opened
	DESRS_INVALID_DEB_PTR	X'41E' Address of the DEB is 0 or DEB was input but the DCB pointed to by the DEB did not point back to the DEB
DESRC_CALR 16	Caller has a problem	
	Reason Code	**Description**
	DESRS_DEBCHK_FAILED	X'41D' The DEBCHK macro failed. The DCB or DEB was not valid
	DESRS_AUTH_ERROR	X'449' Caller not supervisor state or system key
DESRC_ENVR 20	Resources unavailable	
DESRC_IOER 24	I/O error	
DESRC_MEDE 28	Media error	
DESRC_DSLE 32	Data set logical error	
DESRC_SEVE 36	Severe error	
	Reason Code	**Description**
	DESRS_UNKNOWN	X'447' Issued by the DESERV recovery routine when entered for an unknown reason (ie. prog chk).
	DESRS_ADD_STACK_FAILED	X'437' Non-zero return code from an IGWFESTK request
	DESRS_SETLOOK_ERR	X'407' Bad return code from SETLOCK
	DESRS_EXTRACT_ERROR	X'406' IGWFTOKM EXTRACT failed
	DESRS_SET_ERROR	X'405' IGWFTOKM SET failed

DA6S3027

Figure 41. PUT return and reason codes

The DESD (DESERV member data descriptor) is the input to the DESERV PUT function. The DESD is an array consisting of the number of entries defined by the DESP field DESP_MEM_DATA2. The DESP_MEM_DATA_PTR points to the DESD and the DESD CSECT of the IGWDES macro maps it. The DESD structure is shown in Table 99 on page 364.

Table 99. DESD Parameter List

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	16	DESD	Member data descriptor (structure)
00 (X'00')	16	DESD_ENTRY	Entry descriptor (character)
00 (X'00')	1	DESD_FLAG	Flags (bitstring)
	1...	DESD_FLAG_ALIAS	Alias entry
1 (X'01')	1	DESD_CODE	Processing code (error=X'02', not processed =X'01', successful =X'00')(unsigned)
	X'02'	DESD_CODE_ERROR	Constant to be used with DESD_CODE
	X'01'	DESD_CODE_NOGO	Constant to be used with DESD_CODE
	X'00'	DESD_CODE_SUCC	Constant to be used with DESD_CODE
2 (X'02')	2	DESD_ERRCODE	Error code (low order halfword of DESERV reason code if error) (unsigned)
4 (X'04')	2	-	Reserved
6 (X'06')	2	DESD_DATA_LEN	Length of data area (unsigned)
08 (X'08')	4	DESD_DATA_PTR	Address of data (address)
12 (X'0C')	4	DESD_NAME_PTR	Address of varying length name (address)

The DESD_NAME_PTR points to the DESN structure. The DESD_DATA_PTR points to the directory entry for the program object being saved in the PDSE directory. The format of the directory entry is different depending on whether the DESD entry represents the primary name or an alias name. For the primary name, the DESD_DATA_PTR points to the CSECT PMAR mapped by IEWPMAR. For an alias name, the DESD_DATA_PTR points to the CSECT PMARA mapped by the macro IEWPMAR. Table 97 on page 360 shows the PMARA structure. The DESD_FLAG_ALIAS identifies the entry in the DESD as a primary or an alias.

The DESERV exit is passed to the current return and reason code that is to be passed back to the DESERV caller. The exit (either the pre-processing or the post-processing) can cause DESERV to return a different return and reason code to the DESERV caller by returning with a return code of 4 in register 15. If the exit returns control to DESERV with a return code of 4 in register 15, the (possibly modified) values of DESX_RETURN_CODE and DESX_REASON_CODE are returned to the caller of DESERV. If the pre-processing exit returns a return code of 0 in register 15, processing continues. Whereas if the post-processing exit returns with a return code of 0 in register 15, the original values of DESX_RETURN_CODE and DESX_REASON_CODE (those that were passed as input to the exit) are returned to the DESERV caller.

The exit processing can include interrogation of the DESERV parameter list (DESP) or interrogation or modification of the other interface structures and buffers. The DESP will not be modified by the DESERV exit. The exit is passed the caller key DESP, but DESERV has already made a key 5 copy of the DESP that is used for DESERV processing. Therefore modifications to the caller key DESP would not influence the DESERV processing. The DESERV interface structures are defined in the macro IGWDES. The macros IGWSMDE and IEWPMAR define the directory entry format.

Parameters Related to the DELETE Function

If the DESERV exit gets control for a DESERV DELETE function invocation, DESX_DESP_PTR points to the DESERV parameter list. If the DESP field DESP_FUNC=X'07' (DESP_FUNC_DELETE), this indicates a DELETE function parameter list. Table 100 on page 365 shows the fields of the DESP that pertain to the DESERV DELETE invocation.

Table 100. DESERV DELETE DESP Fields

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	104	DESP	DE Services parameter list (structure)
00 (X'00')	16	DESP_HEADER	Standard header (character)
00 (X'00')	8	DESP_ID	Eyecatcher IGWDESP (character)
08 (X'08')	4	DESP_LEN	Length of DESP (signed)
	X'04'	DESP_LEN_IV	Constant to be used with DESP_LEN
12 (X'0C')	1	DESP_LEV	Control block level (character)
	X'04'	DESP_LEV_IV	Constant to be used with DESP_LEV
13 (X'0D')	3	-	Reserved
16 (X'10')	1	DESP_FUNC	Function type (GET=X'01', PUT=X'04', DELETE=X'07', RENAME=X'08', UPDATE=X'09')
	X'09'	DESP_FUNC_UPDATE	Constant to be used with DESP_FUNC
	X'08'	DESP_FUNC_RENAME	Constant to be used with DESP_FUNC
	X'07'	DESP_FUNC_DELETE	Constant to be used with DESP_FUNC
	X'04'	DESP_FUNC_PUT	Constant to be used with DESP_FUNC
	X'01'	DESP_FUNC_GET	Constant to be used with DESP_FUNC
	X'00'	DESP_FUNC_OMITTED	Constant to be used with DESP_FUNC
17 (X'11')	3	-	Reserved
20 (X'14')	4	-	Reserved
24 (X'18')	12	DESP_DATA	Function data (character)
24 (X'18')	2	-	Reserved
26 (X'1A')	1	-	Reserved
27 (X'1B')	1	-	Reserved
28 (X'1C')	1	DESP_LIBTYPE	Indicates whether a DCB or a DEB in input. (DCB input X'02', DEB input X'01') (unsigned)
	X'02'	DESP_LIBTYPE_DCB	Constant to be used with DESP_LIBTYPE
	X'01'	DESP_LIBTYPE_DEB	Constant to be used with DESP_LIBTYPE
	X'00'	DESP_LIBTYPE_OMITTED	Constant to be used with DESP_LIBTYPE
29 (X'1D')	1	-	Reserved
30 (X'1E')	1	-	Reserved
31 (X'1F')	1	-	Reserved
32 (X'20')	1	-	Reserved
33 (X'21')	1	-	Reserved
34 (X'22')	1	-	Reserved
35 (X'23')	1	-	Reserved
36 (X'24')	4	DESP_DCB_PTR	DCB address
40 (X'28')	4	DESP_DEB_PTR	DEB address
44 (X'2C')	4	-	Reserved

Table 100. DESERV DELETE DESP Fields (continued)

Offset	Length or Bit Pattern	Name	Description
48 (X'30')	4	-	Reserved
52 (X'34')	4	-	Reserved
56 (X'38')	4	-	Reserved
60 (X'3C')	4	-	Reserved
64 (X'40')	4	-	Reserved
68 (X'44')	4	-	Reserved
72 (X'48')	4	-	Reserved
76 (X'4C')	4	-	Reserved
80 (X'50')	4	DESP_NAME_LIST	List of names to be deleted (DESL) (address)
84 (X'54')	4	DESP_NAME_LIST2	Number of entries in DESL name list (unsigned)
88 (X'58')	4	-	Reserved
92 (X'5C')	4	-	Reserved
96 (X'60')	4	-	Reserved
100 (X'64')	4	-	Reserved

The DESL points to the names to be deleted. The DESP_NAME_LIST_PTR points to the DESL and the DESL DSECT in the IGWDES macro maps it. A DESL is an array consisting of the number of entries the DESP field DESP_NAME_LIST2 defines. The DESL structure is shown in Table 86 on page 351.

Parameters Related to the RENAME Function

If the DESERV exit gets control for a DESERV RENAME function invocation, DESX_DESP_PTR points to the DESERV parameter list. If the DESP field DESP_FUNC=X'08' (DESP_FUNC_RENAME), this indicates a RENAME function parameter list. Table 101 on page 366 shows the fields of the DESP that pertain to the DESERV RENAME invocation.

Table 101. DESERV RENAME DESP Fields

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	104	DESP	DE Services parameter list (structure)
00 (X'00')	16	DESP_HEADER	Standard header (character)
00 (X'00')	8	DESP_ID	Eyecatcher IGWDESP (character)
08 (X'08')	4	DESP_LEN	Length of DESP (signed)
	X'04'	DESP_LEN_IV	Constant to be used with DESP_LEN
12 (X'0C')	1	DESP_LEV	Control block level (character)
	X'04'	DESP_LEV_IV	Constant to be used with DESP_LEV
13 (X'0D')	3	-	Reserved

Table 101. DESERV RENAME DESP Fields (continued)

Offset	Length or Bit Pattern	Name	Description
16 (X'10')	1	DESP_FUNC	Function type (GET=X'01', PUT=X'04', DELETE=X'07', RENAME=X'08', UPDATE=X'09')
	X'09'	DESP_FUNC_UPDATE	Constant to be used with DESP_FUNC
	X'08'	DESP_FUNC_RENAME	Constant to be used with DESP_FUNC
	X'07'	DESP_FUNC_DELETE	Constant to be used with DESP_FUNC
	X'04'	DESP_FUNC_PUT	Constant to be used with DESP_FUNC
	X'01'	DESP_FUNC_GET	Constant to be used with DESP_FUNC
	X'00'	DESP_FUNC_OMITTED	Constant to be used with DESP_FUNC
17 (X'11')	3	-	Reserved
20 (X'14')	4	-	Reserved
24 (X'18')	12	DESP_DATA	Function data (character)
24 (X'18')	2	-	Reserved
26 (X'1A')	1	-	Reserved
27 (X'1B')	1	-	Reserved
28 (X'1C')	1	DESP_LIBTYPE	Indicates whether a DCB or a DEB is input. (DCB,DEB) (DCB input X'02', DEB input X'01') (unsigned)
	X'02'	DESP_LIBTYPE_DCB	Constant to be used with DESP_LIBTYPE
	X'01'	DESP_LIBTYPE_DEB	Constant to be used with DESP_LIBTYPE
	X'00'	DESP_LIBTYPE_OMITTED	Constant to be used with DESP_LIBTYPE
29 (X'1D')	1	-	Reserved
30 (X'1E')	1	-	Reserved
31 (X'1F')	1	-	Reserved
32 (X'20')	1	-	Reserved
33 (X'21')	1	-	Reserved
34 (X'22')	1	-	Reserved
35 (X'23')	1	-	Reserved
36 (X'24')	4	DESP_DCB_PTR	DCB address
40 (X'28')	4	DESP_DEB_PTR	DEB address
44 (X'2C')	4	-	Reserved
48 (X'30')	4	-	Reserved
52 (X'34')	4	-	Reserved
56 (X'38')	4	-	Reserved
60 (X'3C')	4	-	Reserved
64 (X'40')	4	-	Reserved
68 (X'44')	4	-	Reserved
72 (X'48')	4	-	Reserved
76 (X'4C')	4	-	Reserved

Table 101. DESERV RENAME DESP Fields (continued)

Offset	Length or Bit Pattern	Name	Description
80 (X'50')	4	DESP_NAME_LIST	List of pairs of names for rename operations (DESL) (address)
84 (X'54')	4	DESP_NAME_LIST2	Number of entries in DESL name list (unsigned)
88 (X'58')	4	-	Reserved
92 (X'5C')	4	-	Reserved
96 (X'60')	4	-	Reserved
100 (X'64')	4	-	Reserved

The DESL points to the names that are to be renamed and to the new names. The DESP_NAME_LIST_PTR points to the DESL and the DESL DSECT in the IGWDES macro maps it. A DESL is an array consisting of the number of entries the DESP field DESP_NAME_LIST2 defines. The DESL structure is shown in Table 86 on page 351.

Parameters Related to the UPDATE Function

If the DESERV exit gets control for a DESERV UPDATE function invocation, DESX_DESP_PTR points to the DESERV parameter list. If the DESP field DESP_FUNC=X'09' (DESP_FUNC_UPDATE), this indicates an UPDATE function parameter list. Table 102 on page 368 shows the fields of the DESP that pertain to the DESERV UPDATE invocation.

Table 102. DESERV UPDATE DESP Fields

Offset	Length or Bit Pattern	Name	Description
00 (X'00')	104	DESP	DE Services parameter list (structure)
00 (X'00')	16	DESP_HEADER	Standard header (character)
00 (X'00')	8	DESP_ID	Eyecatcher 'IGWDESP' (character)
08 (X'08')	4	DESP_LEN	Length of DESP (signed)
	X'04'	DESP_LEN_IV	Constant to be used with DESP_LEN
12 (X'0C')	1	DESP_LEV	Control block level (character)
	X'04'	DESP_LEV_IV	Constant to be used with DESP_LEV
13 (X'0D')	3	-	Reserved
16 (X'10')	1	DESP_FUNC	Function type (GET=X'01', PUT=X'04', DELETE=X'07', RENAME=X'08', UPDATE=X'09')
	X'09'	DESP_FUNC_UPDATE	Constant to be used with DESP_FUNC
	X'08'	DESP_FUNC_RENAME	Constant to be used with DESP_FUNC
	X'07'	DESP_FUNC_DELETE	Constant to be used with DESP_FUNC
	X'04'	DESP_FUNC_PUT	Constant to be used with DESP_FUNC
	X'01'	DESP_FUNC_GET	Constant to be used with DESP_FUNC
	X'00'	DESP_FUNC_OMITTED	Constant to be used with DESP_FUNC
17 (X'11')	3	-	Reserved
20 (X'14')	4	-	Reserved
24 (X'18')	12	DESP_DATA	Function data (character)

Table 102. DESERV UPDATE DESP Fields (continued)

Offset	Length or Bit Pattern	Name	Description
24 (X'18')	2	-	Reserved
26 (X'1A')	1	-	Reserved
27 (X'1B')	1	-	Reserved
28 (X'1C')	1	DESP_LIBTYPE	Function subtype (DCB,DEB) (DCB input X'02', DEB input X'01') (unsigned)
	X'02'	DESP_LIBTYPE_DCB	Constant to be used with DESP_LIBTYPE
	X'01'	DESP_LIBTYPE_DEB	Constant to be used with DESP_LIBTYPE
	X'00'	DESP_LIBTYPE_OMITTED	Constant to be used with DESP_LIBTYPE
29 (X'1D')	1	-	Reserved
30 (X'1E')	1	-	Reserved
31 (X'1F')	1	-	Reserved
32 (X'20')	1	-	Reserved
33 (X'21')	1	-	Reserved
34 (X'22')	1	-	Reserved
35 (X'23')	1	-	Reserved
36 (X'24')	4	DESP_DCB_PTR	DCB address
40 (X'28')	4	DESP_DEB_PTR	DEB address
44 (X'2C')	4	-	Reserved
48 (X'30')	4	-	Reserved
52 (X'34')	4	-	Reserved
56 (X'38')	4	-	Reserved
60 (X'3C')	4	-	Reserved
64 (X'40')	4	-	Reserved
68 (X'44')	4	-	Reserved
72 (X'48')	4	-	Reserved
76 (X'4C')	4	-	Reserved
80 (X'50')	4	DESP_NAME_LIST	List of SMDEs with imbedded PMAR (and PMARL) to update the directory information (DESL) (address)
84 (X'54')	4	DESP_NAME_LIST2	Number of entries in DESL name list (unsigned)
88 (X'58')	4	-	Reserved
92 (X'5C')	4	-	Reserved
96 (X'60')	4	-	Reserved
100 (X'64')	4	-	Reserved

The DESL points to the SMDEs which are to be updated. The DESP_NAME_LIST_PTR points to the DESL and the DESL DSECT in the IGWDES macro maps it. A DESL is an array consisting of the number of entries the DESP field DESP_NAME_LIST2 defines. The DESL structure is shown in Table 86 on page 351.

Entry Environment for Exit Routine

Interrupts:
Enabled

State and Key:
Supervisor state and key 0

ASC Mode:
P=H=S

AMODE, RMODE:
AMODE=31, RMODE=ANY

LOCKS:
No locks held

Registers:

0
unpredictable

1
address of DESERV EXIT parameter list mapped by DSECT DESX in IGWDES.

2-12
unpredictable

13
eighteen word save area (key 0).

14
return address

15
entry point address of exit routine

The exit is entered in TASK mode and the DESERV recovery environment is an ESTAE; therefore, the exit routine can issue SVCs, if required.

Exit Environment for Exit routine

Interrupts:
Enabled

State and Key:
Supervisor state and key 0

ASC Mode:
P=H=S

AMODE, RMODE:
AMODE=31, RMODE=ANY

LOCKS:
No locks held

Registers:

Unless otherwise specified, all registers must be restored to their contents on entry.

15
Return code

- For pre-processing exit:

  R15 = 0
  Continue processing of this DESERV call.

R15 = 4
Discontinue processing of this DESERV call. Control is immediately returned to the caller of DESERV with the return and reason codes as set by the exit in the DESX fields DESX_RETURN_CODE and DESX_REASON_CODE.

- For post-processing exit:

  R15 = 0
  Continue processing of this DESERV call (that is, return to the caller of DESERV).

  R15 = 4
  Control is returned to the caller of DESERV with the return and reason codes as set by the exit in the DESX fields DESX_RETURN_CODE and DESX_REASON_CODE. Values for DESX_RETURN_CODE and DESX_REASON_CODE are described in the macro IGWDES. The reason code structure is such that the first two bytes are system component id and module id (that is, system diagnostic information). The low order two bytes contain the real reason code as indicated in the macro IGWDES. For additional return and reason codes for the GET, RENAME, DELETE, and UPDATE return and reason codes, refer to *z/OS DFSMS Macro Instructions for Data Sets*.

- **Restriction**: Any other return code from the exits causes DESERV code to take an SVC dump.

Registers on Entry to the DESERV Exit

When your DESERV exit gets control, the general-purpose registers have the following content:

Register
Contents

0
not applicable

1
address of DESX

2-12
not applicable

13
address of register save area

14
return address

15
address of DESERV exit entry point

Registers on Return from the DESERV Exit

When you return control to DESERV, the register contents must be set up as follows:

Register
Contents

0-14
restored to contents at entry

15
return code

DESERV Exit Return and Reason Codes

Return Code
Description

00 (X'00')
Continue with DESERV function

04 (X'04')
Return immediately to the DESERV caller and return the return and reason codes as defined by DESX_RETURN_CODE and DESX_REASON_CODE respectively.

DESERV FUNC=EXIT Return and Reason Codes

The formats of the return and reason codes are:

Offset/length
Description

00 (X'00') 1 byte
SMS Component code (X'27') indicates Common Adaptor (of which DESERV is a part)

01 (X'01') 1 byte
Module ID - used for problem diagnosis

02 (X'02') 2 bytes
Reason code - identifies the error. A program testing the DESERV reason code should only look at these last two bytes. The component id and module id should not be tested. They are reported for diagnostic purposes only.

The following are the two low order byte values for the reason codes that DESERV FUNC=EXIT might return (sorted by return code).

Table 103. Return and Reason Codes for the Exit DESERV Function

Return Code	Reason Code	Symbolic name	Description
X'0'		DESRC_SUCC	Successful
	X'00'	DESRS_SUCC	Successful
X'4'		DESRC_INFO	Not completely successful.
	X'400'	DESRS_NAME_ NOT_DEFINED	Name to be replaced did not previously exist
X'8'		DESRC_WARN	Results questionable
X'12'		DESRC_PARM	Missing or invalid parameters
	X'411'	DESRS_INVALID_ PARM_LIST_HEADER	The id, length, or level of the DESP is not valid
	X'415'	DESRS_PDS_ NOT_SUPPORTED	This function requires a PDSE data set
	X'41E'	DESRS_INVALID_ DEB_PTR	Address of the DEB is 0 or DEB was input but the DCB pointed to by the DEB did not point back to the DEB
	X'41F'	DESRS_DCB_NOT_ OPEN_OUTPUT	With function PUT the DCB must have been opened for output
	X'421'	DESRS_DCB_ NOT_OPEN	The passed DCB is not opened
	X'422'	DESRS_INVALID_ DCB_PTR	The address of the DCB is 0
	X'423'	DESRS_DEB_ REQUIRES_AUTH	To pass the DEB, the caller must be in supervisor state or a privileged key
	X'424'	DESRS_ UNSUPPORTED_FUNC	The FUNC value is incorrect

Table 103. Return and Reason Codes for the Exit DESERV Function (continued)

Return Code	Reason Code	Symbolic name	Description
	X'427'	DESRS_INVALID_ MEM_DATA_CNT	The count of entries in the MEM_DATA block is 0
	X'428'	DESRS_INVALID_ MEM_DATA_PTR	The address of the MEM_DATA block is 0
	X'429'	DESRS_INVALID_ PUT_OPTION	The PUT function requires that the OPTION field be specified
X'16'		DESRC_CALR	Caller has a problem
	X'3FD'	DESRS_PRI_ NM_THIS_FILE	Alias name is same name as primary name for this member
	X'400'	DESRS_NAME_ ALREADY_EXISTS	Name to be replaced did not previously exist
	X'40E'	DESRS_NAME_ ALREADY_EXISTS	The PUT failed because of a name conflict
	X'40F'	DESRS_NP_ PRIMARY_NAME	The MEM_DATA must have one member designated as primary
	X'410'	DESRS_INVALID_ NAME_PREFIX	The first 8 bytes of a name were all X'FF'
	X'412'	DESRS_MORE_ THAN_! _PRIMARY	The MEM_DATA must have only one member designated as primary
	X'413'	DESRS_ INVALID_MLT	MLT is not valid
	X'414'	DESRS_ INVALID_CT	Connect token is not valid
	X'41D'	DESRS_DEBCHK_ FAILED	The DEBCHK macro failed, the DCB or DEB was not valid
	X'425'	DESRS_INVALID_ NAME_LENGTH	The length of an alias name was either 0 or greater than 8
	X'43A'	DESRS_DATA_ LENGTH_ERROR	The DESD data length is not valid, data length must be greater than 0 and less than 108 bytes
	X'43C'	DESRS_NAME_ IS_PRIMARY_NAME	The alias name specified is a primary name and the options did not allow for deleting primary name
X'20'		DESRC_ENVR	Resources unavailable
X'24'		DESRC_IOER	I/O error
X'28'		DESRC_MEDE	Media Error
X'32'		DESRC_DSLE	Data Set logical error
X'36'		DESRC_SEVE	Severe error
	X'407'	DESRS_ SETLOOK_ERR	Bad return code from SETLOCK
	X'437'	DESRS_ADD_ STACK_FAILED	Non-zero return code from an IGWFESTK request

Table 103. Return and Reason Codes for the Exit DESERV Function (continued)			
Return Code	**Reason Code**	**Symbolic name**	**Description**
	X'447'	DESRS_UNKNOWN	Issued by the DESERV recovery routine when entered for an unknown reason (for example, a program check) while the exit routine was in control. Most likely an exit error.
X'0C'	X'469'	DESRS_DST_EXIT_PTR_ NOT_COMMON	Global exit address is not in common storage.

Additional Return and Reason Codes: For the GET, RENAME, DELETE, and UPDATE return and reason codes, refer to *z/OS DFSMS Macro Instructions for Data Sets*.

Example of the DESERV Exit

The following program segment establishes and deletes the task mode DESERV exit and thereby restores the previous task. The sample is generic enough to apply to either the global or the task level exit but shows here the task level support. When establishing a global exit, obtain the DST storage in common storage.

Establishing and Deleting a Task Level DESERV Exit Part 1 of 2

```
SAMPLE   CSECT
       USING  *,12
       STM    14,12,12(13)      SAVE REGISTERS
       LR     12,15             ESTABLISH BASE REGISTER
       LA     2,SAVE            ADDRESS REGISTER SAVE AREA
       ST     2,8(13)           FORWARD CHAIN SAVE AREA
       ST     13,SAVE+4         BACKWARD CHAIN SAVE AREA
       LR     13,2              ESTABLISH SAVE AREA
       .
       .
       .
*
*
* ESTABLISH THE TASK LEVEL EXIT
*
*     BUILD THE DST TO REPRESENT MY TASK LEVEL EXIT
*
       LA     3,MY_DST             ADDRESSABILITY TO MY DST
       USING  DST,3                MAP DST
       XC     DST,DST              CLEAR MY DST STORAGE
       MVC    DST_ID,DST_ID_CONST  SET EYECATCHER IN DST
       LA     15,DST_LEN_IV(,0)    GET LENGTH OF DST
       ST     15,DST_LEN           SET LENGTH OF DST
       MVI    DST_LEV,DST_LEV_IV   SET LEVEL OF DST USED
       OI     DST_FLAGS,DST_FLAGS_PROP  SET FLAG TO PROPAGATE THIS      EXIT
*                                  TO ATTACHED TASKS.
       MVC    DST_EXIT_PTR,ADDR_DESEXIT  SET ADDRESS OF TASK EXIT
*
*     CALL DESERV TO ENABLE MY TASK LEVEL EXIT
*        THIS MODULE ASSUMES THAT IT IS RUNNING EITHER SUPERVISOR STATE
*        OR SYSTEM KEY.
*
       DESERV FUNC=EXIT,                                              *
              EXIT_SCOPE=TASK,                                        *
              EXIT_OPTION=REPLACE,                                    *
              EXIT_DST=MY_DST,                                        *
              EXIT_PREV_DSTPTR=MY_PREV_DSTPTR,                        *
              MF=S
       .
       .
       .
*
* QUERY THE CURRENT DST ADDRESS, RETURNED IN CURRENT_DSTPTR.
* THERE IS NO NEED TO QUERY PRIOR TO DOING THE DELETE, THIS IS
* HERE JUST TO SHOW THE INVOCATION SYNTAX.
*
```

```
       DESERV FUNC=EXIT,                                                *
              EXIT_SCOPE=TASK,                                          *
              EXIT_OPTION=QUERY,                                        *
              EXIT_PREV_DSTPTR=CURRENT_DSTPTR,                          *
              MF=S
```

Establishing and Deleting a Task Level DESERV Exit Part 2 of 2

```
*
*    HERE THE APPLICATION WOULD DO SOMETHING TO
*    CAUSE A DESERV CALL TO BE DONE. AN EXAMPLE OF THIS TYPE OF THING
*    WOULD BE ATTACHING THE BINDER AS A SUBTASK (THE BINDER USES
*    DESERV.)
*
       .
       .
       .
*
* DELETE THE TASK LEVEL EXIT
*
       DESERV FUNC=EXIT,                                            *
              EXIT_SCOPE=TASK,                                      *
              EXIT_OPTION=DELETE,                                   *
              EXIT_DST=MY_DST,                                      *
              EXIT_PREV_DSTPTR=MY_PREV_DSTPTR,                      *
              MF=S
       .
       .
       .
*
* RESTORE REGISTERS AND RETURN
       L      13,SAVE+4
       LM     14,12,12(13)
       SR     15,15          SET RETURN CODE
       BR     14             RETURN TO CALLER
* THE FOLLOWING IS A BLOCK USED BY THIS APPLICATION. THIS BLOCK CAN
* BE USED BY THE EXIT. THE EXIT ALSO (SMARTLY) FINDS THE
* PREV_DST_PTR FIELD IN MY_BLOCK WHEN IT GIVES CONTROL TO THE
* PREVIOUSLY ESTABLISHED TASK LEVEL EXIT.
MY_BLOCK        DS  0D             MY APPLICATION BLOCK
MY_APPLICATION_STUFF DS  CL8       MY APPLICATION STUFF
MY_DST          DS  CL(DST_LEN_IV) MY DST IMBEDDED IN MY BLOCK
MY_PREV_DSTPTR  DS  F              ADDRESS OF PREVIOUS TASK LEVEL
                EXTRN   DESEXIT
ADDR_DESEXIT    DC  A(DESEXIT)     ADDRESS OF MY TASK LEVEL EXIT
DST_ID_CONST    DC  CL8'IGWDST  '  DST EYECATCHER
*                                   DST
CURRENT_DSTPTR  DS  F              ADDRESS OF CURRENT TASK LEVEL
*                                   DST
SAVE            DS  18F            REGISTER SAVE AREA
                END
```

The following program segment shows a sample DESERV exit. This is the exit established by the previous segment of code. This sample shows how to pass control to a previous exit. Note that this sample exit is not reentrant, so it assumes there is only one subtask.

Sample DESERV Exit Routine Part 1 of 3

```
DESEXIT CSECT
DESEXIT AMODE  31               Must be AMODE 31
DESEXIT RMODE  ANY              Could be RMODE 24 if required
*
*  entry code to save registers and establish base register.
        USING  *,12
        STM    14,12,12(13)     SAVE REGISTERS
        LR     12,15            ESTABLISH BASE REGISTER
        LA     2,SAVE           ADDRESS REGISTER SAVE AREA
        ST     2,8(13)          FORWARD CHAIN SAVE AREA
        ST     13,SAVE+4        BACKWARD CHAIN SAVE AREA
        LR     13,2             ESTABLISH SAVE AREA
*
* Assume this exit is only interested in the output from GET functions.
* Therefore ignore entry for all other functions and only
* process the get function invocations where get processing is complete
*
        SLR    2,2              clear reg
        ST     2,EXIT_RC        initialize return code
        LR     2,1              get the DESX address in reg 2
```

```
        USING  DESX,2            map the DESX
        L      3,DESX_DST_PTR    get address of DST
        LR     4,3               get address of DST
        LA     5,MY_DST-MY_BLOCK(,0) get offset to DST within MY_BLOCK
        SR     4,5               get the address of MY_BLOCK
        USING  MY_BLOCK,4        map MY_BLOCK
*
* First give control to any previously established DESERV
* exits. If the value in PREV_DST_PTR is zero, then there was no
* previous DST (that is, no previous DESERV exit). PREV_DST_PTR was
* saved in MY_BLOCK by the SAMPLE CSECT that enabled this
* exit.
        SR     15,15             simulate previous exit's return code
        ICM    5,15,PREV_DST_PTR get previous DST address
        BZ     NOPREVDST         branch if zero, no previous DST
* There was a previous exit, to which you transfer control.
* First build a DESX then branch to the previous exit.
*
*
* Getmain dynamic storage for interfacing with the other exit
*
*
        GETMAIN RU,LV=DESX_LEN_IV,SP=230,KEY=0,LOC=(ANY)
        ST     1,MY_DESX_STG_PTR
        MVC    0(L'DESX,1),DESX  copy the DESX that was input to this
*                                 routine.
        ST     5,DESX_DST_PTR-DESX(,1) Set previous DST address in DESX
        L      15,DST_EXIT_PTR-DST(,5)  get address of previous exit
        BALR   14,15             call the previous exit in 31 bit amode
```

Sample DESERV Exit Routine Part 2 of 3

```
NOPREVDST EQU  *
        ST     15,EXIT_RC        save previous exit's return code
* If this invocation of the exit was the pre-processing exit and the
* exit we called just returned with a return code of 4 then they
* might have returned the data in question.
        TM     DESX_FLAGS,DESX_POST_BIT is this post-processing call
        BO     POST              yes, branch
* this is a pre-processing call, did the exit just called request an
* immediate return, and therefore maybe fill in the output fields
* if the return code was zero, the called routine wants deserv to
* perform the function.
        CLC    EXIT_RC,ZERO      was return code zero
        BE     RETURN            yes, branch to return to deserv
POST    EQU    *                 either called exit performed the
*                                 deserv function, or this is a post
*                                 processing exit call.
        L      6,DESX_DESP_PTR   get the DESERV parameter list (DESP)
        USING  DESP,6            map DESP
        CLI    DESP_FUNC,DESP_FUNC_GET is this a function GET call
        BNE    RETURN            no, branch
        CLC    DESX_REASON_CODE+2(2),=AL2(DESRS_SUCC) check low order
*                                 halfword of reason code for success
        BE     GOODDATA          branch if DESERV GET was successful
        CLC    DESX_REASON_CODE+2(2),=AL2(DESRS_NOTFOUND) check for
*                                 some members not found
        BNE    BADDATA           branch if some other error
GOODDATA EQU   *
*
* Process the entries in the names list that were found
*
        .
        .
        .
        B      RETURN            return to the caller (DESERV)
BADDATA EQU    *
        .
        .
        .
*
RETURN  EQU    *                  Return to caller (DESERV)
*
*  restore registers, FREEMAIN storage, etc.
*
        L      15,EXIT_RC         set return code
        L      13,SAVE+4
        L      14,12(13)          restore return address
        LM     0,12,20(13)        restore callers other registers
        BR     14                 return to deserv
```

Sample DESERV Exit Routine Part 3 of 3

```
*
* CONTROL INFORMATION
*
SAVE            DS  18F            REGISTER SAVE AREA
ZERO            DC  F'0'           constant of zero
EXIT_RC         DC  F'0'           Return code to pass back to DESERV
DESX_ID_CONST   DC  CL8'IGWDESX '  EYECATCHER FOR DESX
MY_DESX_STG_PTR DC  F'0'           ADDRESS OF GETMAINED STORAGE FOR
*                                   DESX TO PASS TO PREVIOUS EXIT.
MY_BLOCK        DSECT              MY APPLICATION BLOCK
MY_APPLICATION_STUFF DS CL8        MY APPLICATION STUFF
MY_DST          DS  CL(DST_LEN_IV) MY DST IMBEDED IN MY BLOCK
PREV_DST_PTR    DS  F              ADDRESS OF PREVIOUS TASK LEVEL EXIT
                IGWDES             DESERV MAPPINGS
                IGWSMDE            DIRECTORY ENTRY
                IEWPMAR            PROGRAM MANAGEMENT ATTRIBUTE RECORD
*                                  THE PMAR IS A SUB RECORD OF THE SMDE
                END
```

Chapter 11. Managing Hierarchical File System Data Sets

A hierarchical file system (HFS) data set is a data set that contains a POSIX-compliant hierarchical file system, which is a collection of files and directories that are organized in a hierarchical structure that can be accessed by using z/OS UNIX System Services . You can use many of the standard BSAM, QSAM, BPAM, and VSAM interfaces to access data in z/OS UNIX HFS files. Most applications that use these access methods can access HFS files without reassembly or recompilation.

The contents of HFS data sets are structured like a tree, based on a root directory with various subdirectories. The files within an HFS data set are identified by their path and file names.

DFSMShsm can automatically back up HFS data sets if it is using DFSMSdss as its data mover, but it cannot back up individual files within an HFS data set.

Once you evaluate z/OS UNIX and decide to install it in your enterprise, you may proceed with the following planning tasks:

- Deciding which DASD devices will contain the HFS data sets.
- Deciding how to control access to them.
- Structuring the file systems.
- Determining backup, restore, and expiration date policies.
- Determining HFS naming policies (file names can be up to 255 characters long and path names can be up to 1023 characters).
- Defining data classes, management classes, and coding ACS routines and JCL statements for HFS data sets.

Creating Hierarchical File System Data Sets

You specify HFS in the DSNTYPE parameter to allocate an HFS data set. You can also define a data class for HFS data sets. Both cataloged and uncataloged HFS data sets can reside on a single non-SMS-managed volume. A cataloged HFS data set can also be a multivolume if it resides on Storage Management Subsystem managed volumes. This type of data set can expand to as many as 255 extents of direct access storage device (DASD) space on multiple volumes (59 volumes maximum with 123 extents per volume).

RACF or an equivalent security product must be installed and active on your system to use z/OS UNIX or HFS data sets. z/OS UNIX maintains system security by verifying user identities and file access control information.

This information covers the following:

- “Defining the Root File System” on page 380
- “Creating and Mounting the Root File System” on page 380
- “Creating Additional File Systems and Directories” on page 380
- “Adding and Mounting File Systems to the Root File System” on page 380

Defining the Root File System

During installation of z/OS UNIX, a system programmer or storage administrator codes the ROOT statement in the BPXPRMxx member of SYS1.PARMLIB. This identifies the HFS data set containing the root file system for the system to logically mount when it starts z/OS UNIX system services.

```
ROOT   FILESYSTEM('OMVS.ROOT')
       TYPE(HFS)
       MODE(RDWR)
```

The root file system is the starting point of the overall file structure. It consists of the root directory and any related HFS files or subdirectories. After installation is complete and the MVS system is running, you can create (allocate) an HFS data set, which contains the root file system.

See *z/OS MVS Initialization and Tuning Reference* for information about member BPXPRM*xx*.

Creating and Mounting the Root File System

Create an HFS data set to contain the root file system by running a job to allocate the data set. During allocation, z/OS UNIX builds a basic root directory, which you can alter to meet your specific needs. You specify DSNTYPE=HFS to designate an HFS data set.

```
//STEP1  EXEC PGM=IEFBR14
//MKFS   DD   DSNAME=OMVS.ROOT,DISP=(NEW,CATLG),
//            DSNTYPE=HFS,SPACE=(CYL,(1,1,1))
```

Tip: To make the HFS data set SMS-managed, use the ACS routines or specify the STORCLAS parameter in the JCL. HFS data sets do not have to be SMS-managed.

Write operations present the greatest exposure to file system damage. For this reason the root file system should be small, minimizing the amount of write activity to it, thus offering the least exposure to damage.

Additionally, if all users' files are in file systems that are mounted on the root file system, rather than defined as part of the root itself, and users are denied write access to the root, the root is further protected from inadvertent damage. Damaged user directories or files can be unmounted and replaced without causing z/OS UNIX system services to fail.

Creating Additional File Systems and Directories

After allocating an HFS data set for the root file system and logging on as a TSO/E user, you can define additional directories in the root file system using the MKDIR command. For example, to create the /u/joe directory, issue:

```
MKDIR ‘/u’
MKDIR ‘/u/joe’
```

These directories can be used as mount points for additional mountable file systems. You can use an IBM-supplied program that creates directories, pseudo-TTY pairs, and device files. Interactive users and application programs can then add files to those additional file systems.

The MKDIR command requires written permission on the parent directory of the directory to be created.

Adding and Mounting File Systems to the Root File System

If you have appropriate authority, you can create other mountable file systems with their own directory and data file structures, and mount them on a directory in the root file system or in another file system. Each file system can be logically mounted to a directory (mount point) in another file system by using the TSO/E MOUNT command. Use the UNMOUNT command to unmount a file system. To create and mount additional file systems:

1. Allocate an additional HFS data set by using either the TSO/E ALLOCATE command as shown in the following example or a JCL DD statement similar to that shown in "Creating and Mounting the Root File System" on page 380.

```
ALLOCATE DSNAME('OMVS.USER.JOE') NEW DSNTYPE(HFS) BLKSIZE(0)
  LRECL(0) RECFM(U) DSORG(PO) SPACE(1,1) CYLINDERS
```

   The new data set is allocated with an empty root directory.
2. Have an authorized user enter a TSO/E MOUNT command to logically mount the new file system in the directory of an existing file system.

```
MOUNT FILESYSTEM('OMVS.USER.JOE') TYPE(HFS) MOUNTPOINT('/u/joe')
```

You can specify additional file systems to be logically mounted automatically every time z/OS UNIX is started by adding MOUNT commands to the BPXPRMxx member of SYS1.PARMLIB. The following restrictions apply to mounting file systems:

- The mount point must be a directory.
- Any files in the directory are not accessible while the file system is mounted.
- Only one mount can be active at any time for a mount point.
- A file system can be mounted on only one directory at a time.

You can also create special HFS files to perform the following tasks:

- Represent hardware devices (character special files).
- Allow the use of alias names for HFS files (symbolic links).
- Send data from one process to another so that the receiving process reads the data first-in-first-out (FIFO special files, also called named pipes). The term process as used here is defined as either a program that is created by the fork function, or a program that requests z/OS UNIX system services.

Managing File System Size

File system size increases as users add files and extend existing files. Eventually, a file system can outgrow the space on its volume. In this case, the storage administrator or system programmer responsible for HFS data sets can either make more space available on the volume by moving individual HFS files to other file systems that have space available, or do one of the following:

- Move the entire file system to another set of volumes as follows:
  1. Have an authorized user enter a TSO/E UNMOUNT command to logically unmount the file system.
  2. Allocate an HFS data set with a different data set name on a volume, or set of volumes, that has adequate space available.
  3. Use the DFSMSdss DUMP function to logically dump the old file system.
  4. Use the DFSMSdss RESTORE function to restore the dumped file system with a new name to a volume, or set of volumes, that has sufficient space. If you want to maintain the original file system name, delete the existing file system first, and then restore it using DFSMSdss without renaming it.
- Remove files from the file system by either deleting them or by moving them to another file system. If it is impossible to remove the chosen files from a particular directory in the file system, it may be possible to remove other files from a different directory in the same file system. The objective is to reduce the size of the file system.
- Create a new file system on another volume, or set of volumes, and move some files from the full file system to the new file system. To avoid problems that can result from this approach, define symbolic links using the original names.
- Add another volume to the file system candidate volume list with the IDCAMS ALTER ADDVOLUMES command. The file system must be unmounted and remounted for the additional volumes to be usable by the HFS.

- The storage administrator or system programmer can monitor the space in a file system by mounting an HFS with parm FSFULL. For example, mount parm('FSFULL(70,10)') will cause HFS to issue message IGW023A when the file system is 70 percent full and then issues an additional IGW023I messages when the file system is 80 and 90 percent full.

The BPX1PCT callable service can be used to extend the file system (see "Using pfsctl (BPX1PCT) Physical File System Control for HFS" on page 384.

Managing File System Activity

If activity for a file system becomes so extensive that accesses are slow, you can do one of the following tasks:

- Move the file system to a volume that processes I/O more quickly, for example, a volume that has a faster channel or a cached storage control.
- Move a subtree from the full file system into a new file system on a different volume. Mount the new file system on the now-empty directory, which was the head of the subtree. This divides I/O activity between two volumes. To avoid failures from this action, define symbolic links using the original names.

Accessing HFS Data Set Attributes

Any application (for example, Interactive System Productivity Facility (ISPF) option 3.4I or DCOLLECT) that requests size information about HFS file system requires OMVS to service that request. This means that the user ID must be defined to OMVS and have Resource Access Control Facility (RACF) authority to access the data set. The HFS file system must be either mounted on this file system or mountable (that is, not already mounted READ/WRITE on another system).

Transporting a File System

You might also want to copy a data set to a storage medium that can be physically transported to another location. In order to do that, perform one of the following tasks:

- Use either the PAX, CPIO, or TAR shell commands to copy the file system in tape archive (TAR) format.
- Have an authorized user logically unmount the file system, allocate an HFS data set with a different data set name, and use the DFSMSdss dump utility to copy the old file system to the new data set.

Removing (Deleting) a File System

If you mount the file system (HFS data set) to a different file system or you want to delete it, first logically unmount it using the TSO/E UNMOUNT command against the HFS data set containing it. Then you can use either or both of the MOUNT commands (see "Adding and Mounting File Systems to the Root File System" on page 380).

You can eliminate the file system by using any of the following techniques:

- Use the DELETE command (IDCAMS or TSO).
- Execute an IEFBR14 job with DISP=(OLD,DELETE) specified for the HFS data set.
- ISPF option 3.x.
- Use the SCRATCH and UNCATLG commands of IEHPROGM.

Migrating a File System

If a file system is unmounted and remains so for a predetermined time, the system can migrate it to a lower priority storage medium. The system automatically recalls a migrated file system from migration storage if a mount command is issued for the file system.

If you plan to migrate HFS data sets, consider migrating them only to level 1 (DASD) storage. Recalling a data set that was migrated to tape could adversely affect performance because of the time required to physically mount the tape volume.

If the tape is in an automated library, then recalling it should be much faster than if the system has to request an operator to mount it.

Backing Up File Systems

DFSMShsm provides automatic backup facilities for HFS data sets. You can back up mountable file systems by periodically backing up the HFS data sets that contain them; the data sets can be restored if necessary. DFSMShsm is also used for migrating and restoring unmounted file systems.

You can manually back up a mountable file system, including the root file system, periodically with DFSMSdss data set dumps. To do this, issue the DFSMSdss DUMP command. This quiesces activity against the specified HFS data set, then invokes backup processing. When the backup is complete, the file system is unquiesced and user activity can resume.

Retain periodic DFSMSdss data set dumps of file systems in case a program fails and damages files and directories. Keep this backup in another area or different building in case the computer site experiences physical damage.

There is no facility for automatically backing up individual files within an HFS data set. You can manually back up files with the PAX, CPIO, and TAR commands.

For more information on using DFSMSdss, refer to the DFSMSdss section of *z/OS DFSMSdfp Storage Administration*.

Recovering a Backed-Up File System

If a file system is damaged, you can recover by replacing it with a saved version that was created from an earlier backup. To recover a backed up file system, you must perform the following tasks:

1. Notify all users to stop all activity on the damaged file system.
2. Have an authorized user enter the TSO/E UNMOUNT command with the IMMEDIATE option to logically unmount the damaged file system. If the unmount fails, reenter the UNMOUNT command with the FORCE option.
3. Use the DFSMSdss dump utility to restore the backed up file system to a replacement file system (HFS data set). You can do this simultaneously with the previous steps.
4. Have an authorized user enter a TSO/E MOUNT command to logically mount the replacement file system. Ensure that the MOUNT commands in the BPXPRMxx member in SYS1.PARMLIB are consistent with this MOUNT command.
5. Issue a broadcast message to all users or a message to all z/OS UNIX users when they invoke the shell, telling them that you have mounted a back-level file system and informing them of the mount point. Users must recreate and add any files added since the file system was backed up.

You might choose a more disruptive method to recover a backed up file system for the root file system. Refer to *z/OS UNIX System Services Planning*, “Shutting Down z/OS Unix“, for more information on how to perform a shutdown.

HFS Deferred File System Synchronization

Normal HFS disk hardening executes under a sync daemon that runs periodically to write out all file and metadata changes that have occurred since the last time the sync daemon ran. If a large number of files are created, modified, or deleted within the time span that is between the file system sync intervals, then with the next file system sync, these file changes are collected and the disk version of the file system is sync'ed with a greatly reduced number of media manager calls. This reduces the actual I/O activity.

The reduced I/O can have a dramatic effect on performance, because the default sync daemon interval is one minute. The sync daemon runs in the OMVS address space and is independent of any user request to the file system.

When the sync daemon runs, all the HFS changes are batched into one large I/O request that gets passed to the media manager. The intent is to perform one long I/O operation to the HFS on disk. However, even with optimal conditions, where the HFS resides on one volume and is contained within a single extent, it is necessary to have a few media manager calls during the disk hardening I/O operation. If the HFS has multiple extents or resides on multiple volumes, multiple channel programs must be built, because a single I/O operation cannot span multiple extents.

In addition to the sync daemon interval, HFS also supports the individual file sync (fsync). The fsync can be performed at any time by the application program. However, be aware that an fsync will result in the entire file system being sync'ed. It is very important to note that during the HFS file system sync operation, the sync task runs independently of the users that are currently accessing the HFS. When the sync task runs, it obtains an exclusive latch for the file system being sync'ed. This latch is held for the duration of the file system sync operation. While this latch is held UNIX system services users of the HFS file system will not be able to access the HFS. When the sync task runs, the HFS must:

1. Update all of the new or changed files' metadata
2. Delete structures for any removed files
3. Update the internal HFS storage maps within the HFS attributes directory
4. Call the media manager to perform the I/O operation that makes the changes to the disk version of the HFS data set.

The I/O operation is synchronous, which means that HFS sync task will be suspended for the duration of the media manager call. Normally, this file system sync lockout condition is brief, lasting for a few seconds, and is rarely noticed by the end user. However, in some situations the file system sync lockout could last much longer. The duration of the lockout depends on the sync interval, the amount of work that must be performed by the SYNC task, and the performance of the I/O subsystem.

To minimize the effect on applications from the file system sync operations that take a long time to complete, you can do one or more of the following:

1. Mount the affected HFS with a shorter sync interval. This will reduce overall HFS performance, but the sync lockout will not be as long.
2. Split the HFS into two or more smaller HFS data sets.
3. Avoid high impact commands such as "rm -R" on large directories.

How to specify a SYNC value

The default HFS SYNC daemon interval value is 60 seconds. You may override this default by changing the SYNCDEFAULT option of the FILESYSTYPE parameter in the BPXPRM*xx* member of SYS1.PARMLIB. Individual HFS data sets can have an overriding SYNC value. This can be accomplished by specifying a SYNC value on the ROOT parameter, or a SYNC value on the MOUNT parameter in the BPXPRM*xx* member of SYS1.PARMLIB.

Refer to *z/OS MVS Initialization and Tuning Reference* for details on how to specify FILESYSTYPE, ROOT, and MOUNT parameters. Additionally, individual HFS data sets can be mounted by the user with the MOUNT command that specifies a SYNC value.

Using pfsctl (BPX1PCT) Physical File System Control for HFS

The following information describes the use of pfsctl, BPX1PCT callable service, for HFS.

The pfsctl callable service, BPX1PCT, conveys a command and an argument to a physical file system. The meaning of the command and argument are specific to the physical file system and are defined by the physical file system.

Format:

```
CALL BPX1PCT,(File_System_type,
              Command,
              Argument_length,
              Argument,
              Return_value,
              Return_code,
              Reason_code)
```

File_System_type:

```
Type:              Character string
Length:            8 bytes
Character string:  HFS
```

Command:

```
Type:              Integer
Length:            Fullword
```

If the physical file system is an HFS data set, you can code one of these values:

```
X'40000001'           DisplayBufferLimits
X'00000002'           ChangeBufferLimits
X'40000003'           DisplayGlobalStats
X'40000004'           DisplayFSStats
X'00000005'           ExtendFS
```

These HFS-defined commands and the data areas for each are defined in GFUMPCTL, located in SYS1.MODGEN.

Argument_Length: A 4-byte integer. See below for the valid values.

Argument: Structure as described below and in Table 105 on page 389, Table 106 on page 390, Table 107 on page 391, or Table 108 on page 392.

Return Value: Fullword

Return_Code: A 4-byte integer that BPX1PCT sets to one of the values described in Table 104 on page 388.

Reason_Code: A 4-byte integer that BPX1PCT sets to one of the values described in Table 104 on page 388.

If the physical file system is an HFS data set, then the following apply:

- “DisplayBufferLimits Command” on page 385
- “Output from TRKCALC” on page 299
- “ChangeBufferLimits Command” on page 386
- “DisplayGlobalStats Command” on page 386
- “DisplayFSStats Command” on page 387
- “BPX1PCT Return and Reason Codes” on page 388

DisplayBufferLimits Command

Function:
The DisplayBufferLimits command returns the storage limits in the data area, specifically the VMAX and FMIN values for the HFS buffers currently in effect.

Command:
X'40000001'

Argument_Length:
The argument length is the length of the data area. The data area must be at least the length of PCTL_BFRLIMITS_TYPE. See Table 105 on page 389.

Argument:
PCTL_BFRLIMITS_TYPE

Usage Notes

1. The caller does not have to be authorized.
2. VMAX and FMIN values are both in megabytes (MB).
3. This command is equivalent to the confighfs shell command:

```
confighfs -l
```

ChangeBufferLimits Command

Function:
The ChangeBufferLimits command modifies the storage limits, specifically the VMAX and FMIN values, for HFS buffers as requested in the data area.

Command:
X'00000002'

Argument_length:
The argument length is the length of the data area. The data area must be at least the length of PCTL_BFRLIMITS_TYPE. See Table 105 on page 389.

Argument:
PCTL_BFRLIMITS_TYPE

Usage Notes

1. The following fields must be set within this structure:
   - The PCTL_BL_ACTION flag must be set.
   - Modify VMAX by setting the PCTL_BL_VMAX flag and specifying the new VMAX value in the field PCTL_BL_VMAX_VAL.
   - Modify FMIN by setting the PCTL_BL_FMIN flag and specifying the new FMIN value in the field PCTL_BL_FMIN_VAL.
2. The caller must be a superuser.
3. A request to modify both VMAX and FMIN can be made in one call.
4. The VMAX and FMIN values are in megabytes (MB).
5. This command is equivalent to the confighfs shell commands. The following confighfs command sets virtual storage maximum (VMAX) to n, where n is in MB.:

   ```
   confighfs -v n
   ```

   The following confighfs command sets virtual storage minimum (FMIN) to n, where n is in MB.

   ```
   confighfs -f n
   ```

DisplayGlobalStats Command

Function:
The DisplayGlobalStats command returns HFS global system level statistics in the data area.

Command:
X'40000003'

Argument_Length:
The argument length is the length of the data area. The data area must be at least the length of PCTL_GLOBALSTATS_TYPE. See Table 106 on page 390.

Argument:
PCTL_GLOBALSTATS_TYPE

Usage Notes

1. The caller does not have to be authorized.
2. This command is equivalent to the command:

```
configfs -q
```

DisplayFSStats Command

Function:
The DisplayFSStats command returns HFS file system level statistics in the data area. The values returned for file system statistics are associated with the current mount of the specified file system.

Command:
X'40000004'

Argument_Length:
The argument length is the length of the data area. The data area must be the length of PCTL_FSSTATS_TYPE. See Table 107 on page 391.

Argument:
PCTL_FSSTATS_TYPE

Usage Notes

1. The file system name must be supplied as input in the field PCTL_FS_NAME within the PCTL_FSSTATS_TYPE structure.
2. The caller does not have to be authorized.
3. This command is equivalent to the configfs shell command:

```
configfs pathname
```

where the pathname is an absolute or relative pathname that identifies the HFS.

ExtendFS Command

Function:
The ExtendFS command attempts to extend (allocate additional disk space for) the specified file system by the requested amount as specified in the data area.

Command:
X'00000005'

Argument_Length:
The argument length is the length of the data area. The length of the data area must be at least the length of PCTL_EXTENDFS_TYPE. See Table 108 on page 392.

Argument:
PCTL_EXTENDFS_TYPE

Usage Notes

- The following fields must be set within this structure:
  - Set PCTL_EXT_FSNAME to the name of the file system to be extended.
  - If your request is to allocate space on a new volume, set the PCTL_EXT_NEW_VOL flag on.
  - Set PCTL_EXT_UNIT to one of the following units to extend:
    1. ExtendFS_UNIT_MB to indicate that the amount to extend is in megabytes (MB).

2. ExtendFS_UNIT_TRK to indicate that the amount to extend is in tracks.
3. ExtendFS_UNIT_CYL to indicate that the amount to extend is in cylinders.

- Set PCTL_EXT_AMT to the amount to be extended.
- The caller must be a superuser.
- If the flag PCTL_EXT_NEW_VOL = off, an attempt to obtain additional space is only made on the last volume of the HFS. The HFS will not be extended to a new volume.
- If the flag PCTL_EXT_NEW_VOL = on, an attempt to obtain additional space is only made on a new volume. In this case, candidate volumes must exist for the HFS, and the candidate volumes must have existed at the time of the current mount of the HFS.
- If a secondary space amount exists for the HFS, it is ignored.
- If the entire amount requested cannot be obtained, the operation fails.

- This command is equivalent to the confighfs shell command:

```
   confighfs -x size pathname         To extend on the same volume
```

```
   confighfs -xn size pathname        To extend to a new volume
```

  where size is the amount to be extended suffixed by the extend unit of M (megabytes), T (tracks), or C (cylinders) and pathname is an absolute or relative pathname that identifies the HFS.

BPX1PCT Return and Reason Codes

Using BPX1PCT for HFS returns a non-zero return code and reason code only if the return value is -1. Table 104 on page 388 contains a list of return and reason codes returned by BPX1PCT for the HFS DisplayBufferLimits, ChangeBufferLimits, DisplayGlobalStats, DisplayFSStats, and ExtendFS commands.

Table 104. BPX1PCT - Return Codes and Reason Codes

Return Code	Reason Code	Description	Command
0	0	Successful completion	
EINVAL	5B360106	The HFS pfsctl service ExtendFS function was called with an invalid PCTL_EXT_UNIT parameter.	ExtendFS only
	5B360108	The HFS pfsctl ExtendFS function was called with an extend value of zero.	ExtendFS only
EFAULT	5B360102	The HFS pfsctl service was called with an unsupported command code.	
	5B360103	The data area address provided for the HFS pfsctl command is zero.	
	5B360107	The HFS pfsctl service ChangeBufferLimits or ExtendFS function was called by an unauthorized caller.	
EMVSERR	5B360101	An error occurred during the add local recovery routine function.	
ENOBUFS	5B360104	The data area length provided for the HFS pfsctl command was less than the minimum size required to complete the request.	

Table 104. BPX1PCT - Return Codes and Reason Codes (continued)

Return Code	Reason Code	Description	Command
ENOENT	5B360105	An HFS pfsctl service was called with an unmounted HFS.unted	DisplayFSStats and ExtendFS only.
ENOSPC	5B27C005	DADSM error occurred; no space available to extend the HFS data set	ExtendFS only
	>80000000	File system was in an HFS Out of Space error state and the extend amount was insufficient to meet the requirement to complete the sync shadow write. The reason code contains the 2's complement of the additional number of tracks required.	ExtendFS only
EIO	5B27C00A	DADSM error occurred; I/O error VTOC	ExtendFS only
EMVSPFSPERM	5B27xxxx	Unexpected error encountered	ExtendFS only
	5BB3xxxx	Unexpected error encountered	ChangeBufferLimits only
EMVSPARM	5B36000E	VMAX below minimum (VMAX set to minimum)	ChangeBufferLimits only
	5B36000F	FMIN exceeded VMAX (FMIN set to VMAX)	ChangeBufferLimits only
	5B360010	FMIN at maximum available (FMIN set to maximum)	ChangeBufferLimits only
	5B360014	VMAX below FMIN (VMAX set to FMIN)	ChangeBufferLimits only

Table 105 on page 389 shows the GFUMPCTL structure for the DisplayBufferLimits and the ChangeBufferLimits commands:

Table 105. Structure for the DisplayBufferLimits and ChangeBufferLimits Commands (GFUMPCTL)

Offsets	Type	Length	Name	Description
0 (0)	STRUCTURE	28	PCTL_BFRLIMITS_TYPE	
0 (0)	CHARACTER	1	PCTL_BL_FLGS	Flags
	1...	1	PCTL_BL_ACTION	`off`: display buffer limits
				on: change buffer limits
1 (1)	CHARACTER	1	PCTL_BL_LIMIT	Limits to be modified
	1...		PCTL_BL_VMAX	Modify VMAX request
	.1..		PCTL_BL_FMIN	Modify FMIN request
2 (2)	CHARACTER	1	*	Reserved
4 (4)	SIGNED	4	PCTL_BL_VMAX_VAL	Modify VMAX value
8 (8)	SIGNED	4	PCTL_BL_FMIN_VAL	Modify FMIN value
12 (C)	CHARACTER	16	*	Reserved
28 (1C)	CHARACTER		PCTL_BL_END	

Table 106 on page 390 shows the GFUMPCTL structure for the DisplayGlobalStats command:

Table 106. Structure for the DisplayGlobalStats Command (GFUMPCTL)

Offsets	Type	Length	Name	Description
0 (0)	(0) STRUCTURE	200	PCTL_GLOBALSTATS_TYPE	
0 (0)	CHARACTER	8	PCTL_GS_CURR_TIME	Current timestamp
8 (8)	CHARACTER	8	*	Reserved
16 (10)	SIGNED	4	PCTL_GS_TOTVIRT	Combined total virtual (in pages) in use by all buffer pools
20 (14)	SIGNED	4	PCTL_GS_TOTFIX	Combined total fixed (in pages) in use by all buffer pools
24 (18)	CHARACTER	28	PCTL_GS_BP1	Stats for buffer pool 1
24 (18)	UNSIGNED	2	PCTL_GS_BP1_BFRSIZE	Size of each buffer (in pages)
26 (1A)	UNSIGNED	2	PCTL_GS_BP1_DSCNT	Number of data spaces in pool
28 (1C)	SIGNED	4	PCTL_GS_BP1_TOTVIRT	Total virtual used by this pool
32 (20)	SIGNED	4	PCTL_GS_BP1_TOTFIX	Total fixed used by this pool
36 (24)	UNSIGNED	8	PCTL_GS_BP1_FIXD_Y	Times a buffer was already fixed prior to an I/O request
44 (2C)	UNSIGNED	8	PCTL_GS_BP1_FIXD_N	Times a buffer was not already fixed prior to an I/O request
52 (34)	CHARACTER	28	PCTL_GS_BP2	Stats for buffer pool 2
52 (34)	UNSIGNED	2	PCTL_GS_BP2_BPRSIZE	Size of each buffer (in pages)
54 (36)	UNSIGNED	2	PCTL_GS_BP2_DSCNT	Number of data spaces in pool
56 (38)	SIGNED	4	PCTL_GS_BP2_TOTVIRT	Total virtual used by this pool
60 (3C)	SIGNED	4	PCTL_GS_BP2_TOTFIX	Total fixed used by this pool
64 (40)	UNSIGNED	8	PCTL_GS_BP2_FIXD_Y	Times a buffer was already fixed prior to an I/O request
72 (48)	UNSIGNED	8	PCTL_GS_BP2_FIXD_N	Times a buffer was not already fixed prior to an I/O request
80 (50)	CHARACTER	28	PCTL_GS_BP3	Stats for buffer pool 3
80 (50)	UNSIGNED	2	PCTL_GS_BP3_BFRSIZE	Size of each buffer (in pages)
82 (52)	UNSIGNED	2	PCTL_GS_BP3_DSCNT	Number of data spaces in pool
84 (54)	SIGNED	4	PCTL_GS_BP3_TOTVIRT	Total virtual used by this pool
88 (58)	SIGNED	4	PCTL_GS_BP3_TOTFIX	Total fixed used by this pool
92 (5C)	UNSIGNED	8	PCTL_GS_BP3_FIXD_Y	Times a buffer was already fixed prior to an I/O request
100 (64)	UNSIGNED	8	PCTL_GS_BP3_FIXD_N	Times a buffer was not already fixed prior to an I/O request
108 (6C)	CHARACTER	28	PCTL_GS_BP4	Stats for buffer pool 4
108 (6C)	UNSIGNED	2	PCTL_GS_BP4_BFRSIZE	Size of each buffer (in pages)
110 (6E)	UNSIGNED	2	PCTL_GS_BP4_DSCNT	Number of data spaces in pool
112 (70)	SIGNED	4	PCTL_GS_BP4_TOTVIRT	Total virtual used by this pool

Table 106. Structure for the DisplayGlobalStats Command (GFUMPCTL) (continued)

Offsets	Type	Length	Name	Description
116 (74)	SIGNED	4	PCTL_GS_BP4_TOTFIX	Total fixed used by this pool
120 (78)	CHARACTER	8	PCTL-GS_BP4_FIXD_Y	Times a buffer was already fixed prior to an I/O request
128 (80)	CHARACTER	8	PCTL_GS_BP4_FIXD_N	Times a buffer was not already fixed prior to an I/O request
136 (88)	CHARACTER	8	PCTL_GS_META_Y	Times metadata in cache on a lookup
144 (90)	CHARACTER	8	PCTL_GS_META_N	Times metadata not in cache on a lookup
152 (98)	CHARACTER	8	PCTL_GS_RPN0_Y	Times RPN0 in cache on read/write
160 (A0)	CHARACTER	8	PCTL_GS_RPN0_N	Times RPN0 not in cache on read/write
168 (A8)	CHARACTER	32	*	Reserved
200 (C8)	CHARACTER		PCTL_GS_END	

Table 107 on page 391 contains the GFUMPCTL structure DisplayFSStats command:

Table 107. Structure for the DisplayFSStats Command (GFUMPCTL)

Offsets	Type	Length	Name	Description
0 (0)	STRUCTURE	228	PCTL_FSSTATS_TYPE	
0 (0)	CHARACTER	44	PCTL_FS_NAME	File system name
44 (2C)	BITSTRING	4	PCTL_FS_FLAGS	Flags from RFS (an internal HFS control block)
44 (2C)	BITSTRING	1	PCTL_FS_RFS_MTAB_FLAGS	MTAB flags
45 (2D)	BITSTRING	1	PCTL_FS_RFS_FLAGS	RFS flags
46 (2E)	BITSTRING	1	PCTL_FS_RFS_SYNC_ERR	RFS error flags
47 (2F)	BITSTRING	1	*	Reserved
48 (30)	CHARACTER	8	PCTL_FS_CURR_TIME	Current timestamp
56 (38)	CHARACTER	8	PCTL_FS_MOUNT_TIME	Mount timestamp
64 (40)	UNSIGNED	2	PCTL_FS_SYNC	SYNC interval (in seconds)
66 (42)	CHARACTER	2	*	Reserved
68 (44)	SIGNED	4	PCTL_FS_SIZE	File system size (in pages)
72 (48)	SIGNED	4	PCTL_FS_USED	Number of pages used in FS
76 (4C)	SIGNED	4	PCTL_FS_AD_ALLOC	Number of pages allocated to AD in FS
80 (50)	UNSIGNED	8	PCTL_FS_SEQ_IO	Number of sequential I/O requests for data
88 (58)	UNSIGNED	8	PCTL_FS_RANDOM_IO	Number of random I/O requests for data
96 (60)	UNSIGNED	8	PCTL_FS_META_Y	Times metadata in cache on a lookup
104 (68)	UNSIGNED	8	PCTL_FS_META_N	Times metadata not in cache on a lookup

Table 107. Structure for the DisplayFSStats Command (GFUMPCTL) (continued)

Offsets	Type	Length	Name	Description
112 (70)	UNSIGNED	8	PCTL_FS_RPN0_Y	Times RPN0 in cache on read/write
120 (78)	UNSIGNED	8	PCTL_FS_RPN0_N	Times RPN0 NOT in cache on read/write
128 (80)	UNSIGNED	8	PCTL_FS_IX_TOPS	Number of index new tops
136 (88)	UNSIGNED	8	PCTL_FS_IX_SPLITS	Number of index splits
144 (90)	UNSIGNED	8	PCTL_FS_IX_JOINS	Number of index joins
152 (98)	UNSIGNED	8	PCTL_FS_IX_RDHIT	Number of index page read hits
160 (A0)	UNSIGNED	8	PCTL_FS_IX_RDMISS	Number of index page read misses
168 (A8)	UNSIGNED	8	PCTL_FS_IX_WRHIT	Number of index page write hits
176 (B0)	UNSIGNED	8	PCTL_FS_IX_WRMISS	Number of index page write misses
184 (B8)	UNSIGNED	4	PCTL_FS_PGS_CACHED	Number of data buffer pages cached by this file system
188 (BC)	CHARACTER	4	PCTL_FS_HFRFN	HFRFN from DMIB
192 (C0)	UNSIGNED	4	PCTL_FS_MEM_CNT	Member count from RFS
196 (C4)	UNSIGNED	4	PCTL_FS_APM_CNT	Number of available page maps
200 (C8)	CHARACTER	28	*	Reserved
228 (E4)	CHARACTER		PCTL_FS_END	

Table 108 on page 392 contains the GFUMPCTL structure ExtendFS command:

Table 108. Structure for the ExtendFS Command (GFUMPCTL)

Offsets	Type	Length	Name	Description
0 (0)	STRUCTURE	68	PCTL_EXTENDFS_TYPE	
0 (0)	CHARACTER	44	PCTL_EXT_FSNAME	File system name
44 (2C)	CHARACTER	1	PCTL_EXT_FLGS	Flags
	1...		PCTL_EXT_NEW_VOL	Extend to new volume
45 (2D)	CHARACTER	1	PCTL_EXT_UNIT	Unit of amount to extend (‘M’, ‘T’ or ‘C’)
46 (2E)	CHARACTER	2	*	Reserved
48 (30)	SIGNED	4	PCTL_EXT_AMT	Amount to be extended
52 (34)	CHARACTER	16	*	Reserved
68 (44)	CHARACTER		PCTL_EXT_END	

Table 109. Constants for Extend Units Supported

Type	Value	Name	Description
CHARACTER	1	EXTENDFS_UNIT_MB	Constant (‘M’, Megabytes (MB) for pctl_ext_unit
CHARACTER	1	EXTENDFS_UNIT_TRK	Constant (‘T’, Tracks) for pctl_ext_unit
CHARACTER	1	EXTENDFS_UNIT_CYL	Constant (‘C’, Cylinders) for pctl_ext_unit

Chapter 12. User Access to Subsystem Statistics, Status, and Counts Information

This information documents programming interfaces provided by IDCAMS.

Your program can call a system service that returns the information that the access method services LISTDATA command returns. This information is subsystem statistics, status and count information. To obtain this information, your program must be APF-authorized. Your program will call IDCSS01 with the LINK macro. You pass to IDCSS01 a three-word parameter list pointed to by register 1. If your program is not authorized and attempts to link to IDCSS01, the system issues ABEND with system code 047. The following shows an example of the IDCSS01 call:

```
         ST    R5,AddrSGARGL        Set address of SSGARGL
         LINK  EP=IDCSS01,PARAM=(,MyArgL,MyRetCode)   Call LISTDATA
         SR    R15,R15              Prepare for ICM
         ICM   R15,B'0011',MyRetCode  Obtain and test return code
         BNZ   ListDataFail         Branch if call failed
         . . .
MyArgL   DC    A(AddrSSGARGL)       Address of SSGARGL address
AddrSGARGL DC  A(0)                 Address of SSGARGL
MyRetcode DC   H'0'                 Return code from service
         IDCDF70 ,                  Map the parameter list
```

Your program must point register 13 to a standard 18-word save area when calling IDCSS01.

Register 1 Parameter List

Word 1
must be zero.

Word 2
contains an address of a pointer to the argument list SSGARGL (detailed below), which IDCSS01 requires. Within this argument list is a field named SSGOADR, which points to the buffer area in which IDCSS01 returns subsystem statistics, status, or counts information. The buffer area may be obtained by the caller or may be left for IDCSS01 to obtain.

Word 3
points to an area to receive a 2-byte binary return code. Possible return codes are listed at the end of SSGARGL. IDCSS01 also returns this return code in the low order bytes of register 15.

Passed Argument List -- SSGARGL

The following describes SSGARGL. SSGARGL is the area pointed to by the word that Word 2 points to. Word 2 is part of the parameter list passed to IDCSS01. The caller must establish some fields; IDCSS01 establishes other fields. The caller must set an option flag to indicate whether information requested is status information (SSGRSS), counts information (SSGRPD), Space Efficient Volume status (SSGSEV), or Extent Pool Configuration status (SSGEPC). If the caller passes SSGADDN, IDCSS01 establishes SSGAVOL and SSGUNIT. .

If counts information is requested, the caller must indicate whether it applies to all subsystems (SSGALL), a specific subsystem (SSG1SS), or a specific device (SSGDEV). In addition, the caller must either pass the ddname (through SSGADDN) of a DD statement that allocates a caching subsystem volume, or the caller must identify the volume and unit (through SSGAVOL and SSGUNIT) of a caching subsystem volume for which information is being requested.

If Space Efficient Volume status, or Extent Pool Configuration status is requested, the caller must follow the instruction described in the mapping for the space efficient volume status output buffer, or mapping for the extent pool configuration status output buffer described in later of this section.

Note: The SSGOLN may be larger than the data returned in the SSGBUFR buffer. To detect end of data, the following should be considered:

- If "current index into the SSGBUFR buffer" is within SSGOLN-4 bytes, then the full six bytes of SSGDAVOL being zero is the way to determine the premature "end of data".
- If "current index into the SSGBUFR buffer" is beyond SSGOLN-4 bytes of the data, then they are at "end of data".

The following list contains other fields that IDCSS01 establishes:

```
================================================================================
  OFFSET   OFFSET
 DECIMAL     HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
======== ======== ========= ========  ============== ================================
       0      (0) STRUCTURE       96  SSGARGL
       0      (0) CHARACTER        8   SSGHEAD       SSGARGL IDENTIFIER
       8      (8) ADDRESS          4   SSGADDN       DDNAME ADDRESS
      12      (C) ADDRESS          4   SSGAVOL       VOLUME ADDRESS
      16     (10) BITSTRING        4   SSGUNIT       DEVICE TYPE
      16     (10) BITSTRING        1    SSGUNIT1     GENERAL FLAGS
      17     (11) BITSTRING        1    SSGUNIT2     GENERAL FLAGS
      18     (12) BITSTRING        1    SSGUNIT3     DEVICE CLASS
      19     (13) BITSTRING        1    SSGUNIT4     DEVICE TYPE WITHIN THE CLASS
      20     (14) SIGNED           4   SSGOLN        LENGTH OF OUTPUT BUFFER
      24     (18) ADDRESS          4   SSGOADR       ADDRESS OF OUTPUT BUFFER
      28     (1C) BITSTRING        2   SSGOPT        OPTIONS BYTE
                  1... ....             SSGRPD       SENSE SUBSYSTEM COUNTS
                  .1.. ....             SSGRSS       SENSE SUBSYSTEM STATUS
                  ..1. ....             SSGCACHE     ON=CACHING,OFF=PAGING
                  ...1 ....             SSGSDS       ON=1 SD, OFF=2 SD'S
                  .... 1...             SSGALL       SENSE FOR ALL SUBSYSTEMS
                  .... .1..             SSG1SS       SENSE FOR SPECIFIED SUBSYSTEM
                  .... ..1.             SSGDEV       SENSE FOR SPECIFIED DEVICE
                  .... ...1             SSGAMD       PTR TO 3880 MODEL PASSED
                                                     INSTEAD OF PTR TO VOLUME
      29     (1D) 1... ....             SSG2SD       PRINT 2 SD
                  .1.. ....             SSGACD       ACCESSCODE
                  ..1. ....             SSGOFFL      SSGAVOL POINTS TO DEVICE ID OF
                                                     AN OFFLINE DEVICE
                  ...1 ....             SSGARGL2     This flag indicates that the
                                                     parameter list is extended by
                                                     the 16 bytes defined as
                                                     SSGARGLX.
                  .... 1...             SSGLINKP     Link performance statistics
                                                     supported by caller and
                                                     request return of statistics
                                                     when request for Subsystem
                                                     counts are made (SSGRPD=ON)
                                                     for one or more Subsystems
                                                     (SSG1SS=ON or SSGALL=ON).
                                                     Results passed back in area
                                                     pointed to by SSGLPOAR.
                                                     Extended size statistics are
                                                     provided if SSGEXRQ is
                                                     specified.  The returned field,
                                                     SSGLLSET, will be set to the
                                                     size for each set of link
                                                     statistics.
                  .... .1..             SSGARGL3     This flag indicates that the
                                                     parameter list is extended with
                                                     area defined as SSGARLGY.
      30    (1E) BITSTRING          1   SSGMDLID      CU MODEL IDENTIFIER
      31    (1F) CHARACTER          1   SSGRCIOS      RETURN CODE FROM
                                                     FAILING FUNCTION
                                                     - ESTAE OR IOS
     Parameter List Extension
     Note: IDCSS01 will not reference fields contained in SSGARGLX
           unless the bit defined as SSGARGL2 is set to '1'.
================================================================================
      32     (20) CHARACTER       16    SSGARGLX     parm list extension
      32     (20) UNSIGNED         1    SSGATIME     I/O timeout value in
                                                     seconds.When SSGATIME is
                                                     not zero, it will be stored
                                                     into IOSXTIME in the IOSB. If
                                                     the I/O is active or queued
                                                     longer that SSGATIME, the I/O
                                                     will be terminated and IDCSS01
                                                     will return a return code of
```

```
                                                  60. Hex values for SSGATIME
                                                  support seconds from 1 to 255.
      33     (21) CHARACTER        2    SSGADEVN     Binary device number that
                                                  received an IO (RC8) or
                                                  timeout (RC60) from IDCSS01
      35     (23) UNSIGNED         1    SSGLPRET     Reason code from link
                                                  performance statistics
                                                  processing
      36     (24) ADDRESS          4    SSGLPOAR     Address of link performance
                                                  Statistics output buffer
                                                  queue. When zero on return,
                                                  statistics not available for
                                                  devices found on counts
                                                  request or statistics not
                                                  requested by the caller.
      40     (28) CHARACTER        1     SSGSCHST    SUBCHANNEL SET ID
                                                  NOTE: Set SSGSCHST to
                                                  subchannel set id + 1 if using
                                                  branch entry interface to
                                                  issue STATUS command to an offline
                                                  device in an alternate
                                                  subchannel set.
      41     (29) CHARACTER        7     *           RESERVED

     Parameter List Extension number 3
     Note: IDCSS01 will not reference fields contained in SSGARGLY
           unless the bit defined as SSGARGL3 is set to '1'.
================================================================================
      48     (30) CHARACTER       48   SSGARGLY      Parm list extension 3
      48     (30) BITSTRING        4   SSGOPTY       Extended options
                  1... ....            SSGRANKP     RANK performance statistics
                                                  are supported by caller and
                                                  request return of statistics
                                                  when request for Subsystem
                                                  counts are made (SSGRPD=ON)
                                                  for one or more Subsystems
                                                  (SSG1SS=ON or SSGALL=ON).
                                                  Results passed back in area
                                                  pointed to by SSGRSOAR.
                   .1.. ....           SSGSEGMP     Extent pool statistics are
                                                  supported by caller and
                                                  request return of statistics
                                                  when request for Subsystem
                                                  counts are made (SSGRPD=ON)
                                                  for one or more Subsystems
                                                  (SSG1SS=ON or SSGALL=ON).
                                                  Results passed back in area
                                                  pointed to by SSGSPOAR.
                  ..1. ....            SSGSEV       Space Efficient Volume
                                                  (SEVOL) status is
                                                  supported by caller and
                                                  request return of status
                                                  in SSGSEBUF pointed by
                                                  SSGSEOAR when request for
                                                  SEVOL are made (SSGSEV
                                                  =ON) for the scope of
                                                  single device
                                                  (SSGDEV=ON), online SEVOL
                                                  devices in the subsystem
                                                  (SSG1SS=ON), or all
                                                  online SEVOL devices
                                                  (SSGALL=ON). For single
                                                  offline SEVOL status,
                                                  caller set address of
                                                  the MVS addr character
                                                  in SSGAVOL and set
                                                  offline (SSGOFFL=ON).
                                                  The Fixed Block (FB)
                                                  SEVOL device status is
                                                  returned when the
                                                  reporting scope is single
                                                  (SSGDEV=ON), the FB user
                                                  request is active
                                                  (SSGURFBD=ON), FB LSS
                                                  number is in SSGFBLSS,
                                                  and FB device number is
                                                  in SSGFBDEV. FB SEVOL
                                                  status is available
                                                  through CKD I/O device
                                                  that resides in the same
                                                  Storage Facility Image as
                                                  FB SEVOL and resides in
```

```
                                    the same Even or Odd
                                    number LSS as even or odd
                                    FB LSS.  SEVOL status is
                                    not supported when other
                                    statistics or status is
                                    requested by caller.
...1 ....           SSGEPC          Extent Pool Config (EPC)
                                    status is supported by
                                    caller and request return
                                    of status in SSGEPBUF
                                    pointed by SSGEPOAR when
                                    request for EPC of
                                    Storage Facility Image
                                    oriented by I/O device in
                                    SSGADDN or SSGAVOL are
                                    made (SSGEPC=ON).  When
                                    Extent Pool ID (SSGEPID)
                                    is set as 'FFFF'X by
                                    caller, EPC summary
                                    status is returned.  When
                                    even numbered Extent Pool
                                    ID is specified in
                                    SSGEPID with the I/O
                                    device in the even LSS,
                                    detail status of EPC for
                                    the specified Extent Pool
                                    is returned.  When odd
                                    numbered Extent Pool ID
                                    is specified with the I/O
                                    device in the odd LSS,
                                    detail status of EPC for
                                    the specified Extent Pool
                                    is returned. SSGURBMP is
                                    internal use only. When
                                    Extent Pool ID (SSGEPID)
                                    is set as 'FFFF'X with
                                    SSGURBMP is ON, branch
                                    entry caller receives
                                    error. Whether ON or OFF
                                    of SSGURBMP does not make
                                    difference in EPC detail
                                    status for branch entry
                                    caller. EPC status report
                                    is not supported when
                                    other statistics or
                                    status is requested
                                    by caller.
.... 1...           SSGURFBD        Request Fixed Block SEVOL
                                    status by turning
                                    SSGURFBD flag ON.
.... .1..           SSGURBMP        IDCAMS internal use.
                                    Refer SSGEPC bit
                                    description.
.... ..1.           SSGEXRQ         By setting this attribute bit
                                    along with other request(s),
                                    caller supports and requests the
                                    return of extended length
                                    statistics.   The extended
                                    length is any set of statistics
                                    that is larger than the standard
                                    length of statistics. The
                                    extended length statistics will
                                    be returned if target storage
                                    facility supports it, otherwise
                                    the standard size statistics
                                    will be returned. Ignored this
                                    attribute bit for requests where
                                    the statistics do not support
                                    the extended length and where
                                    caller does not make any
                                    request.
                                    Current request that supports
                                    the extended length is:
                                    - Link performance statistics
.... ...1           SSGVOL          Volume status is
                                    supported by
                                    caller and
                                    request return of
                                    status in
                                    SSGSEBUF pointed
                                    by SSGSEOAR when
                                    request for
```

```
                                                  volume status are
                                                  made (SSGVOL=ON)
                                                  for the scope of
                                                  single device
                                                  (SSGDEV=ON),
                                                  online devices in
                                                  the subsystem
                                                  (SSG1SS=ON), or
                                                  all online
                                                  devices
                                                  (SSGALL=ON). For
                                                  single offline
                                                  volume status,
                                                  caller sets
                                                  address of the
                                                  MVS addr
                                                  character in
                                                  SSGAVOL and set
                                                  offline
                                                  (SSGOFFL=ON).
                                                  Volume status is
                                                  not supported
                                                  when other
                                                  statistics or
                                                  status is
                                                  requested by
                                                  caller.
      49   (31) 1... ....       SSGSIOLP       Synch I/O Link
                                                  statistics are
                                                  supported by
                                                  caller and
                                                  request return of
                                                  statistics when
                                                  request for
                                                  Subsystem counts
                                                  are made
                                                  (SSGRPD=ON) for
                                                  one or more
                                                  Subsystems
                                                  (SSG1SS=ON or
                                                  SSGALL=ON).
                                                  Results passed
                                                  back in area
                                                  pointed to by
                                                  SSGSLOAR.
           .1.. ....       SSGSILDP       Synch I/O Link
                                                  Diagnostic
                                                  Parameters are
                                                  supported by
                                                  caller and
                                                  request return
                                                  of statistics
                                                  when request for
                                                  Subsystem counts
                                                  are made
                                                  (SSGRPD=ON)
                                                  for one or more
                                                  Subsystems
                                                  (SSG1SS=ON or
                                                  SSGALL=ON).
                                                  Results passed
                                                  back in area
                                                  pointed to by
                                                  SSGSDOAR.
           ..11 ....       SSGEPCV        Extent Pool Detailed
                                                  Info Version
                                                  Version 1 – 00
                                                  Version 2 - 10
                                                  Version 3 - 01
52    (34) CHARACTER        8    SSGRETY        Extended return codes
52    (34) UNSIGNED         1    SSGRSRET       Reason code from Rank
                                                  Performance Statistics
                                                  processing
53    (35) UNSIGNED         1    SSGSGRET       Reason code from Segm
                                                  Performance Statistics
                                                  processing
54    (36) UNSIGNED         1    SSGSLRET       Reason code from
                                                  Synch I/O Link
                                                  Statistics
                                                  processing
55    (37) UNSIGNED         1    SSGSDRET       Reason code from
                                                  Synch I/O Link
```

```
                                                  Diagnostic
                                                  Parameters
                                                  processing
     60     (3C) ADDRESS          4    SSGRSOAR     Address of Rank performance
                                                  Statistics output buffer
                                                  queue. When zero on return,
                                                  statistics not available for
                                                  devices found on counts
                                                  request or statistics not
                                                  requested by the caller.
     64     (40) ADDRESS          4    SSGSPOAR     Address of extent pool
                                                  performance Statistics output
                                                  buffer queue. When zero on
                                                  return, statistics not
                                                  available for devices found on
                                                  counts request or statistics
                                                  not requested by the
                                                  caller.
     68     (44) ADDRESS          4    SSGSEOAR     Address of SE Volume
                                                  status output buffer
                                                  queue. When zero on
                                                  return, status not
                                                  available for devices
                                                  found on SEVOL status
                                                  request or status not
                                                  requested.
     72     (48) ADDRESS          4    SSGEPOAR     Address of Extent Pool
                                                  Config (EPC) status
                                                  output buffer. When zero
                                                  on return, status not
                                                  available for Storage
                                                  Facility Image found on
                                                  EPC status request or
                                                  status not requested
                                                  by the caller.
     76     (4C) UNSIGNED         2    SSGEPID      Extent Pool ID.
                                                 'FFFF'X indicates no
                                                  EP ID is provided
                                                  by caller.
     78     (4E) UNSIGNED         2    SSGFBLSS     Fix Block LSS number
     80     (50) UNSIGNED         2    SSGFBDEV     Fix Block device number
     82     (52) CHARACTER        2    *            reserved for
                                                  future use
     84     (54) ADDRESS          4    SSGSLOAR     Address of Synch
                                                  I/O Link
                                                  statistics output
                                                  buffer queue.
                                                  When zero on
                                                  return, statistics
                                                  not available for
                                                  devices found on
                                                  counts request or
                                                  statistics not
                                                  requested by the
                                                  caller.
      88   (58) ADDRESS          4    SSGSDOAR     Address of Synch
                                                  I/O Link
                                                  Diagnostics
                                                  Parameters output
                                                  buffer queue.
                                                  When zero on
                                                  return, statistics
                                                  not available for
                                                  devices found on
                                                  counts request or
                                                  statistics not
                                                  requested by the
                                                  caller.
      92   (62) CHARACTER         4   *            reserved for
                                                  future use

        MAPPING FOR THE SUBSYSTEM GET OUTPUT BUFFER
================================================================================
  OFFSET   OFFSET
  DECIMAL     HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
 ======== ======== ========= ========  ============== ================================
        0      (0) STRUCTURE        *  SSGBUFR
        0      (0) CHARACTER        6   SSGDAVOL      VOLUME SERIAL
        6      (6) BITSTRING        1   SSGDAFLG
                   1... ....            SSG_UA_FLAG   'ONE' UNIT ADDR FORMAT
                   .1.. ....            SSG_FND_STG2  2ND STG DIR PATH FND SS01 TO
                                                      LA01 IDCSS01 PASSED TO IDCLA01
```

```
                    ..11 ....             SSG_SCHST    SUBCHANNEL SET NUMBER
                                                        SCHSET NUMBER 0 - 00
                                                        SCHSET NUMBER 1 - 01
                                                        SCHSET NUMBER 2 - 10
                                                        SCHSET NUMBER 3 - 11
                    ..11 1111             *            RESERVED
          7      (7) CHARACTER        1   SSGDCUID     Real CUID for data in
                                                        SSGDADA
          8      (8) CHARACTER        4   SSGDAUA1     FIRST UNIT ADDRESS
         12      (C) CHARACTER        2   SSGDAUB1     FIRST UNIT ADDRESS (BINARY)
         14      (E) SIGNED           2   SSGDALN      DATA LENGTH
         16     (10) CHARACTER      240   SSGDADA      DATA AREA
         16     (10) CHARACTER      240   SSGDAXPF     CACHING SUBSYSTEM PERFORMANCE
                                                        STATISTICS
         16     (10) CHARACTER      160   SSGDASPF     CACHING SUBSYSTEM COUNTS
         16     (10) CHARACTER       80   SSGDA2SD     STATUS FOR 2 SD
         16     (10) CHARACTER       80   SSGDAIPF     PAGING SUBSYSTEM COUNTS
         16     (10) CHARACTER       44   SSGDASS      SUBSYSTEM STATUS (LNGTHND)
         16     (10) CHARACTER       40   SSGDASS_40   STATUS FOR 3380
         16     (10) CHARACTER       24   SSGDAACD     ACCESSCODES
         16     (10) CHARACTER       12   SSGDAAC0     STORAGE CLUSTER 0 WORD
         28     (1C) CHARACTER       12   SSGDAAC1     STORAGE CLUSTER 1 WORD

        MAPPING FOR THE SUBSYSTEM GET OUTPUT BUFFER
        FOR LINK PERFORMANCE STATISTICS
        Output buffer is a chain, anchored by SSGLPOAR, of link
        performance statistics tables. Each table entry represents
        data read from a single storage subsystem box. The
        statistics from a single box may be described with more than
        table entry. The chain is a single threaded queue. Each
        chain element needs to be freed by the caller of this
        interface. Output buffer is built for storage boxes
        that support link performance statistics and are returned
        only if requested by caller (SSGLINKP) on a request for
        performance data (SSGRPD) for requests with scope of
        subsystem (SSG1SS) or all subsystems (SSGALL).
        The parameter list extension must also be passed as
        indicated with SSGARLG2.
================================================================================
  OFFSET   OFFSET
 DECIMAL     HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
======== ======== ========= ========  ============== ================================
       0      (0) STRUCTURE        *  SSGLBUFR
       0      (0) CHARACTER       56   SSGLHDR       Queue element header
       0      (0) UNSIGNED         4    SSGLQDAL     Length of this queue element,
                                                      header plus data
       4      (4) UNSIGNED         4    SSGLQDAO     Offset to the start of the
                                                      statistics sets
       8      (8) ADDRESS          4    SSGLFWDP     Pointer to next queue element,
                                                      when zero end of queue

          Orientation to subsystem that stats pertain
================================================================================
      12      (C) CHARACTER        6    SSGLVOL      Volume serial of device that
                                                      stats where read from
      18     (12) CHARACTER        2    SSGLDEVN     Device number of device that
                                                      Stats where read from
      20     (14) BITSTRING        1    SSGLFLG
                  11.. ....              *           RESERVED
                  ..11 ....              SSGL_SCHST  SUBCHANNEL SET NUMBER
                                                      SCHSET NUMBER 0 - 00
                                                      SCHSET NUMBER 1 - 01
                                                      SCHSET NUMBER 2 - 10
                                                      SCHSET NUMBER 3 - 11
                  .... 1111              *           RESERVED
      21     (15) CHARACTER        7    *            RESERVED
      28     (1C) CHARACTER       28    SSGLLHDR     Link stats info
      28     (1C) UNSIGNED         2    SSGLNSET     Num of link stats sets read
      30     (1E) UNSIGNED         2    SSGLLSET     Size of each set
                                                      The extended size would be set
                                                      if the target storage facility
                                                      supports it and SSGEXRQ is set,
                                                      otherwise the standard 96-bytes
                                                      entry size for each interface
                                                      ID will be set.
      32     (20) CHARACTER        6    SSGLCUT      Control unit type
      38     (26) CHARACTER        3    SSGLCUM      Control unit model
      41     (29) CHARACTER       10    SSGLSEQ      Control unit sequence number
      51     (33) CHARACTER        1    SSGLVER      Version of link statistics.
                                                      When value of version is X'00',
                                                      it indicates that the link
                                                      statistics are defined in the
```

```
                                                    96 byte format. When value of
                                                    version is X'01', it indicates
                                                    that the link statistics are
                                                    defined in the 156 byte format.
                                                    When value of version is X'02',
                                                    it indicates the link statistics
                                                    are defined in the 220 byte
                                                    format.
      52     (34) CHARACTER        4     *          Not used
      56     (38) CHARACTER        *   SSGLDADA     Link performance stats
                                                    read

         SSGLDADA consists of link statistics sets with Interface ID
         containing one or more table information entries. Each
         entry can be 96 bytes defined in SSGLLSTA or 156 bytes
         defined in SSGLXSTA or 220 bytes defined in SSGLXST2
         depending on version number in SSGLVER.
================================================================================
  OFFSET   OFFSET
 DECIMAL     HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
======== ======== ========= ========  ============== ==============================
      56     (38) CHARACTER       96    SSGLLSTA(*)  Stats entry
      56     (38) UNSIGNED         1     SSGLLTYP    Link type
                                                     00 - Link Not
                                                          Operational
                                                     01 - ESCON
                                                     02 - Fibre Channel 1 Gb/s
                                                     03 - Fibre Channel 2 Gb/s
                                                     04 - Fibre Channel 4 Gb/s
                                                     05 - Fibre Channel 8 Gb/s
                                                     06 - Fibre Channel 16 Gb/s
                                                     07 - Fibre Channel 32 Gb/s
                                                     08-0F - Reserved
                                                     10 - Fibre Channel Over
                                                          Ethernet 10 Gb/s
                                                     11 - Fibre Channel Over
                                                          Ethernet 40 Gb/s
      57     (39) UNSIGNED         1     *          Unused
      58     (3A) UNSIGNED         1     SSGLLBYI   Byte increment
                                                      Value Description
                                                      01  Low order bit represents
                                                          128K bytes.
      59     (3B) UNSIGNED         1     SSGLLTMI   Time increment
                                                      Value Description
                                                      01  Low order bit represents
                                                          16 milliseconds.
      60     (3C) BITSTRING        1     SSGLLFLG   Flags
                  1... ....              SSGLCPBS   CPU Percent Busy Scope
                  .1.. ....              SSGLPSSV   Port Security Statistics Valid
                                                     When set to one, SSGLPSCG,
                                                     SSGLRMLG, SSGLRLSC,
                                                     SSGLRLLA, and SSGLRLEF
                                                     are valid.
                  ..11 1111        *                Not Used
      61     (3D) CHARACTER        1     *          Unused
      62     (3E) CHARACTER        2     SSGLLIID   Interface id
      64     (40) CHARACTER       88     SSGLLSAT   Link statistics
      64     (40) UNSIGNED         4      SSGLERDB  Eckd read bytes in byte
                                                     increment
      68     (44) UNSIGNED         4      SSGLEWRB  Eckd write bytes in byte
                                                     increment
      72     (48) UNSIGNED         4      SSGLERDO  Eckd read operations. for
                                                     escon ports, one count per
                                                     chain which transfers data to
                                                     the host. for ficon ports, one
                                                     count per command which
                                                     transfer data to the host
      76     (4C) UNSIGNED         4      SSGLEWRO  Eckd write operations. for
                                                     escon ports, one count per
                                                     chain which transfers data to
                                                     the host. for ficon ports, one
                                                     count per command which
                                                     transfer data to the host
      80     (50) UNSIGNED         4      SSGLERDT  Eckd read accumulated time on
                                                     channel. the active processing
                                                     time for each command is
                                                     accumulated based on increment
                                                     value.
      84     (54) UNSIGNED         4      SSGLEWRT  Eckd write accumulated time on
                                                     channel. the active processing
                                                     time for each command is
                                                     accumulated based on increment
```

```
                                                  value.
   88     (58) UNSIGNED         4      SSGLPRDB   Pprc send bytes in byte
                                                  increment
   92     (5C) UNSIGNED         4      SSGLPWRB   Pprc received bytes in byte
                                                  increment
   96     (60) UNSIGNED         4      SSGLPRDO   Pprc send operations. each
                                                  pprc write command sent by the
                                                  pprc primary
  100     (64) UNSIGNED         4      SSGLPWRO   Pprc received operations. each
                                                  pprc write command received by
                                                  the pprc secondary.
  104     (68) UNSIGNED         4      SSGLPRDT   Pprc send accumulated time
                                                  based on incremant value
  108     (6C) UNSIGNED         4      SSGLPWRT   Pprc received accumulated time
                                                  based on increment value
  112     (70) UNSIGNED         4      SSGLSRDB   Scsi read bytes in byte
                                                  increment
  116     (74) UNSIGNED         4      SSGLSWRB   Scsi write bytes in byte
                                                  increment
  120     (78) UNSIGNED         4      SSGLSRDO   Scsi read operations. each
                                                  scsi read is counted
  124     (7C) UNSIGNED         4      SSGLSWRO   Scsi write operations. each
                                                  scsi write is counted
  128     (80) UNSIGNED         4      SSGLSRDT   Scsi read accumulated time
                                                  based on increment value
  132     (84) UNSIGNED         4      SSGLSWRT   Scsi write accumulated time
                                                  based on increment value
  136     (88) BITSTRING        1      SSGLPSCG   Port Security Configuration
               1... ....                 SSGLPCSE  Port is Configured
                                                   for Security Enabled
               .1.. ....                 SSGLPCSF  Port is Configured
                                                   for Security Enforced
               ..11 1111                 *         Not Used. Set to zero.
  137     (89) UNSIGNED         1      *          Not Used
  138     (8A) UNSIGNED         2      SSGLRMLG   Remote Logins
  140     (8C) UNSIGNED         2      SSGLRLSC   Remote Logins -
                                                  Security Capable
  142     (8E) UNSIGNED         2      SSGLRLLA   Remote Logins -
                                                  Link Authenticated
  144     (90) UNSIGNED         2      SSGLRLEF   Remote Logins -
                                                  Encryption of Data
                                                  In Flight (EDIF)
  146     (92) CHARACTER        5      *          Not Used
  151     (97) UNSIGNED         1      SSGLPBHA   Percent Busy for Host Adapter
                                                  or Host Port.
   SSGLDADA consists of table entris defined by SSGLXSTA shown
   below when SSGLVER is decimal value of 1.
================================================================================
  OFFSET   OFFSET
 DECIMAL     HEX  TYPE         LENGTH NAME (DIM)     DESCRIPTION
======= ======== ========= ======== ============== ================================
      56     (38) CHARACTER      156   SSGLXSTA(*)  Stats entry
      56     (38) UNSIGNED        96    *           Mapped in SSGLLSTA
     152     (98) UNSIGNED         4    SSGLFLKF    Fibre Channel Link Failure
                                                    Error Count.
                                                    This count is the number of
                                                    miscellaneous fibre channel link
                                                    errors, such as unexpected NOS
                                                    received or a link state machine
                                                    failure detected.
     156     (9C) UNSIGNED         4    SSGLFLSY    Fibre Channel Loss of
                                                    Synchronization Error Count.
                                                    This count is the number of loss
                                                    of synchronization errors where
                                                    it is a confirmed and
                                                    a persistent synchronization
                                                    loss on the fibre channel link.
     160     (A0) UNSIGNED         4    SSGLFLSG    Fibre Channel Loss of Signal
                                                    Error Count.
                                                    This count is the number of times
                                                    that a loss of signal was
                                                    detected on the fibre channel
                                                    link when a signal was
                                                    previously detected.
     164     (A4) UNSIGNED         4    SSGLFPSQ    Fibre Channel Primitive Sequence
                                                    Error Count.
                                                    This count is the number of
                                                    primitive sequence protocol error
                                                    counts where an unexpected
                                                    primitive sequence was received.
     168     (A8) UNSIGNED         4    SSGLFITW    Fibre Channel Invalid
                                                    Transmission Word Error Count.
```

					This count is the number of times a "bit" error was detected. Examples of a "bit" errors are a code violation, invalid special code alignment, or disparity errors.
172	(AC)	UNSIGNED	4	SSGLFCRC	Fibre Channel CRC Error Count. This count is the number of times a received frame's CRC is in error.
176	(B0)	UNSIGNED	4	SSGLFLR1	Fibre Channel Link Recovery (LR) Sent Count Count. This count is the number of times the port has transitioned from an active (AC) state to a Link Recovery (LR1) state.
180	(B4)	UNSIGNED	4	SSGLFLR2	Fibre Channel Link Recovery (LR) Received Count Recovery Count. This count is the number of times the port has transitioned from an active (AC) state to a Link Recovery (LR2) state.
184	(B8)	UNSIGNED	4	SSGLFILF	Fibre Channel Illegal Frame Count. This count is the number of frames that violated Fibre Channel protocol. One example is an invalid frame header. One common reason is when the first frame of data sequence is missing and a subsequent data frame is detected as illegal.
188	(BC)	UNSIGNED	4	SSGLFOOD	Fibre Channel Missing Frame Count. This count is the number of times that a missing frame is detected. The frame is either missing from a data sequence or it is received beyond the port's sequence reassembly threshold.
192	(C0)	UNSIGNED	4	SSGLFOOA	Fibre Channel Out of Order ACK Count. This count is the number of times that a out of order ACK frame is detected. The frame is either missing from a data sequence or it is received beyond the port's sequence reassembly threshold.
196	(C4)	UNSIGNED	4	SSGLFDPF	Fibre Channel Duplicate Frame Count. This count is the number of times a frame was received that has been detected as previously processed.
200	(C8)	UNSIGNED	4	SSGLFIRO	Fibre Channel Invalid Relative Offset Count. This count is the number of times that a frame was received with bad relative offset in the frame header.
204	(CC)	UNSIGNED	4	SSGLFSQT	Fibre Channel Sequence Timeout Count. This count is the number of times the port has detected a timeout on receiving the next frame in a fibre channel sequence.
208	(D0)	UNSIGNED	4	SSGLFBER	Fibre Channel Bit Error Rate Count. This count is the number of the bit error (invalid transmission word) bursts for the previous 5 minute counting window.

SSGLDADA consists of table entris defined by SSGLXST2 shown below when SSGLVER is decimal value of 2.

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
56	(38)	CHARACTER	220	SSGLXST2(*)	Stats entry
56	(38)	UNSIGNED	156	*	Mapped in SSGLXSTA
212	(D4)	UNSIGNED	4	SSGLFRBZ	Fibre Channel Receive Buffer

				Zero Credit. This count is the number of one second intervals where the receive buffer credit was zero.
216	(D8) UNSIGNED	4	SSGLFSBZ	Fibre Channel Send Buffer Zero Credit. This count is the number of one second intervals where the send buffer credit was zero.
220	(DC) UNSIGNED	4	SSGLFEHC	Fibre Channel Exchange High Count. This count is the number of times the Fibre Channel exchange count crossed the High Threshold.
224	(E0) UNSIGNED	4	SSGLFQFB	Fibre Channel Queue Full or Busy Status. This count is the number of times the FCP port returned QUEUE FULL or BUSY status.
228	(E4) UNSIGNED	4	SSGLFEOC	Fibre Channel Exchange Overrun Count. This count is the number of Fibre Channel exchanges that were lost due to overdriving the host adapter port.
232	(E8) UNSIGNED	4	SSGLFARC	Fibre Channel Abort Received Count. This count is the number of times a port received an abort.
236	(EC) UNSIGNED	4	SSGLRDRP	Read Diagnostic Parameter: Rx Power. This value contains the measured received optical power in units of 0.1uW (Range 0-6.5mW). If the value is 0, the function is not supported by the current level of microcode.
240	(F0) UNSIGNED	4	SSGLRDTP	Read Diagnostic Parameter: Tx Power. This value contains the measured coupled Tx output power in units of 0.1uW (Range 0-6.5mW). If the value is 0, the function is not supported by the current level of microcode.
244	(F4) UNSIGNED	2	SSGLRDTT	Read Diagnostic Parameter: Internally Measured Transceiver Temperature. This value is reported in units of 1/256 C (Range -128C to +128C). If the value is 0, the function is not supported by the current level of microcode.
246	(F6) UNSIGNED	2	SSGLRDSV	Read Diagnostic Parameter: Internally Measured Supply Voltage. This value is reported in units of 100 uV (Range 0-6.55V). If the value is 0, the function is not supported by the current level of microcode.
248	(F8) UNSIGNED	2	SSGLRDTC	Read Diagnostic Parameter: Measured Transmitter Laser Bias Current. This value is reported in units of 2uA (Range 0-131mA). If the value is 0, the function is not supported by the current level of microcode.
250	(FA) UNSIGNED	2	SSGLRDFG	Read Diagnostic Parameter: Flags 0 Forward Error Correction 0 Forward Error Correction is not active on the link. 1 Forward Error Correction is active on the link. 1-7 Reserved. Set to zero. 8-9 Connector Type 00 Unknown 01 SFP+ 10-11 Reserved 10 SFP Diagnostic Parameters

```
                                                      Validity
                                                      0  Diagnostic Parameters
                                                         are Valid
                                                      1  Diagnostic Parameters are
                                                         NOT Valid. The response
                                                         does not include valid
                                                         values for Temperature,
                                                         Vcc, Tx Bias, Tx Power,
                                                         and Rx Power.
                                                11    Optical Port
                                                      0  Not an Optical Port
                                                      1  Optical Port
                                                12-15 Port Tx Type
                                                      0000      Not Optical or Other
                                                      0001      Short Wave Laser
                                                      0010      Long Wave Laser
                                                                LC 1310nm
                                                      0011      Long Wave Laser
                                                                LL 1550nm
                                                      0100-1111 Reserved
       252     (FC) UNSIGNED         4    SSGLUFEC    Number of Bad Blocks that were
                                                      Uncorrectable by Forward Error
                                                      Correction (FEC)
       256    (100) UNSIGNED         4    SSGLCFEC    Number of Bad Blocks that were
                                                      correctable by Forward Error
                                                      Correction (FEC)
       260    (104) UNSIGNED         4    SSGLTMCR    Transport Mode Command Retries.
                                                      This value is the total number
                                                      of retries requested for
                                                      Transport Mode write operations
                                                      due to not enough buffers
                                                      to receive unsolicited data.
       264    (108) UNSIGNED         4    SSGLTMTR    Transport Mode Transfer Readies.
                                                      This value is the total number
                                                      of Transfer Mode operations
                                                      from channels that require
                                                      transfer ready and were received
                                                      from channels that support
                                                      Transfer Mode Command Retry.
       268    (10C) CHARACTER        8    *           Reserved.


        MAPPING FOR THE SUBSYSTEM GET OUTPUT BUFFER
        FOR RANK PERFORMANCE STATISTICS
        Output buffer is a queue, anchored by SSGRSOAR, of rank
        performance statistics. Each queue element represents data
        read from a single storage subsystem box that describes sets
        of statistics. The queue is a single threaded queue. Each
        queue element needs to be freed by the caller of this
        interface. Output buffer built for storage subsystem boxes
        that support rank performance statistics and are returned
        only if requested by caller (SSGRANKP) on a request for
        performance data (SSGRPD) for requests with scope of
        subsystem (SSG1SS) or all subsystems (SSGALL).
================================================================================
  OFFSET   OFFSET
 DECIMAL     HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
======== ======== ========= ========  ============== ==============================

       0      (0) STRUCTURE        *  SSGRBUFR
       0      (0) CHARACTER       56   SSGRHDR       Queue element header
       0      (0) UNSIGNED         4    SSGRQDAL     Length of this queue element,
                                                      header plus data
       4      (4) UNSIGNED         4    SSGRQDAO     Offset to the start of the
                                                      statistics sets
       8      (8) ADDRESS          4    SSGRFWDP     Pointer to next queue element,
                                                      when zero end of queue
      20     (14) BITSTRING        1    SSGRFLG
                  11.. ....              *           RESERVED
                  ..11 ....              SSGL_SCHST  SUBCHANNEL SET NUMBER
                                                      SCHSET NUMBER 0 - 00
                                                      SCHSET NUMBER 1 - 01
                                                      SCHSET NUMBER 2 - 10
                                                      SCHSET NUMBER 3 - 11
                  .... 1111              *           RESERVED
      21     (15) CHARACTER        7    *           RESERVED

          Orientation to subsystem that stats pertain
================================================================================
      12      (C) CHARACTER        6    SSGRVOL      Volume serial of device that
                                                      stats where read from
      18     (12) CHARACTER        2    SSGRDEVN     Device number of device that
```

```
                                                    Stats where read from
      20     (14) CHARACTER        8    *           Reserved
      28     (1C) CHARACTER       28    SSGRRHDR    Rank stats info
      28     (1C) UNSIGNED         2     SSGRNSET   Num of rank stat sets
                                                    available to be returned on
                                                    the ESS starting from first
                                                    rank ID SSGRRID. Can be used
                                                    to determine buffer
                                                    requirements
      30     (1E) UNSIGNED         2     SSGRLSET   Size of each rank stats set
                                                    excluding size of array info
                                                    SSGRRAR
      32     (20) CHARACTER        6     SSGRCUT    Control unit type
      38     (26) CHARACTER        3     SSGRCUM    Control unit model
      41     (29) CHARACTER       10     SSGRSEQ    Control unit sequence number
      51     (33) CHARACTER        1     SSGRVER    Version of rank statistics
      52     (34) UNSIGNED         2     SSGRARNM   Num of array info sets avail
                                                    to be returned on the ESS
                                                    starting from first rank id
                                                    SSGRRID. Can be used to
                                                    determine buffer requirements
      54     (36) UNSIGNED         2     SSGRARSZ   Size of each array information
                                                    set SSGRRAR
      56     (38) CHARACTER        *   SSGRDADA     Rank performance stats read
      56     (38) CHARACTER       56    SSGRRSTA(*) Rank statistics entry
      56     (38) UNSIGNED         2     SSGRRKID   Rank Identifier

          The returned data consists of rank sets with each rank set
          containing one or more array information entries.
================================================================================
  OFFSET   OFFSET
 DECIMAL     HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
======== ======== ========= ========  ============== ==============================
       0      (0) STRUCTURE        *  SSGRRSTT       Rank stats entry mapping
                                                    includes one or more entries
                                                    for array information
       0      (0) UNSIGNED         2   SSGRRID       Rank identifier
       2      (2) UNSIGNED         2   SSGRRPNM      Extent pool number
       4      (4) UNSIGNED         1   SSGRRCNT      Count of arrays in rank
       5      (5) BITSTRING        1   SSGRRRTQ      Rank Type Qualifier
                  1... ....            SSGRRDER      Data Encrypted Rank. When '1',
                                                    the data stored on the physical
                                                    media (i.e disk) is encrypted.
                  .111 111.        *   RESERVED
                  .... ...1            SSGRRAPV      Adapter Pair ID Valid. When
                                                    '1',the Adapter Pair ID in
                                                    Bytes 6-7 is valid.
       6      (6) CHARACTER        2   SSGRRAPI      Adapter Pair ID

     In the statistics below bytes are accumulated in units of
     128 KB, and time is accumulated in units of 16 milliseconds
================================================================================
       8      (8) UNSIGNED         4   SSGRRBYR      Rank bytes read
      12      (C) UNSIGNED         4   SSGRRBYW      Rank bytes written
      16     (10) UNSIGNED         4   SSGRRROP      Rank read operations
      20     (14) UNSIGNED         4   SSGRRWOP      Rank write operations
      24     (18) UNSIGNED         4   SSGRRKRT      Rank read response time
      28     (1C) UNSIGNED         4   SSGRRKWT      Rank write response time
      32     (20) CHARACTER       24   SSGRRAR(*)    Array information
                                                    mapping
     The format of the next 24 bytes is repeated the number
     times as indicated by SSGRRCNT.
      32     (20) UNSIGNED         2    SSGRRAID     Rank array id
      34     (22) CHARACTER       16    SSGRREBC     Array type in ebcdic
      50     (32) UNSIGNED         1    SSGRRTYP     Array type
                                                    01 = RAID-5
                                                    02 = RAID-10
                                                    03 = RAID-6
                                                    04-FF NOT USED
      51     (33) UNSIGNED         1    SSGRRAWD     Array width
      52     (34) UNSIGNED         1    SSGRRASP     Array speed (1000 RPM)
      53     (35) BITSTRING        1    SSGRRACS     Array Device Class and Array
                                                    Status
                    11.. ....           SSGRRADC     Mask bits for Array Device Class
                                                    Device Class
                                                    '00'b Enterprise
                                                    '01'b Near-line (An Advanced
                                                          Technology Attachment
                                                          (ATA) Drive)
                                                    '10'b SATA (Serial Advanced
                                                          Technology Attachment
                                                          (ATA) Drive)
```

```
                                                  '11'b Solid State Drive (SSD)
                  ..1. ....          SSGRRAS1     Raid Degraded.  One or more
                                                  array members need rebuilding
                  ...1 ....          SSGRRAS2     DDM Throttling.  A Near-line DDM
                                                  in the array is throttling
                                                  performance due to temperature
                                                  or workload.
                  .... 1...          SSGRRAS3     RPM Exception.  A DDM with an
                                                  slower RPM than the normal
                                                  array DDMs is a member of
                                                  the array as a result of
                                                  a sparing action.
       54     (36) CHARACTER       2 SSGRRACP     Array capacity (GB)

    SSGRDADA consists of table entries defined by SSGRDADA1
    shown below when SSGLVER is decimal value of 1.
================================================================================
  OFFSET   OFFSET
 DECIMAL     HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
======== ======== ========= ======== ============== ==============================
      56     (38) CHARACTER       88  SSGRDADA1(*)   Rank performance stats read
      56     (38) UNSIGNED         2  SSGRRKID1      Rank Identifier

    The returned data consists of rank sets with each rank set
    containing one or more array information entries.
================================================================================
  OFFSET   OFFSET
 DECIMAL     HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
======== ======== ========= ========  ============== ==============================
       0      (0) STRUCTURE        *  SSGRRSTT1       Rank stats entry mapping
       0      (0) UNSIGNED        32  *               Mapped in SSGRRSTT
      32     (20) UNSIGNED         4  SSGRARLT        Accumulated Rank Loading Time.
      36     (24) UNSIGNED         4  SSGRDTQC        Number of Times Destages were
                                                      Queued due to Internal
                                                      Processing
      40     (28) UNSIGNED         4  SSGRDTQT        Accumulated Time Destages were
                                                      Queued due to Internal
                                                      Processing
      44     (2C) UNSIGNED         4  SSGRFCBR        Ver 01: Not used. Set to zero.
                                                      Ver 02: Flash Copy Rank Bytes
                                                      Read
      48     (30) UNSIGNED         4  SSGRFCBW        Ver 01: Not used. Set to zero.
                                                      Ver 02: Flash Copy Rank Bytes
                                                      Written
      52     (34) UNSIGNED         4  SSGRFCRO        Ver 01: Not used. Set to zero.
                                                      Ver 02: Flash Copy Rank Read
                                                      Operations
      56     (38) UNSIGNED         4  SSGRFCWO        Ver 01: Not used. Set to zero.
                                                      Ver 02: Flash Copy Rank Write
                                                      Operations
      60     (3C) UNSIGNED         4  SSGRFCRR        Ver 01: Not used. Set to zero.
                                                      Ver 02: Flash Copy Rank Read
                                                      Response Time
      64     (40) UNSIGNED        24  SSGRRAR1(*)     Array information mapping

    The format of the next 24 bytes is repeated the number times as indicated
    by SSGRRCNT.
      64     (40) UNSIGNED         2  SSGRRAID1       Rank array id
      66     (42) CHARACTER       16  SSGRREBC1       Array type in ebcdic
      82     (52) UNSIGNED         1  SSGRRTYP1       Array type
                                                      01 = RAID-5
                                                      02 = RAID-10
                                                      03 = RAID-6
                                                      04-FF NOT USED
      83     (53) UNSIGNED         1  SSGRRAWD1       Array width
      84     (54) UNSIGNED         1  SSGRRASP1       Array speed (1000 RPM)
      85     (55) BITSTRING        1  SSGRRACS1       Array Device Class and Array
                                                      Status
                  11.. ....            SSGRRADC1       Mask bits for Array Device
                                                      Class
                                                      Device Class
                                                      '00'b Enterprise
                                                      '01'b Near-line (An Advanced
                                                            Technology Attachment
                                                            (ATA) Drive)
                                                      '10'b SATA (Serial Advanced
                                                            Technology Attachment
                                                            (ATA) Drive)
                                                      '11'b Solid State Drive (SSD)
                  ..1. ....            SSGRRAS1S1       Raid Degraded. One or more
                                                        array members need rebuilding
                  ...1 ....            SSGRRAS1S2       DDM Throttling. A Near-line
```

```
                                                    DDM in the array is throttling
                                                    performance due to temperature
                                                    or workload.
                 .... 1...           SSGRRAS1S3     RPM Exception. A DDM with an
                                                    slower RPM than the normal
                                                    array DDMs is a member of the
                                                    array as a result of a sparing
                                                    action.
      86     (56) CHARACTER        2  SSGRRACP1     Array capacity (GB)

     SSGRDADA consists of table entries defined by SSGRDADA2 shown below when
     SSGLVER is decimal value of 2.
================================================================================
  OFFSET   OFFSET
 DECIMAL     HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
======== ======== ========= ========  ============== ==============================
      56     (38) CHARACTER      152  SSGRDADA2(*)   Rank performance stats read
      56     (38) UNSIGNED         2  SSGRRKID2      Rank Identifier

     The returned data consists of rank sets with each rank set
     containing one or more array information entries.
================================================================================
  OFFSET   OFFSET
 DECIMAL     HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
======== ======== ========= ========  ============== ==============================
       0      (0) STRUCTURE        *  SSGRRSTT2      Rank stats entry mapping
       0      (0) UNSIGNED        64  *              Mapped in SSGRRSTT1
      64     (40) UNSIGNED         4  SSGRFCRW       Flash Copy Rank Write Response
                                                     Time
      68     (44) UNSIGNED         4  SSGRETBR       Easy Tier Rank Bytes Read
      72     (48) UNSIGNED         4  SSGRETBW       Easy Tier Rank Bytes Written
      76     (4C) UNSIGNED         4  SSGRETRO       Easy Tier Rank Read Operations
      80     (50) UNSIGNED         4  SSGRETWO       Easy Tier Rank Write Operations
      84     (54) UNSIGNED         4  SSGRETRR       Easy Tier Rank Read Response
                                                     Time
      88     (58) UNSIGNED         4  SSGRETRW       Easy Tier Rank Write Response
                                                     Time
      92     (5C) UNSIGNED         4  SSGRCRBR       Cloud Rank Bytes Read
      96     (60) UNSIGNED         4  SSGRCRBW       Cloud Rank Bytes Written
     100     (64) UNSIGNED         4  SSGRCRRO       Cloud Rank Read Operations
     104     (68) UNSIGNED         4  SSGRCRWO       Cloud Rank Write Operations
     108     (6C) UNSIGNED         4  SSGRCRRR       Cloud Rank Read Response Time
     112     (70) UNSIGNED         4  SSGRCRRW       Cloud Rank Write Response Time
     116     (74) CHARACTER       12  *              Reserved
     128     (80) UNSIGNED        24  SSGRRAR2(*)    Array information mapping

     The format of the next 24 bytes is repeated the number times as indicated
     by SSGRRCNT.
     128     (80) UNSIGNED         2  SSGRRAID2      Rank array id
     130     (82) CHARACTER       16  SSGRREBC2      Array type in ebcdic
     146     (92) UNSIGNED         1  SSGRRTYP2      Array type
                                                     01 = RAID-5
                                                     02 = RAID-10
                                                     03 = RAID-6
                                                     04-FF NOT USED
     147     (93) UNSIGNED         1  SSGRRAWD2      Array width
     148     (94) UNSIGNED         1  SSGRRASP2      Array speed (1000 RPM)
     149     (95) BITSTRING        1  SSGRRACS2      Array Device Class and Array
                                                     Status
                  11.. ....           SSGRRADC2      Mask bits for Array Device
                                                     Class
                                                     Device Class
                                                     '00'b Enterprise
                                                     '01'b Near-line (An Advanced
                                                           Technology Attachment
                                                           (ATA) Drive)
                                                     '10'b SATA (Serial Advanced
                                                           Technology Attachment
                                                           (ATA) Drive)
                                                     '11'b Solid State Drive (SSD)
                  ..1. ....           SSGRRAS2S1      Raid Degraded. One or more
                                                      array members need rebuilding
                  ...1 ....           SSGRRAS2S2      DDM Throttling. A Near-line
                                                      DDM in the array is throttling
                                                      performance due to temperature
                                                      or workload.
                  .... 1...           SSGRRAS2S3      RPM Exception. A DDM with an
                                                      slower RPM than the normal
                                                      array DDMs is a member of the
                                                      array as a result of a sparing
                                                      action.
     150     (96) CHARACTER        2  SSGRRACP2      Array capacity (GB)
```

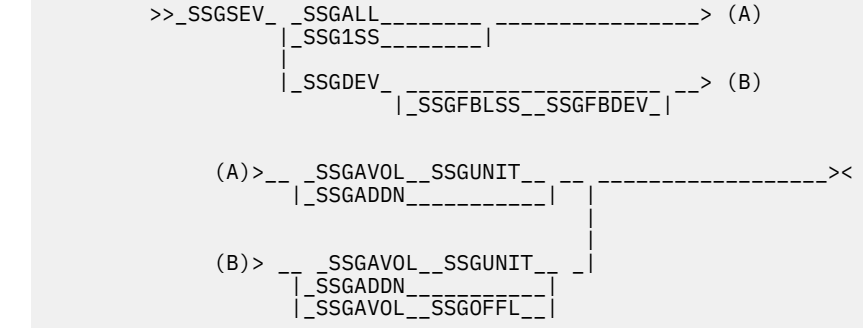
```
        MAPPING FOR THE SUBSYSTEM GET OUTPUT BUFFER
        FOR EXTENT POOL PERFORMANCE STATISTICS
        Output buffer is a queue, anchored by SSGSPOAR, of segment
        pool perf statistics. Each queue element represents data
        read from a single storage subsystem box that describes sets
        of statistics. The queue is a single threaded queue. Each
        queue element needs to be freed by the caller of this
        interface. Output buffer built for storage subsystem boxes
        that support rank performance statistics and are returned
        only if requested by caller (SSGSEGMP) on a request for
        performance data (SSGRPD) for requests with scope of
        subsystem (SSG1SS) or all subsystems (SSGALL).
================================================================================
  OFFSET   OFFSET
 DECIMAL     HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
======== ======== ========= ========  ============== ================================
       0      (0) STRUCTURE        *  SSGSBUFR
       0      (0) CHARACTER       56  SSGSHDR        Queue element header
       0      (0) UNSIGNED         4  SSGSQDAL       Length of this queue element,
                                                      header plus data
       4      (4) UNSIGNED         4  SSGSQDAO       Offset to the start of the
                                                      statistics sets
       8      (8) ADDRESS          4  SSGSFWDP       Pointer to next queue element,
                                                      when zero end of queue

          Orientation to subsystem that status pertains
================================================================================
      12      (C) CHARACTER        6  SSGSVOL        Volume serial of device that
                                                      stats where read from
      18     (12) CHARACTER        2  SSGSDEVN       Device number of device that
                                                      Stats where read from
      20     (14) CHARACTER        8  *              Reserved
      20     (14) BITSTRING        1  SSGSSFLG
                  11.. ....           *              RESERVED
                  ..11 ....           SSGSS_SCHST    SUBCHANNEL SET NUMBER
                                                      SCHSET NUMBER 0 - 00
                                                      SCHSET NUMBER 1 - 01
                                                      SCHSET NUMBER 2 - 10
                                                      SCHSET NUMBER 3 - 11
                  .... 1111             *            RESERVED
      21     (15) CHARACTER        7  *              RESERVED
      28     (1C) CHARACTER       28  SSGSSHDR       Extent pool stats info
      28     (1C) UNSIGNED         2  SSGSNSET       Num of pool stats sets read
      30     (1E) UNSIGNED         2  SSGSLSET       Size of each set
      32     (20) CHARACTER        6  SSGSCUT        Control unit type
      38     (26) CHARACTER        3  SSGSCUM        Control unit model
      41     (29) CHARACTER       10  SSGSSEQ        Control unit sequence number
      51     (33) CHARACTER        1  SSGSVER        Version of pool statistics
      52     (34) BITSTRING        1  SSGSFLG        Flags
                  1... ....           SSGSVLD        Extent pool statistics valid
                                                      When set to one, SSGSSRSC,
                                                      SSGSSVCP, SSGSSNMV, SSGSSVSC,
                                                      SSGSSSDY, and SSGSSTDY are
                                                      valid.
                  .1.. ....           SSGSVLD2       Extent pool statistics valid 2
                                                      When set to one, SSGSSEPS,
                                                      SSGSSBCE, and SSGSSBAE are
                                                      valid.
                  ..11 1111           *              Not used
      53     (35) CHARACTER        3  *              Not used
      56     (38) CHARACTER        *  SSGSDADA       Extent pool perf stats read
      56     (38) CHARACTER       52  SSGSSSTA(*)    Extent pool stats entry
      56     (38) UNSIGNED         2  SSGSSGID       Extent pool identifier

  OFFSET   OFFSET
 DECIMAL     HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
======== ======== ========= ========  ============== ================================
       0      (0) STRUCTURE       52  SSGSSSTT       Extent pool statistics entry
                                                      includes one or more entry
       0      (0) UNSIGNED         2  SSGSSPID       Extent pool identifier
       2      (2) UNSIGNED         1  SSGSSPLT       Extent pool type
                                                       Hex  Description
                                                         04  FB 1 GB
                                                         84  CKD 1 GB(1113 cylinders)
       3      (3) BITSTRING        1  SSGSSPTQ       Extent Pool Type Qualifier
                  1... ....           SSGSSDEP       Data Encrypted Extent Pool.
                                                      When '1', the data stored on
                                                      the physical media (i.e disk)
                                                      is encrypted.
                  .1.. ....           SSGSSETS       Tiered Storage.
                                                      When '1', the Extent Pool
```

```
                                                contains multiple disk
                                                storage classes.
             ..11 1111           *              Reserved, set to zero
             .... ...1           SSGSSESV       Extent Sizes Valid
                                                When '1', SSGSSEPS contains
                                                the extent sizes in the Extent
                                                Pool.
 4      (4) UNSIGNED         4  SSGSSCAP        Real extent pool capacity (GB)
 8      (8) UNSIGNED         4  SSGSSNMS        Num real extents in extent pool
12      (C) UNSIGNED         4  SSGSSNMA        Num real allocated extents in
                                                in extent pool
16     (10) UNSIGNED         4  SSGSSRSC        Number of Extents Allocated
                                                (Real Extent Conversions)
20     (14) UNSIGNED         4  SSGSSVCP        Virtual extent pool
                                                capacity (GB)
24     (18) UNSIGNED         4  SSGSSNMV        Number of virtual extents in
                                                extent pool
28     (1C) UNSIGNED         4  SSGSSVSC        Number of Extents Freed
                                                (Virtual Extent Conversions)
32     (20) UNSIGNED         4  SSGSSSDY        Number of extents that were
                                                sources of a dynamic extent
                                                relocation
36     (24) UNSIGNED         4  SSGSSTDY        Number of extents that were
                                                targets of a dynamic extent
                                                relocation
40     (28) BITSTRING        1  SSGSSEPS        Extent Sizes in Extent Pool
             1... ....           *              Reserved
             .1.. ....           SSGS01GB       1 GB Extents
                                                (CKD 1113 cylinders)
             ..11 111.           *              Reserved
             ....  ...1          SSGS16MB       16 MB Extents
                                                (CKD 21 cylinders)
41     (29) CHARACTER        3  *              Not used
44     (2C) UNSIGNED         4  SSGSSBCE        Safeguarded Copy Virtual
                                                Backup Capacity Extents
48     (30) UNSIGNED         4  SSGSSBAE        Safeguarded Copy Backup
                                                Allocated Extents
```

MAPPING FOR THE SPACE EFFICIENT VOLUME STATUS OUTPUT BUFFER

Output buffer is a queue, anchored by SSGSEOAR, of Space Efficient Volume (SEVOL) status. Each queue element represents status of the space efficient volume specified in the header. The queue is a single threaded queue. Each queue element needs to be freed by the caller of this interface. Output buffer is built for SEVOL(s) when SEVOL status is requested (SSGSEV=ON).
Caller sets the scope of SEVOL(s) by turning one of flag bit that is the single device (SSGDEV), subsystem (SSG1SS), or all subsystems (SSGALL). For a single SEVOL status request, the SEVOL status can be obtain when the path of SEVOL is available. For a subsystem (SSG1SS) and all systems (SSGALL) scope, the SEVOL status is available when SEVOL is ONLINE. When the scope is the single device (SSGDEV=ON) and the Fix Block SEVOL is requested (SSGURFBD=ON), the status of FB SEVOL oriented by LSS (SSGFBLSS) and Device (SSGFBDEV) is reported. FB SEVOL status is not available with the scope of the subsystem (SSG1SS=ON) or the all subsystems (SSGALL=ON). FB SEVOL status is available when LSS number of both the I/O device (SSGAVOL or SSGDDN) and FB SEVOL are numbered in the same even or odd number.

Flag and variable combinations of SEVOL status request are shown as below in the railroad chart format.



MAPPING FOR THE SPACE EFFICIENT VOLUME STATUS OUTPUT

```
        BUFFER
================================================================================
  OFFSET   OFFSET
 DECIMAL     HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
 ======== ======== ========= ======== ============== ==============================
       0      (0) STRUCTURE      124   SSGSEBUF
       0      (0) CHARACTER       68   SSGSEHDR       Queue element header
       0      (0) UNSIGNED         4   SSGSEQAL       Length of this queue
                                                      element, header plus
                                                      data
       4      (4) UNSIGNED         4   SSGSEQAO       Offset to the start
                                                      of the statistics sets
       8      (4) ADDRESS          4   SSGSEFWP       Pointer to next queue
                                                      element, when zero end
                                                      of queue
          Orientation to device that status pertains
       ============================================================

       12    (C) CHARACTER         6  SSGSEVOL       Volume serial of device
                                                      that stat where read
                                                      from
       18   (12) CHARACTER         2  SSGSEDEV       Device number of device
                                                      that Status where read from
       20   (14) CHARACTER         2  SSGSESCX       Subsystem ID
       22   (16) CHARACTER         2  *              not used
       24   (18) UNSIGNED          4  SSGSETTL       Total SE VOL Count,
                                                      only valid for head of
                                                      buffer chain
       28   (1C) CHARACTER        28  SSGSVHDR       SE VOL header
       28   (1C) UNSIGNED          2  SSGSESZ        Size of data
       30   (1E) CHARACTER         6  SSGSECUT       Control unit type
       36   (24) CHARACTER         3  SSGSECUM       Control unit model
       39   (27) CHARACTER         3  SSGSECUC       Control unit maker
                                                      company
       42   (2A) CHARACTER         2  SSGSECUP       Control unit plant
       44   (2C) CHARACTER        12  SSGSESEQ       Control unit sequence
                                                      number
       56   (38) UNSIGNED          4  SSGSEETL       Total ESE VOL Count,
                                                      only valid for head of
                                                      buffer chain
       60   (3C) UNSIGNED          1  SSGSESTL       Total STD VOL Count,
                                                      only valid for head of
                                                      buffer chain
       64   (40) CHARACTER         3  SSGSALCM       Allocation Method
       67   (43) BITSTRING         1  SSGSGFLG       SGC Flag Byte
                 1... ....             SSGSGCBI      Report SGC Backup Vol
                                                      Info
                 .111 1111             *             RESERVED
       68   (44) CHARACTER        60   SSGSEDAD      SE VOL Statistics

          Specified volume
       ============================================================
       68   (44) BITSTRING         1  SSGSEFLG       Flag Byte 1
                 1... ....             SSGSEVF       ON=TSE Volume
                 .1.. ....             SSGSEAN       Space currently
                                                      allocated is not
                                                      available
                 ..1. ....             SSGSEEV       ON=ESE Volume
                 ...1 ....             SSGSEBV       ON=Info Reported for Bckup Vol
                 .... 1...             SSGSEBE       ON=Backup Volume Expanding
                 .... .111             *             not used
       69   (45) CHARACTER         1  *              not used
       70   (46) UNSIGNED          2  SSGSEEP        Extend Pool ID

          The following two fields are a remaining percentage of
          usable space that will initiate a notification to the host
       ============================================================
       72   (48) UNSIGNED          1  SSGSECWW       Capacity Limit
                                                      Warning Watermark
       73   (49) UNSIGNED          1  SSGSEGWW       Guaranteed Capacity
                                                      Warning Watermark
       74   (4A) CHARACTER         2  *              not used

          The following fields are expressed in number of
          cylinders for CKD device. The following fields are
          expressed in TENTH of binary Gigabytes for Fixed Block
          device.
       ============================================================
       76   (4C) UNSIGNED          4  SSGSEC         SE Volume Capacity Limit
       80   (50) UNSIGNED          4  SSGSEG         SE Volume Guaranteed
                                                      Capacity
       84   (54) UNSIGNED          4  SSGSESCA       Space Currently Allocated
```

```
                                            in this Volumes Extent
                                            Pool
88   (58) UNSIGNED          4  SSGSEEPR       Size of Space to Configure
                                            Volumes
92   (5C) UNSIGNED          4  SSGSECAP       SE Volume Capacity

   Backup volume
================================================================
96   (60) BITSTRING         1  SSGSEFLGB      Flag Byte 1
          1... ....             SSGSEVFB      ON=TSE Volume
          .1.. ....             SSGSEANB      Space currently
                                              allocated is not
                                              available
          ..1. ....             SSGSEEVB      ON=ESE Volume
          ...1 ....             SSGSEBVB      ON=Info Reported for Bckup Vol
          .... 1...             SSGSEBEB      ON=Backup Volume Expanding
          .... .111             *             not used
97   (61) CHARACTER         1  *              not used
98   (4C) UNSIGNED          2  SSGSEEPB       Extent Pool ID

   The following two fields are a remaining percentage of
   usable space that will initiate a notification to the
   host.
================================================================
100  (64) UNSIGNED          1  SSGSECWWB      Capacity Limit
                                            Warning Watermark
101  (65) UNSIGNED          1  SSGSEGWWB      Guaranteed Capacity
                                            Warning Watermark
102  (66) CHARACTER         2  *              not used

   The following fields are expressed in number of
   cylinders for CKD device.   The following fields are
   expressed in TENTH of binary Gigabytes for Fixed Block
   device.
================================================================
104  (68) UNSIGNED          4  SSGSECB        SE Volume capacity
                                            Limit
108  (6C) UNSIGNED          4  SSGSEGB        SE Volume Guaranteed
                                            Capacity
112  (70) UNSIGNED          4  SSGSESCAB      Space Currently
                                            Allocated in this
                                            Volumes Extent Pool
116  (74) UNSIGNED          4  SSGSEEPRB      Size of Space to
                                            Configure Volumes
120  (70) UNSIGNED          4  SSGSECAPB      SE Volume Capacity
124  (74) UNSIGNED          4  SSGSESEQB      Sequence Number of
                                            the last Consistency
                                            Group in the prior
                                            SGC Backup Capacity
                                            Volume Size


MAPPING FOR THE EXTENT POOL CONFIGURATION STATUS
OUTPUT BUFFER
Output buffer is a queue, anchored by SSGEPOAR, of Extent
Pool Configuration status.  Each queue element represents
the status of the Extent Pools defined in Storage Facility
image. If the caller provide a Extent Pool ID ( SSGEPID
has value between '0'X and 'FFFE'X ), then in addition to
the summary status of Extent Pool, the detail status of
that Extent Pool identified by Extent Pool ID is reported.
Caller is responsible for freeing the output buffer.

The flag and variable combinations for EPC status is
shown as below railroad chart.

            Below variable defines scope

>>__SSGEPC__ _SSGEPID=('FFFF'X)_____________________>(A)
            |_SSGEPID=('0'X to 'FFFE'X of even number)_>(B)
            |_SSGEPID=('1'X to 'FFFD'X of odd number)__>(C)


 (A)>_____________________________ _____________________>
 (B)>_(Even LSS I/O dev follows)_ _|
 (C)>_(Odd LSS I/O dev follows)__|

    Following variable and flag represent I/O device

 >______ _SSGAVOL__SSGUNIT__ _______________________________><
        |_SSGADDN___________|
        |_SSGAVOL__SSGOFFL__|
```

MAPPING FOR THE EXTENT POOL CONFIGURATION STATUS OUTPUT BUFFER

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	*	SSGEPBUF	
0	(0)	CHARACTER	44	SSGEHDR	Queue element header
0	(0)	UNSIGNED	4	SSGEPQAL	Length of this queue element, header plus data
4	(4)	UNSIGNED	4	SSGEPQAO	Offset to the start of the statistics sets
8	(8)	UNSIGNED	4	SSGEPFWD	Pointer to next queue element, when zero end of queue
		Orientation to device that stats pertain			
12	(C)	CHARACTER	6	SSGEPVOL	Volume serial of device that status where read from
18	(12)	CHARACTER	2	SSGEPDEV	Device number of device that tats where read from
20	(14)	UNSIGNED	2	SSGEPIDC	Ext Pool ID provided by caller
22	(16)	CHARACTER	6	*	Reserved
28	(1C)	ADDRESS	4	SSGEPSIP	Internal code use only. Branch Entry caller shall set zero.
32	(20)	ADDRESS	4	SSGEPSI1P	Internal code use Only. Branch Entry caller shall set zero.
36	(24)	ADDRESS	4	SSGEPSI2P	Internal code use Only. Branch Entry caller shall set zero.
40	(28)	UNSIGNED	2	SSGEPSDO	Offset to Summary Data
42	(2A)	UNSIGNED	2	SSGEPDDO	Offset to Detail Data
44	(2C)	CHARACTER	*	SSGEPDAD	Ext Pool Config status
44	(2C)	CHARACTER	32	SSGEPDHL	Fixed length of EPC data header
44	(2C)	UNSIGNED	2	SSGEPLEN	Length of data returned
46	(2E)	UNSIGNED	2	SSGEPCNT	Count of Extent Pools
48	(30)	BITSTRING	2	SSGEPFL1	Header Flags
		1...		SSGEPIDV	Ext Pool detailed data valid
		.1..		SSGEPLSS	Bitmap represents even LSS's when set to zero and odd LSS's when set to one.
		..1.		SSGEPCC	Specified Extent Pool ID owned by Partner CEC, no data reported in Extent Pool Detailed Information.
		...1 1111			Not Used
49	(31)	1111 1111			Not Used
50	(32)	CHARACTER	6	SSGEPCUT	Control unit type
56	(38)	CHARACTER	3	SSGEPCUM	Control unit model
59	(3B)	CHARACTER	3	SSGEPCUC	Control unit maker/company
62	(3E)	CHARACTER	2	SSGEPCUP	Control unit plant
64	(40)	CHARACTER	12	SSGEPSEQ	Control unit sequence number
		The format of the next eight bytes is repeated the number times as indicated by SSGEPCNT.			
76	(4C)	CHARACTER	8	SSGEPSUM	Ext Pool summary data
76	(4C)	CHARACTER	2	SSGEPIDS	Extent Pool ID
78	(4E)	CHARACTER	1	SSGEPRWP	SE Repository Capacity Available Warning Percentage
79	(4F)	CHARACTER	1	SSGEPFWP	Real Ext Threshold Percentage
80	(50)	BITSTRING	2	SSGEPFLG	Ext Pool Summary Flags
		1...		SSGEPFB	Extent Pool Type 0=CKD Ext Pool, 1=FB Ext Pool
		.1..		SSGEPRC	TSE Volumes Repository Configured
		..1.		SSGEOPV	ESE Volumes Configured
		...1		SSGESTD	STD (FP) Volumes

```
                                                    Configured
                .... 1...              SSGEESV      Extent Sizes Valid
                                                    When set to one, SSGEESZ
                                                    contains the extent sizes
                                                    in the Extent Pool.
                .... .1..              SSGEPWP      Real Capacity at or
                                                    below Warning Percentage
                .... ..1.              SSGEPF       No Available Real Capacity
                                                    (i.e. Full)
                .... ...1              SSGESGC      SGC Backup Volumes
                                                    Configured
    82     (52) BITSTRING         1   SSGEESZ      Extent Sizes in Extent Pool
                1... ....              *            Reserved
                .1.. ....              SSGS01GB     1 GB Extents
                ..11 111.              *            Reserved
                .... ...1              SSGS16MB     16 MB Extents
    83     (53) CHARACTER         1    *           not used

         The following fields contain detailed Extent Pool
         configuration information for Extent Pool identified
         by Ext Pool ID specified by caller (Ver 1).  The address
         of SSGEPDAT is the sum of SSGEPOAR and SSGEPDDO.
================================================================================
 DECIMAL    HEX    TYPE        LENGTH  NAME (DIM)     DESCRIPTION
 OFFSET   OFFSET
======== ======== ========= ========  ============== ==============================
      0      (0)  CHARACTER   12316    SSGEPDAT
      0      (0)  CHARACTER       2    SSGEPIDD      Ext Pool ID for detailed
                                                    data
      2      (2) BITSTRING        1    SSGEPDFG      Extent Pool Detailed Flags
                 1... ....              SSGEPARN     SE Repository Capacity
                                                    is Not Available
                 .1.. ....              SSGEPRAW     SE Repository at or below
                                                    Warning Percentage
                 ..1. ....              SSGEPRF      No Available SE Repository
                                                    Capacity (i.e. full)
                 ...1 ....              SSGEPESV     Extent Sizes Valid
                                                    When set to one, SSGEPESZ
                                                    contains the extent sizes
                                                    in the Extent Pool.
                 .... 1111              *            RESERVED
      3      (3) BITSTRING        1    SSGEPESZ      Extent Sizes in Extent Pool
                 1... ....              *            Reserved
                 .1.. ....              SSGED01GB    1 GB Extents
                 ..11 111.              *            Reserved
                 .... ...1              SSGED16MB    16 MB Extents

         The following fields are expressed in number of cylinders
         for CKD device. The following fields are expressed
         in TENTH of binary Gigabytes for Fixed Block device.
       ============================================================
      4      (4) UNSIGNED         4    SSGEPSZ       Sizes of Extent Pool
      8      (8) UNSIGNED         4    SSGEALOC      Space Currently
                                                    Allocated in Extent
                                                    Pool
     12      (C) UNSIGNED         4    SSGEPRSZ      Size of TSE Extent
                                                    Pool Repository
     16     (10) UNSIGNED         4    SSGEPRAL      Space Currently
                                                    Allocated in TSE
                                                    Extent Pool Repository
     20     (14) UNSIGNED         4    SSGEPGSZ      Amount of Guaranteed
                                                    Space in Repository
     24     (18) UNSIGNED         4    SSGEPGAL      Allocated Guaranteed
                                                    Capacity

         The next three fields are 32k bitmaps
       ============================================================
     28    (1C) BITSTRING       4096 SSGENORM        Standard (FP) Vols
   4124  (101C) BITSTRING       4096 SSGEPOPV        Extent Space Efficient
                                                    Vols
   8220  (201C) BITSTRING       4096 SSGEPSEV        Track Space Efficient
                                                    Vols

         The following fields contain detailed Extent Pool
         configuration information for Extent Pool identified
         by Ext Pool ID specified by caller (Ver 2).  The address
         of SSGEPDV2 is the sum of SSGEPOAR and SSGEPDDO.
================================================================================
 DECIMAL    HEX    TYPE        LENGTH  NAME (DIM)     DESCRIPTION
 OFFSET   OFFSET
======== ======== ========= ========  ============== ==============================
      0      (0)  CHARACTER   16512    SSGEPDV2
```

```
     0      (0)  CHARACTER       2    SSGEPIDD2     Ext Pool ID for detailed
                                                     data
     2      (2) BITSTRING        1    SSGEPDFG2     Extent Pool Detailed Flags
                1... ....             SSGEPARN2     SE Repository Capacity
                                                     is Not Available
                .1.. ....             SSGEPRAW2     SE Repository at or below
                                                     Warning Percentage
                ..1. ....             SSGEPRF2      No Available SE Repository
                                                     Capacity (i.e. full)
                ...1 ....             SSGEPESV2     Extent Sizes Valid
                                                     When set to one, SSGEPESZ2
                                                     contains the extent sizes
                                                     in the Extent Pool.
                .... 1111             *             RESERVED
     3      (3) BITSTRING        1    SSGEPESZ2     Extent Sizes in Extent Pool
                1... ....             *             Reserved
                .1.. ....             SSGED01GB2    1 GB Extents
                ..11 111.             *             Reserved
                .... ...1             SSGED16MB2    16 MB Extents

          The following fields are expressed in number of cylinders
          for CKD device.   The following fields are expressed
          in TENTH of binary Gigabytes for Fixed Block device.
       ============================================================
     4      (4) UNSIGNED         4    SSGEPSZ2      Sizes of Extent Pool
     8      (8) UNSIGNED         4    SSGEALOC2     Space Currently
                                                     Allocated in Extent
                                                     Pool
    12      (C) UNSIGNED         4    SSGEPRSZ2     Size of TSE Extent
                                                     Pool Repository
    16     (10) UNSIGNED         4    SSGEPRAL2     Space Currently
                                                     Allocated in TSE
                                                     Extent Pool Repository
    20     (14) UNSIGNED         4    SSGEPGSZ2     Amount of Guaranteed
                                                     Space in Repository
    24     (18) UNSIGNED         4    SSGEPGAL2     Allocated Guaranteed
                                                     Capacity
    28     (1C) UNSIGNED         4    SSGEPECC2     ESE Volumes Capacity
                                                     Configured
    32     (20) UNSIGNED         4    SSGEPTCC2     STD (FP) Volumes
                                                     Capacity Configured
    36     (24) UNSIGNED         4    SSGEPFCC2     TSE Volumes
                                                     Capacity Configured
    40     (28) UNSIGNED         4    SSGEPRMC2     Amount of Real
                                                     Capacity consumed
                                                     in provisioning
                                                     Metadata Virtual
                                                     Capacity
    44     (2C) UNSIGNED         4    SSGEPECU2     ESE Volumes
                                                     Capacity Consumed
    48     (30) UNSIGNED         4    SSGEPSVC2     Size of Customer
                                                     Virtual Capacity
                                                     Configured in Extent
                                                     Pool
    52     (34) UNSIGNED         4    *             Not Used
    56     (38) UNSIGNED         4    SSGEPVCP2     Customer Virtual
                                                     Capacity in
                                                     Preparation Phase
    60     (3C) UNSIGNED         4    SSGEPPCP2     Amount of Physical
                                                     Capacity in
                                                     Preparation Phase
    64     (40) UNSIGNED         4    SSGEPCPI2     Amount of Capacity
                                                     Performing
                                                     Initialization
    68     (44) UNSIGNED         4    SSGEPSCC2     SGC Backup Volumes
                                                     Capacity Configured
    72     (48) UNSIGNED         4    SSGEPSCU2     SGC Backup Volumes
                                                     Capacity Consumed
    76     (4C) CHARACTER       52    *             Reserved

          The next four fields are 32k bitmaps
       ============================================================
   128     (80) BITSTRING       4096 SSGENORM2      Standard (FP) Vols
  4224   (1080) BITSTRING       4096 SSGEPOPV2      Extent Space Efficient
                                                     Vols
  8320   (2080) BITSTRING       4096 SSGEPSEV2      Track Space Efficient
                                                     Vols
 12416   (3080) BITSTRING       4096 SSGEPSGC2      Safeguarded Copy
                                                     Backup Vols

          The following fields contain detailed Extent Pool
          configuration information for Extent Pool identified
```

```
          by Ext Pool ID specified by caller (Ver 3).  The address
          of SSGEPDV3 is the sum of SSGEPOAR and SSGEPDDO.
================================================================================
 DECIMAL    HEX    TYPE        LENGTH  NAME (DIM)     DESCRIPTION
 OFFSET   OFFSET
======== ======== ========= ========  ============== ==============================
      0      (0)  CHARACTER   16896    SSGEPDV3
      0      (0)  CHARACTER       2    SSGEPIDD3      Ext Pool ID for detailed
                                                      data
      2      (2) BITSTRING        2    SSGEPDFG3      Extent Pool Detailed Flags
                 1... ....             SSGEPFB3       Extent Pool Type
                                                      0=CKD Ext Pool,
                                                      1=FB Ext Pool
                 .1.. ....             SSGEPRC3       TSE Volumes Repository
                                                      Configured
                 ..1. ....             SSGEOPV3       ESE Volumes Configured
                 ...1 ....             SSGESTD3       STD (FP) Volumes
                                                      Configured
                 .... 1...             SSGEESV3       Extent Sizes Valid
                                                      When set to one, SSGEPESZ3
                                                      contains the extent sizes
                                                      in the Extent Pool.
                 .... .1..             SSGRCWP3       Real Capacity at or
                                                      below Warning Percentage
                 .... ..1.             SSGRCF3        No Available Real
                                                      Capacity (i.e. full)
                 .... ...1             SSGEPRAW3      SE Repository at or
                                                      below Warning Percentage
      3      (3) 1... ....             SSGEPRF3       No Available SE
                                                      Repository Capacity
                                                      (i.e. full)
                 .1.. ....             SSGESGC3       SGC Backup Volumes
                                                      Configured
                 ..11 11..             *              Not Used
                 .... ..1.             SSGEPARN3      SE Repository Capacity
                                                      is Not Available
                 .... ...1             SSGEPRIS3      SE Repository contains
                                                      Inaccessible Space
      4      (4) CHARACTER        1     SSGEPETP3      Extent Type
                                                      Hex  Description
                                                      04   FB 1 GB
                                                      84   CKD 1 GB
      5      (5) BITSTRING        1     SSGEPESZ3      Extent Sizes in Extent Pool
                 1... ....              *              Reserved
                 .1.. ....              SSGED01GB3     1 GB Extents
                 ..11 111.              *              Reserved
                 .... ...1              SSGED16MB3     16 MB Extents
      6      (0)  CHARACTER       2    *              Reserved
      8      (0)  UNSIGNED        4    SSGEPNCM3      Number of Cylinders
                                                      per Extent or Number of
                                                      Megabytes per Extent
     12      (C)  CHARACTER       4    *              Reserved
     16     (10)  UNSIGNED        1    SSGEPRLP3      Real Extent Limit
                                                      Percentage
     17     (11)  UNSIGNED        1    SSGEPFWP3      Real Extent Threshold
                                                      Percentage
     18     (12)  CHARACTER       1    SSGEPRES3      Real Extent Status
                                                      Value Description
                                                      00    Percentage
                                                            of Available
                                                            Real Extents
                                                            > 0 and is >
                                                            extent threshold
                                                      01    Percentage
                                                            of Available
                                                            Real Extents
                                                            > 0 and is <
                                                            extent threshold
                                                      10    Percentage
                                                            of Available
                                                            Real Extents
                                                            is 0
     19     (13)  UNSIGNED        1    SSGRCPAL3      Allocated Real Capacity
                                                      Percentage
     20     (14)  UNSIGNED        1    SSGRCPAV3      Available Real Capacity
                                                      Percentage
     21     (15)  UNSIGNED        1    SSGMNERP3      Minimum ESE Region
                                                      Percentage
     22     (16)  UNSIGNED        1    SSGMXERP3      Maximum ESE Region
                                                      Percentage
     23     (17)  UNSIGNED        1    SSGESETP3      ESE Threshold
                                                      Percentage
```

```
 24     (18)  UNSIGNED        1    SSGESESP3     ESE Status Pool
                                                  Percentage
 25     (19)  UNSIGNED        1    SSGALECP3     Allocated ESE Real
                                                  Capacity Percentage
 26     (1A)  UNSIGNED        1    SSGAVECP3     Available ESE Real
                                                  Capacity Percentage
 27     (1B)  UNSIGNED        1    SSGISRCP3     Percentage of Inaccessible
                                                  SE Repository Capacity
 28     (1C)  CHARACTER      36    *             Reserved

      The following fields are expressed in number of cylinders
      for CKD device.   The following fields are expressed
      in TENTH of binary Gigabytes for Fixed Block device.
   ============================================================
 64     (40) UNSIGNED         8    SSGREPCP3     Real Extent Pool
                                                  Capacity
 72     (48) UNSIGNED         8    SSGPVRCP3     Preserved Real
                                                  Capacity
 80     (50) UNSIGNED         8    SSGRTRCP3     Restricted Real
                                                  Capacity
 88     (58) UNSIGNED         8    SSGRVRCP3     Reserved Real
                                                  Capacity
 96     (60) UNSIGNED         8    SSGABRCP3     Allocatable Real
                                                  Capacity
104     (68) UNSIGNED         8    SSGALRCP3     Allocated Real
                                                  Capacity
112     (70) UNSIGNED         8    SSGFPARC3     STD (FP) Logical
                                                  Volumes Allocated
                                                  Real Capacity
120     (78) UNSIGNED         8    SSGESARC3     ESE Logical
                                                  Volumes Allocated
                                                  Real Capacity
128     (80) UNSIGNED         8    SSGMVARC3     Allocated Real
                                                  Capacity used to
                                                  Provision Metadata
                                                  Virtual Capacity
136     (88) UNSIGNED         8    SSGTSARC3     Allocated Real
                                                  Capacity used to
                                                  Provision TSE
                                                  Repository
144     (90) UNSIGNED         8    SSGSDARC3     Allocated Real
                                                  Capacity used for
                                                  Source Dynamic
                                                  Extent Relocation
152     (98) UNSIGNED         8    SSGTDARC3     Allocated Real
                                                  Capacity used for
                                                  Target Dynamic
                                                  Extent Relocation
160     (A0) UNSIGNED         8    SSGRETRC3     Easy Tier Real
                                                  Capacity
168     (A8) UNSIGNED         8    SSGRCPRP3     Real Capacity in
                                                  Preparation Phase
176     (B0) UNSIGNED         8    SSGAVRCP3     Available Real
                                                  Capacity
184     (B8) UNSIGNED         8    SSGRCPIN3     Real Capacity
                                                  Performing
                                                  Initialization
192     (C0) UNSIGNED         8    SSGMVCEP3     Metadata Virtual
                                                  Capacity in Extent
                                                  Pool
200     (C8) UNSIGNED         8    SSGPVCAP3     Preserved Capacity
208     (D0) CHARACTER        8    SSGRTCAP3     Restricted Capacity
216     (D8) UNSIGNED         8    SSGMVCAP3     Metadata Virtual
                                                  Capacity
224     (E0) UNSIGNED         8    SSGABMVC3     Allocatable Metadata
                                                  Virtual Capacity
232     (E8) UNSIGNED         8    SSGCCVVC3     Configured Customer
                                                  Volume Virtual
                                                  Capacity
240     (F0) UNSIGNED         8    SSGCESVC3     Configured ESE
                                                  Volumes Virtual
                                                  Capacity
248     (F8) UNSIGNED         8    SSGCTSVC3     Configured TSE
                                                  Volumes Virtual
                                                  Capacity
256    (100) UNSIGNED         8    SSGSDERC3     Source Dynamic
                                                  Extent Relocation
                                                  Capacity
264    (108) UNSIGNED         8    SSGTDERC3     Target Dynamic
                                                  Extent Relocation
                                                  Capacity
272    (110) UNSIGNED         8    *             Reserved
```

```
  280    (118) UNSIGNED         8    SSGMVCPP3     Metadata Virtual
                                                   Capacity in
                                                   Preparation Phase
  288    (120) UNSIGNED         8    SSGAVMVC3     Available Metadata
                                                   Virtual Capacity
  296    (128) UNSIGNED        32    *             Reserved
  328    (148) UNSIGNED         8    SSGTSERC3     TSE Repository
                                                   Capacity
  336    (150) UNSIGNED         8    SSGALTSC3     Allocated TSE
                                                   Repository Capacity
  344    (158) UNSIGNED         8    SSGAVTSC3     Available TSE
                                                   Repository Capacity
  352    (160) UNSIGNED         8    SSGIATSC3     Inaccessible TSE
                                                   Repository Capacity
  360    (168) UNSIGNED         8    SSGGTSRC3     Guaranteed TSE
                                                   Repository Capacity
  368    (170) UNSIGNED         8    SSGPTSRC3     Preserved TSE
                                                   Repository Capacity
  376    (178) UNSIGNED        32    *             Reserved
  408    (198) CHARACTER        8    SSGMNERC3     Minimum ESE Region
                                                   Real Capacity
  416    (1A0) UNSIGNED         8    SSGMXERC3     Maximum ESE Region
                                                   Real Capacity
  424    (1A8) UNSIGNED         8    SSGABERC3     Allocatable ESE
                                                   Region Real
                                                   Capacity
  432    (1B0) UNSIGNED         8    SSGAVERC3     Available ESE
                                                   Region Real
                                                   Capacity
  440    (1B8) UNSIGNED         4    SSGSGARC3     SGC Backup Volumes
                                                   Allocated Real
                                                   Capacity
  444    (1BC) UNSIGNED         4    SSGCSGVC3     Configured SGC
                                                   Backup Volumes
                                                   Virtual Capacity
  448    (1C0) CHARACTER       64    *             Reserved

        The next four fields are 32k bitmaps
     ============================================================
  512    (200) BITSTRING       4096 SSGENORM3      Standard (FP) Vols
 4608   (1200) BITSTRING       4096 SSGEPOPV3      Extent Space Efficient
                                                   Vols
 8704   (2200) BITSTRING       4096 SSGEPSEV3      Track Space Efficient
                                                   Vols
12800   (3200) BITSTRING       4096 SSGEPSGC3      Safeguarded Copy
                                                   Backup Vols

        MAPPING FOR THE DEVICE PERFORMANCE STATISTICS
        OUTPUT BUFFER
        Output buffer is anchored by SSGOADR, of a device performance
        statistics output table.   Output buffer needs to be freed
        by the caller of this interface. Output buffer is built
        for a device that supports device performance statistics and
        returned only if requested by caller for performance data
        (SSGRPD) for a request with scope of of device (SSGDEV).
        The parameter list extension must also be passed as
        indicated with SSGARGL2.
================================================================================

  OFFSET   OFFSET
 DECIMAL     HEX  TYPE        LENGTH  NAME (DIM)     DESCRIPTION
======== ======== ========= ========  ============== ==============================
       0      (0) STRUCTURE      256  SSGDPBUF       DEVICE PERFORMANCE
                                                     STATISTICS OUTPUT BUFFER
       0      (0) CHARACTER        6  SSGDPVOL       VOLUME SERIAL
       6      (6) BITSTRING        1  SSGDPFLG
                  1... ....            SSGDP_UA_FLAG  'ONE' UNIT ADDR FORMAT
                  .1.. ....            SSGDP_FND_STG2 2ND STG DIR PATH FND
                                                       SS01 TO LA01 IDCSS01
                                                       PASSED TO IDCLA01
                  ..11 ....            SSGDP_SCHST   SUBCHANNEL SET NUMBER
                                                      SCHSET NUMBER 0 - 00
                                                      SCHSET NUMBER 1 - 01
                                                      SCHSET NUMBER 2 - 10
                                                      SCHSET NUMBER 3 - 11
                  .... 1111            *             RESERVED
       7      (7) CHARACTER        1   SSGDPCUI      Real CUID for data
                                                     in SSGDPDA
       8      (8) CHARACTER        4   SSGDPUA1      FIRST UNIT ADDRESS
      12      (C) CHARACTER        2   SSGDPUB1      FIRST UNIT ADDRESS
                                                     (BINARY)
      14      (E) SIGNED           2   SSGDPLN       DATA LENGTH
```

```
16     (10) CHARACTER      240   SSGDPDA        DEVICE PERFORMANCE
                                                STATISTICS DATA AREA
16     (10) BITSTRING        1    SSGDPFTF
          1... ....               SSGDPNAV     Bit =1 : Cache Storage
                                               unavailable (Set to zero
                                               for 2107/1750)
          .111 ....               SSGDPPDR     Mask bits for Format of
                                               Perf Data Returned
                                               Value after masking
                                               ( SSGDPFTF & SSGDPPDR )
                                               '000'b 3990 Format
                                               '001'b 2107/1750 Format
          .... 1111               SSGDPFDR     Mask bits for Format of
                                               data returned.
                                               The same definition and
                                               values of bit 4-7 of
                                               byte 0 from Sense
                                               Subsystem status.
17     (11) CHARACTER        1    SSGDPDUA     Device Unit Address of
                                               the device to which the
                                               statistics pertain
18     (12) CHARACTER        2    SSGDPDST     Device Status. The same
                                               as Sense Subsystem
                                               Status in bytes 26-27
20     (14) UNSIGNED         4   SSGDPRNI     Search/Read Normal I/O
                                               Requests.  The number of
                                               command chains which meet
                                               the following requirements.
                                               - The chain is not part of
                                                 a sequential operation.
                                               - The chain did not include
                                                 a Define Extent command
                                                 which specified Cache
                                                 Fast Write Data.
                                               - The chain contained at
                                                 least one search or read
                                                 command but no write
                                                 commands.
24     (18) UNSIGNED         4   SSGDPRNH     Search/Read Normal I/O
                                               Request Hits.  The number of
                                               command chains which meet
                                               the following requirements.
                                               - The chain is not part of
                                                 a sequential operation.
                                               - The chain did not include
                                                 a Define Extent command
                                                 which specified Cache Fast
                                                 Write Data.
                                               - The chain contained at
                                                 least one search or read
                                                 command but no write
                                                 commands.
                                               - The chain was completed
                                                 without requiring access
                                                 to any DDM.
28     (1C) UNSIGNED         4   SSGDPWNI     Write Normal I/O Requests.
                                               The number of command chains
                                               which meet the following
                                               requirements.
                                               - The chain is not part of
                                                 a sequential operation.
                                               - The chain did not include
                                                 a Define Extent command
                                                 which specified Cache
                                                 Fast Write Data.
                                               - The chain contained at
                                                 least one write command.
32     (20) UNSIGNED         4   SSGDPFWH     DASD Fast Write I/O Request
                                               Hits.  The number of command
                                               chains which meet the
                                               following requirements.
                                               - The chain is not part of
                                                 a sequential operation.
                                               - The chain did not include
                                                 a Define Extent command
                                                 which specified Cache Fast
                                                 Write Data.
                                               - The chain contained at
                                                 least one write command.
                                               - The chain was completed
                                                 without requiring access
                                                 to any DDM.
```

36	(24)	UNSIGNED	4	SSGDPRSI	Search/Read Sequential I/O Requests. The number of sequential mode command chains which meet the following requirements. - The chain contained at least one search or read command but no write commands.
40	(28)	UNSIGNED	4	SSGDPRSH	Search/Read Sequential I/O Request Hits. The number of sequential mode command chains which meet the following requirements. - The chain contained at least one search or read command but no write commands. - The chain was completed without requiring access to any DDM.
44	(2C)	UNSIGNED	4	SSGDPWSI	Write Sequential I/O Requests. The number of sequential mode command chains which meet the following requirements. - The chain contained at least one write command.
48	(30)	UNSIGNED	4	SSGDPWSH	Fast Write Sequential I/O Request Hits. The number sequential write operations that did not require movement of data to or from a storage device before completion of the I/O operation.
52	(34)	UNSIGNED	4	SSGDPRCF	Search/Read Cache Fast Write I/O Requests. The number of command chains which meet the following requirements. - The chain included a Define Extent command which specified the Cache Fast Write Data attribute. - The chain contained at least one search or read command but no write commands. - Cache Fast Write Data must be activated for the subsystem.
56	(38)	UNSIGNED	4	SSGDPRCH	Search/Read Cache Fast Write I/O Request Hits. The number of command chains which meet the following requirements. - The chain included a Define Extent command which specified the Cache Fast Write Data attribute. - The chain contained at least one search or read command but no write commands. - Cache Fast Write Data must be actuated for the subsystem. - The chain was completed without requiring access to any DDM.
60	(3C)	UNSIGNED	4	SSGDPCFW	Cache Fast Write I/O Requests. The number of command chains which meet the following requirements. - The chain included a Define Extent command which specified the Cache Fast Write Data attribute. - The chain contained at least one write command. - Cache Fast Write Data must be actuated for the subsystem.
64	(40)	UNSIGNED	4	SSGDPCFH	Cache Fast Write I/O Request

```
                                                 Hits.  The number of command
                                                 chains which meet the
                                                 following requirements.
                                                 - The chain included a Define
                                                   Extent command which
                                                   specified the Cache Fast
                                                   Write Data attribute.
                                                 - The chain contained at
                                                   least one write command.
                                                 - Cache Fast Write Data must
                                                   be actuated for the
                                                   subsystem.
                                                 - The chain was completed
                                                   without requiring access
                                                   to any DDM.
 68     (44) UNSIGNED         4    SSGDPICL     Inhibit Cache Loading I/O
                                                 Request that operate with
                                                 DASD (Set to zeros for
                                                 2107/1750)
 72     (48) UNSIGNED         4    SSGDPBCI     Bypass Cache I/O Requests
                                                 (Set to zeros for
                                                 2107/1750)
 76     (4C) UNSIGNED         4    SSGDPSDC     Seq DASD to Cache
                                                 Transfer Operations
 80     (50) UNSIGNED         4    SSGDPDCC     DASD to Cache Transfer
                                                 Operation Count
 84     (54) UNSIGNED         4    SSGDPCDC     Cache to DASD Transfer
                                                 Operation Count
 88     (58) UNSIGNED         4    SSGDPFWD     DASD Fast Write Operation
                                                 Delayed Due to
                                                 non-volatile storage
                                                 Space constraints
 92     (5C) UNSIGNED         4    SSGDPNFW     Normal 'DASD Fast Write'
                                                 Write Operation Counts.
                                                 The number of command chains
                                                 which meet the following
                                                 requirements.
                                                 - The chain is not part of
                                                   a sequential operation.
                                                 - The chain did not include
                                                   a Define Extent command
                                                   which specified Cache Fast
                                                   Write Data.
                                                 - The chain contained at
                                                   least one write command.
                                                 - The chain was completed
                                                   without requiring access
                                                   to any DDM.
 96     (60) UNSIGNED         4    SSGDPSFW     Sequential Access 'DASD
                                                 Fast Write' Write
                                                 Operation Counts
100     (64) UNSIGNED         4    SSGDPNRM     Number of record cache
                                                 Read misses
104     (68) CHARACTER        1    SSGDPDS2     Device Status - Group 2
                                                 The device status,
                                                 SSGDPDS2, is identical
                                                 to the data returned by
                                                 the Sense Subsystem
                                                 Status in Byte 36.
                                                 Note: The format of the
                                                 data in SSGDPDS2 is
                                                 determined by the value
                                                 in SSGDPFDR.
105     (69) UNSIGNED         4    SSGDPQWP     Quick Write Promotes
109     (6D) BITSTRING        1    SSGDPDVD     Flag byte
                                                 SSGDPDVD describes the
                                                 validity of data returned
                                                 and the increment values
                                                 used for certain counters

            1... ....              SSGDPCDT     When set on, SSGDPRDT,
                                                 SSGDPWDT, SSGDPRDC, and
                                                 SSGDPWDC are valid.
            .11. ....              SSGDPRVA     Not used. Set to zero.
            ...1 ....              SSGDPSIV     When set on,
                                                 SSGDPSRR,
                                                 SSGDPSRH,
                                                 SSGDPSWR, and
                                                 SSGDPSWH are
                                                 valid.

            .... 1...              SSGDPZHV     When set on, SSGDPZHR and
```

```
                                          SSGDPZHW are valid.
          .... .11.          SSGDPUNI     Mask bits for Units
                                          Value after masking
                                          ( SSGDPDVD & SSGDPUNI )
                                          '00'b  Unit of SSGDPRRT,
                                                 SSGDPWRT,SSGDPBRD,
                                                 SSGDPBWR,SSGDPRAT,
                                                 and SSGDPWAT is
                                                 16 milliseconds.
                                                 Unit of SSGDPPBR
                                                 SSGDPPBW is
                                                 128 KB.
                                          '01'b  Reserved
                                          '10'b  Reserved
                                          '11'b  Reserved
          .... ...1          SSGDPBYV     1 indicates SSGDPBRD,
                                          SSGDPBWR, SSGDPRAT, and
                                          SSGDPWAT are valid
110   (6E) CHARACTER      2   SSGDPSID     SSID
112   (70) UNSIGNED       4   SSGDPITA     Irregular Track Accesses
116   (74) UNSIGNED       4   SSGDPITH     Irregular Track Access
                                          Hits
120   (78) UNSIGNED       4   SSGDPODC     Operation Delayed Due To
                                          Cache Space Constraints
124   (7C) UNSIGNED       4   SSGDPMSL     Milliseconds of lower
                                          interface I/O activity
                                          for the indicated device
                                          (Set to zeros for
                                          2107/1750)
128   (80) BITSTRING      1   SSGDPNVI     Non-volume information
                                          contained in the record
                                          Value Definition
                                          0 No Additional Information
                                          1 RAID Rank Information
                                          2 Extent Pool and Physical
                                            Storage Counter information
                                          3-FF Not Defined
129   (81) CHARACTER      1   SSGDPRVB     Not used
130   (82) CHARACTER      2   SSGDPSPI     Extent Pool ID
132   (84) UNSIGNED       1   SSGDPSTY     Extent Pool Type
                                           Hex Description
                                           04  FB 1 GB
                                           84  CKD 1 GB
133   (85) BITSTRING      1    SSGDPSFL    Extent Pool Flags
          1... ....             SSGDPSF0   Dynamic Segment
                                           Allocation
          .1.. ....             SSGDPSF1   Data Sharing
          ..1. ....             SSGDPSF2   Migrating/Migration Error
                                           State
          ...1 111.       *                Not Used. Set to zero.
          .... ...1             SSGDPSF7   Extent Sizes Valid
                                           When set to one, SSGDPSSZ
                                           contains the extent sizes in
                                           the Extent Pool.
134   (86) BITSTRING      1    SSGDPSSZ    Extent Sizes in Extent Pool
          1... ....             *          Reserved
          .1.. ....             SSGDP01GB  1 GB Extents
          ..11 111.             *          Reserved
          .... ...1             SSGDP16MB  16 MB Extents
135   (87) CHARACTER      1   *            Not used
136   (88) UNSIGNED       4   SSGDPPRO     Physical Storage Read
                                          Operations
140   (8C) UNSIGNED       4   SSGDPPWO     Physical Storage Write
                                          Operations
144   (90) UNSIGNED       4   SSGDPPBR     Physical Storage Bytes
                                          Read. See Increment Value
                                          in SSGDPDVD
148   (94) UNSIGNED       4   SSGDPPBW     Physical Storage Bytes
                                          Written. See Increment
                                          Value in SSGDPDVD
152   (98) UNSIGNED       4   SSGDPRMR     Record Mode Read
                                          Operations
156   (9C) UNSIGNED       4   SSGDPNTR     Number of tracks read
                                          from the concurrent Copy
                                          or XRC Sidefile
160   (A0) UNSIGNED       4   SSGDPNCW     Number of contaminating
                                          writes for a Concurrent Copy
                                          or XRC volume as a result of:
                                         - Update to a Concurrent Copy
                                           protected track.
                                         - Update to an XRC monitored
                                           track.
```

```
164     (A4) UNSIGNED         4    SSGDPTTS     Number of tracks or
                                                portion of tracks that
                                                were transferred to
                                                the secondary device of
                                                a PPRC pair
168     (A8) UNSIGNED         4    SSGDPNSA     NVS Space Allocations
172     (AC) UNSIGNED         4    SSGDPRRT     Physical Storage Read
                                                Response Time. See
                                                Increment Value in
                                                SSGDPDVD
176     (B0) UNSIGNED         4    SSGDPWRT     Physical Storage Write
                                                Response Time. See
                                                Increment Value in
                                                SSGDPDVD
180     (B4) UNSIGNED         4    SSGDPBRD     Bytes Read. See Increment
                                                Value in SSGDPDVD
184     (B8) UNSIGNED         4    SSGDPBWR     Bytes Written. See
                                                Increment Value in
                                                SSGDPDVD
188     (BC) UNSIGNED         4    SSGDPRAT     Read Accumulated Time.
                                                The accumulated response
                                                time for all read
                                                operations.
                                                See Increment Value in
                                                SSGDPDVD
192     (C0) UNSIGNED         4    SSGDPWAT     Write Accumulated Time.
                                                The accumulated response
                                                time for all write
                                                operations. See Increment
                                                Value in SSGDPDVD
196     (C4) UNSIGNED         4    SSGDPZHR     zHPF Read I/O Requests. This
                                                count is the number of Transport
                                                Mode read command chains. The
                                                Read I/O Request will also be
                                                counted in one of the following
                                                counters:
                                                 - SSGDPRNI, Search/Read Normal
                                                   I/O Requests
                                                 - SSGDPRSI, Search/Read
                                                   Sequential I/O Requests
                                                 - SSGDPRCF, Search/Read Cache
                                                   Fast Write I/O Requests
200     (C8) UNSIGNED         4    SSGDPZHW    HPF Write I/O Requests.  This
                                                count is the number of Transport
                                                Mode write command chains. The
                                                Write I/O Request will also be
                                                counted in one of the following
                                                counters:
                                                - SSGDPWNI, Write Normal I/O
                                                  Requests
                                                - SSGDPWSI, Write Sequential I/O
                                                  Requests
                                                - SSGDPCFW, Cache Fast Write I/O
                                                  Requests

204     (CC) UNSIGNED         4      SSGDPRRP  zHyperLink Read Pre-fetch I/O
                                                Requests. The number of
                                                zHyperLink chains which meet
                                                the following requirements:
                                                - The zHyperLink Read
                                                Operation failed due to Read
                                                Data Not Immediately
                                                Available (RC 0211) and a
                                                Pre-fetch request was made
                                                to obtain resources prior to
                                                a retry attempt.
208     (D0) UNSIGNED         4      SSGDPZHL  zHPF List Pre-fetch I/O
                                                Requests. This count is the
                                                number of command chains which
                                                meet the following requirements:
                                                - The Transport Mode operation
                                                  specified a non-zero Imbedded
                                                  Locate Record Count.
212     (D4) UNSIGNED         4      SSGDPZHH  zHPF List Pre-fetch I/O Requests
                                                Hits. This count is the number
                                                of common chains which meet the
                                                following requirements:
                                                - The Transport Mode operation
                                                  specified a non-zero Imbedded
                                       204          Locate Record Count.
                                                - The chain was completed
                                                  without requiring access to
```

```
                                                    any DDM.
     216     (D8) UNSIGNED         4      SSGDPGSF  Global Mirror Collisions
                                                    Sidefile Count. A GM collision
                                                    occurs when, during the sending
                                                    of data to the secondary to
                                                    create a consistency group, a
                                                    subsequent host update is
                                                    attempted before the modified
                                                    track has been transmitted to
                                                    the secondary volume. The
                                                    modified track will be moved
                                                    to the sidefile before allowing
                                                    a new host write. This counter
                                                    will be incremented by one when
                                                    a track is added to the
                                                    sidefile.
     220     (DC) UNSIGNED         4      SSGDPGSS  Global Mirror Collisions Send
                                                    Synchronous Count. When a
                                                    write collision occurs, the
                                                    modified track data which
                                                    belongs to the current
                                                    consistency group may be sent
                                                    to the remote control unit
                                                    before allowing the write. The
                                                    data may come from the
                                                    sidefile if it is full or from
                                                    cache if the collision
                                                    sidefile is not being
                                                    utilized.
     224     (E0) UNSIGNED         4      SSGDPSRR  Synch I/O Read
                                                    Requests
     228     (E4) UNSIGNED         4      SSGDPSRH  Synch I/O Read
                                                    Request Hits
     232     (E8) UNSIGNED         4      SSGDPSWR  Synch I/O Write
                                                    Requests
     236     (EC) UNSIGNED         4      SSGDPSWH  Synch I/O Write
                                                    Request Hits
     240     (F0) UNSIGNED         4      SSGDPRDT  Read Channel
                                                    Disconnect Time
     244     (F4) UNSIGNED         4      SSGDPWDT  Write Channel
                                                    Disconnect Time
     248     (F8) UNSIGNED         4      SSGDPRDC  Read Channel
                                                    Disconnect Count
     252     (FC) UNSIGNED         4      SSGDPWDC  Write Channel
                                                    Disconnect Count

  MAPPING FOR THE SUBSYSTEM GET OUTPUT BUFFER
  FOR SYNCH I/O LINK STATISTICS
  Output buffer is a queue, anchored by SSGSLOAR, of Synch
  I/O Link statistics.  Each queue element represents data
  read from a single storage subsystem box that describes sets
  of statistics.  The queue is a single threaded queue.  Each
  queue element needs to be freed by the caller of this
  interface.  Output buffer built for storage subsystem boxes
  that support Synch I/O Link statistics and are returned
  only if requested by caller (SSGSIOLP) on a request for
  performance data (SSGRPD) for requests with scope of
  subsystem (SSG1SS) or all subsystems (SSGALL).
============================================================
 OFFSET OFFSET
DECIMAL   HEX  TYPE     LENGTH NAME (DIM)   DESCRIPTION
======= ====== =======  ====== ==========   =================
      0   (0)  STRUCTURE     * SSGSLBFR
      0   (0)  CHARACTER    60  SSGSLHDR    Queue element
                                             header
      0   (0)  UNSIGNED      4   SSGSLQDAL  Length of this
                                             queue element,
                                             header plus data
      4   (4)  UNSIGNED      4   SSGSLQDAO  Offset to the
                                             start of the
                                             statistics sets
      8   (8)  ADDRESS       4   SSGSLFWDP  Pointer to next
                                             queue element,
                                             when zero end of
                                             queue
     12   (C) CHARACTER      6   SSGSLVOL   Volume serial of
                                             device that stats
                                             where read from
     18  (12) CHARACTER      2   SSGSLDEVN  Device number of
                                             device that Stats
                                             where read from
     20  (14) BITSTRING      1   SSGSLFLG
```

```
              11.. ....           *           RESERVED
              ..11 ....           SSGSL_SCHST SUBCHANNEL SET NUMBER
                                               SCHSET NUMBER 0 - 00
                                               SCHSET NUMBER 1 - 01
                                               SCHSET NUMBER 2 - 10
                                               SCHSET NUMBER 3 - 11
              .... 1111           *           RESERVED
     21  (15) CHARACTER      7   *           Reserved
     28  (1C) CHARACTER     32   SSGSILHDR   Synch I/O Link
                                              stats info
     28  (1C) UNSIGNED       2    SSGSLNSET  Number of synch
                                              I/O link
                                              statistic sets
                                              returned on the
                                              ESS starting
                                              from first
                                              interface ID
                                              SSGSLIID.
     30  (1E) UNSIGNED       2    SSGSLLSET  Size of each
                                              set.
     32  (20) CHARACTER      6    SSGSLCUT   Control unit
                                              type
     38  (26) CHARACTER      3    SSGSLCUM   Control unit
                                              model
     41  (29) CHARACTER     10    SSGSLSEQ   Control unit
                                              sequence number
     51  (33) CHARACTER      1    SSGSLVER   Version of
                                              Synch I/O Link
                                              stats
     52  (34) CHARACTER      1    SSGSLBRF   Byte Reporting Factor
                                               Value Description
                                               01  Low order bit
                                                   represents
                                                   128K bytes.
     53  (35) CHARACTER      1    SSGSLTRF   Time Reporting Factor
                                               Value Description
                                               01   Low order bit
                                                    represents 16
                                                    milliseconds.
     54  (36) CHARACTER      6    *          Not used
     60  (3C) CHARACTER      *   SSGSLDADA   Synch I/O Link
                                              stats read

 SSGSLDADA consists of Synch I/O Link statistics sets
 containing one or more table information entries for each
 Synch I/O interface. Each entry is 128 bytes defined in
 SSGSLSTA based on version number (X'01') in SSGSLVER.
============================================================
 OFFSET OFFSET
DECIMAL   HEX  TYPE     LENGTH NAME (DIM)   DESCRIPTION
======= ====== =======  ====== ==========   ================
     60   (3C) CHARACTER   128 SSGSLSTA(*)  Stats entry
     60   (3C) UNSIGNED      1  SSGSLTYP    Synch I/O Link
                                             Type
                                             00 Not Used
                                             01 Optical PCIe
     61   (3D) UNSIGNED      1  SSGSLSPD    Synch I/O Link
                                             Speed
                                             00 Not Used
                                             01 PCIe Gen 1
                                             02 PCIe Gen 2
                                             03 PCIe Gen 3
                                             04 PCIe Gen 4
     62   (3E) UNSIGNED      1  SSGSLWDH    Synch I/O Link
                                             Width
                                             This value is the
                                             number of PCIe
                                             lanes (1, 2, 4,
                                             8 or 16).
     63   (3F) UNSIGNED      1  SSGSLSTE    Synch I/O Link
                                             State
                                             00 Not Used
                                             01 Link Not
                                                Trained
                                             02 Link Handshake
                                                Incomplete
                                             03 Link Handshake
                                                Complete and
                                                Link
                                                Operational
                                             04 Link in
                                                Service Mode
```

```
                                          (i.e. Link
                                          Quiesced)
  64   (40) UNSIGNED      2  SSGSLIID    Interface id
                                          0 Port type (set
                                            to zero)
                                          1-2 Reserved
                                          3-7 Enclosure
                                          8-11 Adapter. Set
                                               to '1000'b.
                                          12-15 Port. Set
                                                to '0000'b
                                                for Link 0
                                                and to
                                                '0001'b for
                                                Link 1.
  66  (42) CHARACTER      2  *           Not used. Set to
                                          zero.
  68  (44) UNSIGNED       4  SSGSLCBR    Cache Bytes Read
                                          SSGSLBRF
                                          specifies
                                          reporting factor.
  72  (48) UNSIGNED       4  SSGSLCRO    Total Cache Read
                                          Operations
  76  (4C) UNSIGNED       4  SSGSLCRS    Successful Cache
                                          Read Operations
  80  (50) UNSIGNED       4  SSGSLCRT    Cache Read
                                          Accumulated Time
                                          SSGSLTRF
                                          specifies
                                          reporting factor.
  84  (54) UNSIGNED       4  SSGSLCBW    Cache Bytes
                                          Written
                                          SSGSLBRF
                                          specifies
                                          reporting factor.
  88  (58) UNSIGNED       4  SSGSLCWO    Total Cache Write
                                          Operations
  92  (5C) UNSIGNED       4  SSGSLCWS    Successful Cache
                                          Write Operations
  96  (60) UNSIGNED       4  SSGSLCWT    Cache Write
                                          Accumulated Time
                                          SSGSLTRF
                                          specifies
                                          reporting factor.
 100  (64) UNSIGNED       4  SSGSLNBW    NVS Bytes Written
                                          SSGSLBRF
                                          specifies
                                          reporting factor.
 104  (68) UNSIGNED       4  SSGSLNWO    Total NVS Write
                                          Operations
 108  (6C) UNSIGNED       4  SSGSLNWS    Successful Cache
                                          NVS Operations
 112  (70) UNSIGNED       4  SSGSLNWT    NVS Write
                                          Accumulated Time
                                          SSGSLTRF
                                          specifies
                                          reporting factor.
 116  (74) UNSIGNED       4  SSGSLDDR    I/O Mail Dispatch
                                          Delay, I/O
                                          Rejected
 120  (78) UNSIGNED       4  SSGSLDDD    I/O Mail Dispatch
                                          Delay, I/O
                                          Dropped
 124  (7C) UNSIGNED       4  SSGSLSDR    I/O Mail Scan
                                          Delay, I/O
                                          Rejected
 120  (80) UNSIGNED       4  SSGSLSDD    I/O Mail Scan
                                          Delay, I/O
                                          Dropped
 124  (84) UNSIGNED       4  SSGSLCRSAT  Cache Read Success
                                          Accumulated Time
 128  (88) UNSIGNED       4  SSGSLCWSAT  Cache Write Success
                                          Accumulated Time
 132  (92) UNSIGNED       4  SSGSLNWSAT  NVS Write Success
                                          Accumulated Time
 136  (96) CHARACTER     44  *           Not used. Set to zero.


MAPPING FOR THE SUBSYSTEM GET OUTPUT BUFFER
FOR SYNCH I/O LINK DIAGNOSTIC PARAMETERS
Output buffer is a queue, anchored by SSGSDOAR, of Synch
I/O Link Diagnostic Parameters.  Each queue element
```

```
  represents data read from a single storage subsystem box
  that describes sets of statistics.  The queue is a single
  threaded queue.  Each queue element needs to be freed by
  the caller of this interface.  Output buffer built for
  storage subsystem boxes that support Synch I/O Link
  Diagnostic Parameters and are returned only if requested by
  caller (SSGSILDP) on a request for performance data
  (SSGRPD) for requests with scope of subsystem (SSG1SS) or
  all subsystems (SSGALL).
============================================================
 OFFSET OFFSET
DECIMAL   HEX  TYPE     LENGTH NAME (DIM)   DESCRIPTION
======= ====== =======  ====== ==========   =================
      0   (0)  STRUCTURE     * SSGSDBFR
      0   (0)  CHARACTER    60  SSGSDHDR    Queue element
                                             header
      0   (0)  UNSIGNED      4   SSGSDQDAL  Length of this
                                             queue element,
                                             header plus data
      4   (4)  UNSIGNED      4   SSGSDQDAO  Offset to the
                                             start of the
                                             statistics sets
      8   (8)  ADDRESS       4   SSGSDFWDP  Pointer to next
                                             queue element,
                                             when zero end of
                                             queue
     12   (C) CHARACTER      6   SSGSDVOL   Volume serial of
                                             device that stats
                                             where read from
     18  (12) CHARACTER      2   SSGSDDEVN  Device number of
                                             device that Stats
                                             where read from
     20  (14) BITSTRING      1   SSGSDFLG
              11.. ....       *             RESERVED
              ..11 ....       SSGSD_SCHST   SUBCHANNEL SET NUMBER
                                             SCHSET NUMBER 0 - 00
                                             SCHSET NUMBER 1 - 01
                                             SCHSET NUMBER 2 - 10
                                             SCHSET NUMBER 3 - 11
              .... 1111       *             RESERVED
     21  (15) CHARACTER      7   *          Reserved
     28  (1C) CHARACTER     32   SSGSILHDR  Synch I/O Link
                                             stats info
     28  (1C) UNSIGNED       2    SSGSDNSET  Number of synch
                                             I/O link
                                             statistic sets
                                             returned on the
                                             ESS starting
                                             from first
                                             interface ID
                                             SSGSLIID.
     30  (1E) UNSIGNED       2    SSGSDLSET  Size of each
                                             set.
     32  (20) CHARACTER      6    SSGSDCUT   Control unit
                                             type
     38  (26) CHARACTER      3    SSGSDCUM   Control unit
                                             model
     41  (29) CHARACTER     10    SSGSDSEQ   Control unit
                                             sequence number
     51  (33) CHARACTER      1    SSGSDVER   Version of
                                             Synch I/O Lnk
                                             Diag Prm
     54  (34) CHARACTER      8    *          Not used
     60  (3C) CHARACTER      *   SSGSDDADA   Synch I/O Link
                                             Diag Prms read

 SSGSDDADA consists of Synch I/O Link Diagnostic Parameters
 sets containing one or more table information entries for
 each Synch I/O interface. Each entry is 192 bytes defined in
 SSGSDSTA based on version number (X'01') in SSGSDVER.
============================================================
 OFFSET OFFSET
DECIMAL   HEX  TYPE     LENGTH NAME (DIM)   DESCRIPTION
======= ====== =======  ====== ==========   =================
     60   (3C) CHARACTER   192 SSGSDSTA(*)  Stats entry
     60   (3C) UNSIGNED      1  SSGSDTYP    Synch I/O Link
                                             Type
                                             00 Not Used
                                             01 Optical PCIe
     61   (3D) UNSIGNED      1  SSGSDSPD    Synch I/O Link
                                             Speed
                                             00 Not Used
```

```
                                          01 PCIe Gen 1
                                          02 PCIe Gen 2
                                          03 PCIe Gen 3
                                          04 PCIe Gen 4
     62   (3E) UNSIGNED      1  SSGSDWDH    Synch I/O Link
                                          Width
                                          This value is the
                                          number of PCIe
                                          lanes (1, 2, 4,
                                          8 or 16).
     63   (3F) UNSIGNED      1  SSGSDSTE    Synch I/O Link
                                          State
                                          00 Not Used
                                          01 Link Not
                                             Trained
                                          02 Link Handshake
                                             Incomplete
                                          03 Link Handshake
                                             Complete and
                                             Link
                                             Operational
                                          04 Link in
                                             Service Mode
                                             (i.e. Link
                                             Quiesced)
     64   (40) UNSIGNED      2  SSGSDIID    Interface id
                                          0 Port type (set
                                            to zero)
                                          1-2 Reserved
                                          3-7 Enclosure
                                          8-11 Adapter. Set
                                               to '1000'b.
                                          12-15 Port. Set
                                                to '0000'b
                                                for Link 0
                                                and to
                                                '0001'b for
                                                Link 1.
     66  (42) CHARACTER      2  *           Not used. Set to
                                          zero.
     68  (44) UNSIGNED       4  SSGSDPRE    Port Receiver
                                          Error Count
     72  (48) UNSIGNED       4  SSGSDDLLP   Port Bad Data
                                          Link Layer Packet
                                          (DLLP) Count
     76  (4C) UNSIGNED       4  SSGSDTLP    Port Bad
                                          Transaction Layer
                                          Packet (TLP)
                                          Count

 The format of the next four bytes is repeated the number of
 lanes reported in SSGSDWDH (Synch I/O Link Width) minus 1.
 The unused bytes are set to zero.
=============================================================
 OFFSET OFFSET
DECIMAL   HEX  TYPE     LENGTH NAME (DIM)   DESCRIPTION
======= ====== =======  ====== ==========   ================
     80  (50) CHARACTER     16  SSGSDPLL    Power Levels for
                                          Lane 1, 2, 4, 8,
                                          or 16.
     80  (50) UNSIGNED       2   SSGSDTXP   Tx Power: Lane 1,
                                          2, 4, 8, or 16
                                          This value
                                          contains the
                                          measured coupled
                                          TX output power
                                          in units of
                                          0.1 uW (range
                                          0-6.5mW).
     82  (52) UNSIGNED       2   SSGSDRXP   Rx Power: Lane 1,
                                          2, 4, 8, or 16
                                          This value
                                          contains the
                                          measured coupled
                                          Rx output power
                                          in units of
                                          0.1 uW (range
                                          0-6.5mW).
    144  (90) CHARACTER     64  SSGSDCSE    Chip Specific
                                          Error Counts
                                          These 64 bytes
                                          are used by
```

```
                                          engineering only.
                                          The counts are
                                          specific to the
                                          chip.
      208  (D0) CHARACTER     44  *          Not used. Set to zero.
      224  (E0) CHARACTER     32  SSGDPRVY    Not used - zeros

        RETURN CODES FROM IDCSS01
================================================================================

  LEN  TYPE           VALUE          NAME            DESCRIPTION
=====  =========  ================  ============== ==============================
    2  NUMB HEX        0000         SS01CC00        SUCCESSFUL COMPLETION
    2  NUMB HEX        0004         SS01CC04        GETMAIN FAILED
    2  NUMB HEX        0008         SS01CC08        I/O ERROR OCCURRED
    2  NUMB HEX        000C         SS01CC12        VOLUME NOT FOUND
    2  NUMB HEX        0010         SS01CC16        ESTAE FAILED
    2  NUMB HEX        0014         SS01CC20        LRA INSTRUCTION FAILED
    2  NUMB HEX        0018         SS01CC24        BUFFER NOT LARGE ENOUGH
    2  NUMB HEX        001C         SS01CC28        FILE NOT IN TIOT
    2  NUMB HEX        0020         SS01CC32        VOLUME NOT FOUND ALL
    2  NUMB HEX        0024         SS01CC36        NO PATH(S) TO SD(S)
    2  NUMB HEX        0028         SS01CC40        NO ONLINE PATH FROM IOS
    2  NUMB HEX        002C         SS01CC44        REQUEST NOT SUPPORTED
    2  NUMB HEX        0030         SS01CC48        WARNING MSG ISSUED
    2  NUMB HEX        0034         SS01CC52        ERROR MSG ISSUED
    2  NUMB HEX        0038         SS01CC56        SERIOUS ERROR MSG
    2  NUMB HEX        003C         SS01CC60        I/O timeout occured
    2  NUMB HEX        0040         SS01CC64        Reserved
    2  NUMB HEX        0044         SS01CC68        RESERVED
    2  NUMB HEX        0046         SS01CC70        IOSCDR ERROR
    2  NUMB HEX        0047         SS01CC71        UCBINFO ERROR
    2  NUMB HEX        0048         SS01CC72        RESERVED
    2  NUMB HEX        0049         SS01CC73        MISSING CDR INFOR
    2  NUMB HEX        004A         SS01CC74        ERR DURING CDR INFOR
                                                    GATHERING.
    2  NUMB HEX        0063         SS01CC99        RESERVED

        REASON CODES FROM ESS (BOX) PERFORMANCE STATISTICS
         (SSGLPRET, SSGRSRET, SSGSGRET, SSGSLRET, SSGSDRET).
================================================================================
  LEN  TYPE           VALUE          NAME            DESCRIPTION
=====  =========  ================  ============== ==============================
    1  NUMB HEX          01         SS01LP01        Stats requested, all stats
                                                    returned on supporting
                                                    subsystem boxes
    1  NUMB HEX          02         SS01LP02        Stats not requested by the
                                                    caller
    1  NUMB HEX          03         SS01LP03        Stats requested, but no
                                                    supporting subsystem boxes
                                                    found to support them
    1  NUMB HEX          04         SS01LP04        Stats requested, but stats
                                                    from one or more supporting
                                                    subsystem boxes could not be
                                                    read
```

Appendix A. Control Blocks

The following control blocks are described here:

- "Data Extent Block (DEB) Fields" on page 431 for EXCP and EXCPVR.
- "Data Facilities Area (DFA) Fields" on page 438

Data Extent Block (DEB) Fields

The data extent block (DEB) fields shown here should *only* be used with EXCP and EXCPVR (see *z/OS DFSMSdfp Diagnosis*53 for all fields in the DEB).

Common Name:
Data Extent Block

Macro ID:
IEZDEB

DSECT Name:

- DEB (DSECT card precedes AVT section)
- DEBBASIC should be used for USING basic section.
- DEBDASD (DSECT name for direct access section)
- DEBACSMD (DSECT name for access method sections)
- DEBSUBNM (DSECT name for subroutine name section)
- DEBXTN (DSECT name for DEB extension)

Owning Component:
Data Management, subcomponent OPEN/CLOSE/EOV

Eye-Catcher ID:
None

Subpool and Key:
230 and key 5

Size:
Variable (device and access method dependent sections)

Created by:
OPEN

Pointed to by:
DCBDEBAD field of the DCB data area. DEBDEBAD field of the DEB data area (next DEB on the chain)

Serialization:
LOCAL lock serializes the placing of a DEB on the TCB DEB chain and in the DEB table. OPEN/CLOSE/EOV processing is serialized by local lock and DEBCHK.

Function:
The DEB is a repository for information from the DCB, data set label, device and user. Each DEB is associated with a DCB, and the two point to each other. It contains information about the physical characteristics of the data set, and other information used by the control program. The DEB also serves to control the user program's access to the data set and its device.

Table 110. Partial Listing of DEB Fields

Offset	Type	Length	Name	Description
Appendage vector table section of the DEB pointed to by DEBAPPAD. It is not necessarily contiguous with the DEB or DEB prefix.				
0(0)		20	DEBAVT(0)	Appendage vector table

Table 110. Partial Listing of DEB Fields (continued)

Offset	Type	Length	Name	Description
0(0)	ADDRESS	4	DEBEOEA(0)	Address of end-of-extent appendage routine
0(0)	BITSTRING	1	DEBEOEAB	Flag byte
	 xxxx		DEBEOENP	X'0F'– number of 2K pages to be fixed for the end–of–extent appendage
1(1)	ADDRESS	3	DEBEOEAD	Address of end-of-extent appendage routine
4(4)	ADDRESS	4	DEBSIOA(0)	Address of start I/O appendage routine
4(4)	BITSTRING	1	DEBSIOAB	Flag byte
	1.......		DEBPGFX	X'80' Address in DEBSIOAD can be used to determine the entry point to the page fix (PGFX) appendage routine by adding 4 to the address in DEBSIOAD
	.1......		DEBSIOX	X'40'– if zero, do not enter SIO appendage when ERP IS active. If one, enter SIO appendage even when ERP is active.
	..1.....		DEBIOVR	X'20'– If one, EXCPVR request is valid. If zero, EXCPVR request is invalid and will not be executed. Currently has no effect.
	...1....		DEBFIX	X'10' Indication that DEB has been fixed
	 xxxx		DEBSIONP	X'0F'– number of 2K pages to be fixed for the SIO appendage
5(5)	ADDRESS	3	DEBSIOAD	Address of start I/O appendage routine
8(8)	ADDRESS	4	DEBPCIA(0)	Address of PCI appendage routine
8(8)	BITSTRING	1	DEBPCIAB	Flag byte
	 xxxx		DEBPCINP	X'0F'–number of 2K pages to be fixed for the PCI appendage
9(9)	ADDRESS	3	DEBPCIAD	Address of program controlled interruption (PCI) appendage routine
12(C)	ADDRESS	4	DEBCEA(0)	Address of channel end appendage routine
12(C)	BITSTRING	1	DEBCEAB	Flag byte
	1.......		DEBESMVR	X'80',c'x' validity check for EXCPVR caller
	 xxxx		DEBCENP	X'0F'– number of 2K pages to be fixed for the channel–end appendage
13(D)	ADDRESS	3	DEBCEAD	Address of channel end appendage routine
16(10)	ADDRESS	4	DEBXCEA(0)	Address of abnormal end appendage routine
16(10)	BITSTRING	1	DEBXCEAB	Flag byte
	 xxxx		DEBXCENP	X'0F'– number of 2K pages to be fixed for the abnormal–end appendage
17(11)	ADDRESS	3	DEBXCEAD	Address of abnormal end appendage routine
	X''		DEBAVTE	"*" End of appendage vector table
DEB PREFIX TABLE. Addressable as negative offset from DEB basic section. Not necessarily contiguous with DEB appendage vector table.				
-16(10)		16	DEBPREFX(0)	DEB prefix table
-16(10)	BITSTRING	1	DEBWKARA	O/C/E work area (direct access)
-15(F)	BITSTRING	7	DEBDSCBA	DSCB address (BBCCHHR) used by O/C/E (direct access)
-8(8)	ADDRESS	4	DEBXTNP(0)	Pointer to DEB extension
-4(4)	BITSTRING	1	DEBLNGTH	Length of DEB in double words. If it exceeds 2040 bytes (X'FF' doublewords), the excess bits are ignored.
-3(3)	CHARACTER	1	DEBAMTYP	Access method type
	X'0'		DEBAMNON	"0" Access method type not known
	X'1'		DEBAMVSM	"1" VSAM access method type
	X'2'		DEBAMXCP	"2" EXCP access method type
	X'4'			Reserved
	X'8'		DEBAMGAM	"8" Graphics access method type
	X'10'		DEBAMTAM	"16" BTAM access method type

Table 110. Partial Listing of DEB Fields (continued)

Offset	Type	Length	Name	Description
	X'20'		DEBAMBPM	"32" BPAM access method type
	X'20'		DEBAMSAM	"32" Sequential access method type
	X'40'		DEBAMBDM	"64" Direct access method type
	X'81'		DEBAMSUB	"129" subsystem access method type
	X'82'		DEBAMVTM	"130" VTAM® access method type
	X'84'			Reserved
-2(2)	UNSIGNED	2	DEBTBLOF	For system use
	X'24'		DEBPREFE	"*" End of DEB prefix table
DEB basic section				
0(0)			DEBBASIC	"*"
0(0)	ADDRESS	4	DEBTCBAD(0)	Address of TCB for this DEB
0(0)	BITSTRING	1	DEBNMSUB	Number of subroutines loaded by OPEN executor routines and identified in the subroutine name section (DEBSUBID)
1(1)	ADDRESS	3	DEBTCBB	Address of TCB for this DEB
4(4)	ADDRESS	4	DEBDEBAD(0)	Address of the next DEB in the same task
4(4)	BITSTRING	1	DEBAMLNG	Number of bytes in the access method dependent section. For BDAM this field contains the length expressed in number of words.
5(5)	ADDRESS	3	DEBDEBB	Address of the next DEB in the same task
8(8)	ADDRESS	4	DEBIRBAD(0)	IRB storage address used for appendage asynchronous exits
8(8)	BITSTRING	1	DEBOFLGS	Data set status flags
	xx.		DEBDISP	X'C0'– data set disposition flags bit setting disposition
	01.		DEBDSOLD	X'40' old data set
	10		DEBDSMOD	X'80' mod data set
	11.		DEBDSNEW	X'C0' new data set
	. .1.		DEBEOF	X'20'– end of file (EOF) encountered (tape input) format 1 DSCB bit 93.0 indicates that the current volume is the last volume of the data set (DASD input)
	. . .1		DEBRLSE	X'10'– release unused external storage (DASD) emulator tape with second generation format. Tape might contain blocks shorter than 12 characters. (tape)
	. . . . .1. .		DEBSPLIT	X'04'–7–track emulator tape with possible mixed parity records (tape)
	. . . . . .1.		DEBLABEL	X'02'– nonstandard labels
	1		DEBRERR	X'01'– use reduced error recovery procedure (tape) concatenated partitioned organization data sets processed using BPAM (DASD)
9(9)	ADDRESS	3	DEBIRBB	IRB storage address used for appendage asynchronous exits
12(C)	BITSTRING	1	DEBOPATB	Flags indicating both the method of I/O processing and the disposition that is to be performed when an end of volume (EOV) condition occurs
	1.		DEBABEND	X'80'– set by abend indicating a sysabend or sysudump data set
	.0.		DEBZERO	X'40'– always zero
	. .xx		DEBPOSIT	X'30'– data set positioning flags bit setting positioning
	. . 01		DEBRERED	X'10' reread
	. .11		DEBLEAVE	X'30' leave
	 xxxx		DEBACCS	X'0F'– type of I/O accessing being done bit setting accessing
	 0000		DEBINPUT	X'0' INPUT
	 1111		DEBOUTPT	X'F' OUTPUT or EXTEND
	 0011		DEBINOUT	X'3' INOUT

Table 110. Partial Listing of DEB Fields (continued)

Offset	Type	Length	Name	Description
	 0111		DEBOUTIN	X'7' OUTIN or OUTINX
	 0001		DEBRDBCK	X'1' RDBACK
	 0100		DEBUPDAT	X'4' UPDAT
13(D)	BITSTRING	1	DEBQSCNT	PURGE (SVC 16) – Quiesce count. Number of devices executing user's channel programs, as shown by bits 5 and 6 of UCBFL1 fields.
14(E)	BITSTRING	1	DEBFLGS1	Flag field
	1.....		DEBPWCKD	X'80'– password was supplied during OPEN. EOV will not request a password for each additional volume of a multivolume data set.
	.1.....		DEBEOFDF	X'40'– set by EOV to inform close that an end of file has been encountered and, therefore, deferred user label processing is allowed.
	...1....		DEBEXCPA	X'10'– EXCP(VR) is authorized for this DEB
	.....1..		DEBF1CEV	X'04'–not valid with an EXCP DCB.
	......1.		DEBAPFIN	X'02'– if on, authorized programs can be loaded
	1		DEBXTNIN	X'01'– if one, DEB extension exists
15(F)	BITSTRING	1	DEBFLGS2	Flag field two
				The following two flag bits are used by O/C/EOV to maintain a uniform recording mode (compaction or non-compaction) on tape data sets that span over more than one volume.
	....1...		DEBDSCMP	X'08' tape data set compaction mode
	.....1..		DEBDSNCP	X'04' tape data set non-compaction mode
	......1.		DEB31UCB	UCB address fields are four bytes. Use DEBSUCBA, DEBUCBAD, DEBSDVMX or DEBDVMOD31 instead of DEBSUCBB, DEBUCBA, DEBSDVM or DEBDVMOD.
16(10)	ADDRESS	4	DEBUSRPG(0)	Address of purged I/O restore list (PIRL)
16(10)	BITSTRING	1	DEBNMEXT	Number of extent descriptions starting at DEBBASND. One extent per unit for extended format or PDSE data sets.
17(11)	ADDRESS	3	DEBUSRPB	Address of purged I/O restore list (PIRL)
20(14)	ADDRESS	4	DEBRRQ(0)	Pointer to related request queue
20(14)	BITSTRING	1	DEBPRIOR	Priority of the task owning DEB
24(18)	ADDRESS	4	DEBDCBAD(0)	Address of DCB or ACB associated with this DEB
24(18)	BITSTRING	1	DEBPROTG(0)	Task protection key in high order 4 bits
24(18)	BITSTRING	1	DEBDEBID	A hex F in low–order 4 bits to identify this block as a DEB
25(19)	ADDRESS	3	DEBDCBB	Address of DCB or ACB associated with this DEB
28(1C)	ADDRESS	4	DEBAPPAD(0)	Address of the I/O appendage vector table
28(1C)	ADDRESS	1	DEBEXSCL	This field is used to determine the size of the device dependent section. Two to this power gives the length of the device dependent section at DEBBASND. Extent scale – 4 (16 bytes) for direct access device and 3525 card punch with device–associated data set support and 2 (4 bytes) for nondirect access device and communication device.
29(1D)	ADDRESS	3	DEBAPPB	Address of the I/O appendage vector table
32(20)			DEBBASND	End of basic section
After the basic section is one or more device-dependent sections. DEBEXSCL contains its length. DEBNMEXT contains the number of device-dependent sections. All of the device-dependent sections are for the same device class.				
Unit Record, Magnetic Tape, Telecommunications Devices Section				
32(20)			DEBDDS1	"*"
32(20)	ADDRESS	4	DEBSUCBA(0)	See the DEBUCBAD field.

Table 110. Partial Listing of DEB Fields (continued)

Offset	Type	Length	Name	Description
32(20)	BITSTRING	1	DEBSDVM	Device modifier. For magnetic tape, MODESET operation code. For unit record, not defined. Valid only if DEB31UCB is off.
	1101 0011		DEBMTDN4	X'D3' 9–track MODESET CCW CODE DENSITY=6250BPI
	1100 0011		DEBMTDN3	X'C3' 9–track MODESET CCW CODE DENSITY=1600BPI
	1100 1011		DEBMTDN2	X'CB' 9–track MODESET CCW CODE DENSITY= 800BPI 7–track TAPE MODESET skeleton codes (must be completed with parity, translation and/or conversion)
	0000 0011		DEBM7DN0	X'03' 7–track MODESET SKELETON DENSITY=200BPI
	0100 0011		DEBM7DN1	X'43' 7–track MODESET SKELETON DENSITY=556BPI
	1000 0011		DEBM7DN2	X'83' 7–track MODESET SKELETON DENSITY=800BPI 3480 tape operation code
	1100 0011		DEBMSTWI	X'C3' 3480 set tape write immediate ccw code. Tape mode set function byte
	1.		DEBMTRF0	X'80' tape recording format bit 0
	.1.		DEBMTRF1	X'40' tape recording format bit 1
	. .1.		DEBMTWI	X'20' tape write immediate (non-buffered write)
	. . .1		DEBMINHS	X'10' inhibit supervisor commands
	. . . . 1. . .		DEBMCOMP	X'08' compacted recording mode
	. . . . 1. . .		DEBCMPAC	"DEBMCOMP" COMPACTED RECORDING MODE
	. . . . . .1.		DEBM3424	X'02' 3424 mode set flag
	1		DEBMINHE	X'01' inhibit control unit ERP
	X'C2'		DEBM6250	"DEBMTRF0+ DEBMTRF1+ DEBM3424"SET 3424 DENSITY=6250BPI
	X'42'		DEBM1600	"DEBMTRF1+ DEBM3424" SET 3424 DENSITY=1600BPI
33(21)	ADDRESS	3	DEBSUCBB	Address of a UCB associated with a given data set
36(24)		0	DEBDEVED(0)	End of common tape and unit record fields if DEB31UCB is off.
Next four bytes present only if DEB31UCB is on				
36(24)	BITS	1	DEBSDVMX	Device modifier. For magnetic tape, modeset operation code or modeset function byte. For unit record reserved. Present only if DEB31UCB is on.
37(25)	CHARACTER	3		Reserved
40(28)		0	DEBDVEDX	End of section if DEB31UCB is on.
The following fields are present only for the 3525 with device–associated data set support				
36(24)	ADDRESS	4	DEBRDCB(0)	
36(24)	BITSTRING	1	DEBRSV06	
37(25)	ADDRESS	3	DEBRDCBA	
40(28)	ADDRESS	4	DEBPDCB(0)	
40(28)	BITSTRING	1	DEBRSV07	
41(29)	ADDRESS	3	DEBPDCBA	
44(2C)	ADDRESS	4	DEBWDCB(0)	
44(2C)	BITSTRING	1	DEBRSV08	
45(2D)	ADDRESS	3	DEBWDCBA	
			DEBASDSE	Address of DCB for the read associated data set
Direct–access storage device section it follows the basic section. There is one of these sections for each extent, except for a pdse or extended format data set, in which case there is one section per device.				

Table 110. Partial Listing of DEB Fields (continued)

Offset	Type	Length	Name	Description
0(0)	ADDRESS	4	DEBUCBAD	Address of a UCB associated with this extent. The following applies to DEBSUCBA, DEBSUCBB, DEBUCBAD and DEBUCBA: If the actual UCB is above the 16 MB line and the dynamic allocation nocapture option is not in effect, allocation normally captures the UCB to create a 24-bit address. When using EXCP and you specify the nocapture option of dynamic allocation but not the LOC=ANY option on the DCBE, then OPEN or EOV captures the UCB until a later EOV or close. In these cases the high order byte of this word contains the device modifier byte. If you specify nocapture on the dynamic allocation and the actual DASD or tape UCB address is above the line and the DCB is for BSAM, BPAM, QSAM or EXCP and the DCBE has specified LOC=ANY, then the system does not capture the UCB. In that case OPEN turns on the DEB31UCB bit to signify the 31-bit UCB address field is valid and that the device modifier byte is in DEBSDVMX or DEBDVMOD31. It may remain on for subsequent volumes even though they have actual 24-bit addresses in a four-byte field.
0(0)	BITSTRING	1	DEBDVMOD	Device modifier. File mask. Valid only if DEB31UCB is off.
1(1)	ADDRESS	3	DEBUCBA	Address of UCB for this extent. Valid only if DEB31UCB is off.
4(4)	BINARY	1	DEBDVMOD31	Reserved if DEB31UCB is off. File mask (device modifier) if DEB31UCB is on.
5(5)	BINARY	1	DEBNMTRKHI	High order byte of number of tracks in extent. Low order two bytes are in DEBNMTRK. DEBNMTRKHI combined with DEBNMTRK gives total number of tracks in this extent. DEBNMTRKHI is 0 for a basic format data set and may be non-zero for a large format data set.
6(6)	BINARY	2	DEBSTRCC	Low order 16 bits of cylinder number of start of extent
8(8)	BINARY	2	DEBSTRHH	High order 12 bits of cylinder number and four-bit track number of start of extent
10(A)	BINARY	2	DEBENDCC	Low order 16 bits of cylinder number of end of extent. Not set for a PDSE
12(C)	BINARY	2	DEBENDHH	High order 12 bits of cylinder number and four-bit track number of end of extent. For PDSE this field is reserved. For an extended format data set this field contains the track number of the format-1 dscb address.
14(E)	BINARY	2	DEBNMTRK	Number of tracks allocated to a given extent. For a pdse this field is set to one(X'0001'). For an extended format sequential data set the first byte contains the record number of the format-1 dscb address, and the second byte is zero.
			DEBDASDE	"*" end of DASD device section
Access method section. DEBAMLNG contains its length. The following is for EXCP, BSAM and QSAM.				
0(0)	BINARY	2	DEBVOLSQ	Volume sequence number for multivolume sequential data sets
0(0)	BITSTRING	1	DEBVOLBT	First byte of debvolsq. Reserved for system use.
1(1)	SIGNED	1	DEBVLSEQ	For direct access, sequence number of the volume of the data set relative to the first volume of the data set. For tape, sequence number of the volume of the data set relative to the first volume processed.
2(2)	BINARY	2	DEBVOLNM	Total number of volumes in a multivolume sequential data set.
4(4)	BINARY	8	DEBDSNM	Member name. This field appears only when an output data set has been opened for a member name and the dscb specifies a partitioned data set.
4(4)	ADDRESS	4	DEBUTSAA	Address of the user totaling save area
4(4)	BITSTRING	1	DEBRSV13	For system use
5(5)	ADDRESS	3	DEBUTSAB	Address of the user totaling save area
8(8)	BITSTRING	4	DEBRSV14	For system use (if user totaling was specified)
12(C)	SIGNED	2	DEBBLKSI	Maximum block size
14(E)	SIGNED	2	DEBLRECL	Logical record length
BPAM Dependent Section				
0(0)	BINARY	1	DEBEXTNM	For a partitioned data set opened for input, each one–byte field contains the extent number of the first extent entry for each data set except the first, if two or more data sets are concatenated. The number of bytes in the field is equal to one less than the number of data sets concatenated.

Table 110. Partial Listing of DEB Fields (continued)

Offset	Type	Length	Name	Description
0(0)	CHARACTER	8	DEBDSNAM	For a partitioned data set opened for output for a member name, this field is the member name.
BDAM Dependent Section For Fixed Length Records With The Option Of Relative Block Addressing				
0(0)	BINARY	4	DEBDBLK	One four byte field for each extent described in the device dependent section
0(0)	ADDRESS	1	DEBDBPT	Number of blocks per track
1(1)	BINARY	3	DEBDBPE	Number of blocks per extent
Subroutine name section. DEBNMSUB contains how many names this section contains. It follows the access method dependent section or the device dependent section if there is no access method section.				
0(0)	CHARACTER	2	DEBSUBID	Subroutine identification. Each access method subroutine, appendage subroutine, and irb routine will have a unique eight–byte name. The low–order two bytes of each routine name will be in this field if the subroutine is loaded by the OPEN routines.
DEB Extension Pointed To By DEBXTNP In The DEB Prefix				
0(0)	SIGNED	2	DEBXLNGH	Length of DEB extension
2(2)	BITSTRING	1	DEBXFLG1	Flag byte—for system use
3(3)	BITSTRING	1	DEBXFLG2	Flag byte
	.1.		DEBBYP	X'40'– when on EXCP scan routine will set iosbyp on and bypass building a prefix
	. .1.		DEBCHCMP	X'20'– when on EXCP scan routine will set ioschcmp on and bypass building a prefix
4(4)	ADDRESS	4	DEBXDSAB	Pointer to dsab
8(8)	BITSTRING	4		For system use
12(C)	ADDRESS	4	DEBXDBPR	Pointer to DEB
16(10)	CHARACTER	24		For system use
40(28)	BITSTRING	8	DEBXDEF(0)	Define extent data area
40(28)	BITSTRING	1	DEBDEFG1	Flag byte
	1.		DEBNSHED	X'80' no seek head permitted
	.1.		DEBXVDEF	X'40' DEB def ext data parms valid. Must be on for dx data to be used.
41(29)	BITSTRING	1	DEBGATTR	Global attributes
	xx.		DEBECKD	X'C0' extent definition 2 b'00.....' define extent operates as defined for fixed block arch b'11.....' define extent operates as defined for ckd ext-cchh extents
	1.		DEBGAEX1	X'80' extent definition 1
	.1.		DEBGAEX2	X'40' extent definition 2
	. .1.		DEBSTRTP	X'20' data storage type-norm/temp b'..1.....' normal data storage b'..0.....' temporary data storage
	. . .x xx . .		DEBGA345	X'1c' attributes bits 3,4,5 b'...000..' normal cache access b'...001..' bypass cache load b'...010..' inhibit cache load b'...011..' sequential access b'...100..' reserved b'...101..' reserved b'...111..' reserved b'.....xx' reserved
	. . .1		DEBGA1	X'10' attribute 1
	. . . . 1 . . .		DEBGA2	X'08' attribute 2
	. . . . .1 . .		DEBGA3	X'04' attribute 3 b'...000..' normal cache access b'...001..' bypass cache load b'...010..' inhibit cache load b'...011..' sequential access b'...100..' sequential staging mode'...101..' record access mode b'...111..' reserved b'.....xx' reserved
	. . . 0 00 . .		DEBNCACH	X'00' normal cache access
	. . . 0 01 . .		DEBBCACH	X'04' bypass cache load
	. . . 0 10 . .		DEBICACH	X'08' inhibit cache load
	. . . 0 11 . .		DEBSCACH	X'0c' sequential access
	. . . . . .1.		DEBNRD	X'02' non-retentive data access

Table 110. Partial Listing of DEB Fields (continued)

Offset	Type	Length	Name	Description
	1		DEBINHFW	X'01' inhibit fast write b'.....10' use non-retentive data b'.....11' inhibit DASD fast write
42(2A)	BITSTRING	2	DEBBLKSZ	Blocksize in bytes if less than or equal to 32760
44(2C)	CHARACTER	4	DEBEXTOK	Zero,extent locator or token
44(2C)	CHARACTER	2	DEBNRDID	Subsystem function id: non-retentive data id or concurrent copy / xrc session id
46(2E)	CHARACTER	2		
48(30)	CHARACTER	4	DEBIOPID	For system use
52(34)	CHARACTER	4	DEBBLKID	Block id value used to calculate number of blocks in this tape volume for extended block count checking

Data Facilities Area (DFA) Fields

Except for DFADFVAD, all of the fields in the DFA are part of the intended programming interface.

Common Name:
Data Facilities Area

Macro ID:
IHADFA

DSECT Name:
DFA

Eye-Catcher ID:
DFAACRON

Subpool and Key:
Nucleus resident and Key 0; Resident below 16MB

Size:
112 bytes

Created by:
Assembled into nucleus

Pointed to by:
CVTDFA field of the CVT

Serialization:
N/A

Function:
Maps the Data Facilities Area, which contains information that applies to DFSMS.

Table 111. DFA Fields

Offset	Type/Value	Length	Name	Description
0(0)	UNSIGNED	2	DFALEN	LENGTH OF THIS TABLE
Version, release, and modification level information for DFP. The first three digits of DFAREL represents the last level of DFP(X'332'). The fourth digit of DFAREL being nonzero indicates the level of THIS product is later than the indicated level.				
2(2)	BITSTRING	2	DFAREL	FOUR DIGITS = VERSION, RELEASE, MOD, X
The following are the feature bytes. When a bit is on, it means either that the current release supports the feature or that this instance of the system supports the feature. If the software supports a feature but it cannot be used, perhaps because corequisite software or hardware is not available, then the feature bit is off.				
4(4)	SIGNED	4	DFAFEATS(0)	ALL FEATURES BYTES–USED BY CS INSTR
4(4)	BITSTRING	1	DFAFEAT1	FEATURES BYTE 1
	1...		DFAXA	"X'80'" MVS/XA (COPY OF CVTMVSE IN CVT)
	.1..		DFALSR	Multiple VSAM LSR pools supported (MVS/XA DFP 1.1.0 in 1983)

Table 111. DFA Fields (continued)

Offset	Type/Value	Length	Name	Description
	..1.		DFAEOS	DASDM erase on scratch supported (MVS/XA DFP 2.1.0 in 1985)
	...1		DFAXRF	Extended recovery facility (MVS/XA DFP 2.1.0)
	.... 1...		DFAEXPCI	EXPORT by control interval (MVS/XA DFP 2.1.0)
	.... .1..		DFAEOSIC	ERASE on scratch for ICF (MVS/XA DFP 2.1.0)
	.... ..1.		DFASMS	System managed storage (SMS) (MVS/DFP 3.1.0 in 1989)
	1		DFAPDSE	"X'01'" PDSE SUPPORT AVAILABLE ON THE SYSTEM SET WHEN DFP LEVEL IS 3.2.0 OR GREATER AND AN APPROPRIATE LEVEL OF SP EXISTS.
	1		DFAIPDS	"DFAPDSE" IPDS IS OLD NAME FOR PDSE
5(5)	BITSTRING	1	DFAFEAT2	FEATURES BYTE 2
	1...		DFADLS	"X'80'" RESERVED
	.1..		DFAPML	"X'40'" RESERVED
	..1.		DFAFMS	"X'20'" FILE MANAGEMENT SERVICES SUPPORTED
	...1		DFACMPAC	"X'10'" INSTALLATION DEFAULT FOR COMPACTION
	.... 1...		DFABPBLD	"X'08'" BYPASS CHANNEL PROGRAM PREFIX BUILD
	.... .1..		DFASSF	"X'04'" SSF SERVICES ARE AVAILABLE
	.... ..1.		DFAMMEXT	"X'02'" MMS SUPPORTS XTIOT
	1		DFAINDEF	"X'01'" COMPACTION DEFAULT EXPLICITLY SET BY INSTALLATION
6(6)	BITSTRING	1	DFAFEAT3	FEATURES BYTE 3
	1...		DFAVOLSN	"X'80'" VOLSER EXTRACTED FROM SENSE INFO ACCEPTABLE BY THE INSTALLATION
	.1..		DFASAMEX	EXTENDED FORMAT SEQUENTIAL DATA SETS SUPPORTED
	.1..		DFASMSEX	ALIAS FOR DFASAMEX
	..1.		DFAKSDEX	EXTENDED FORMAT KSDS SUPPORTED
	...1		DFACMPCT	DFSMS ACCESS METHOD COMPRESSION SUPPORTED. BIT SET BY SMS SUB-SYSTEM INITIALIZATION.
	.... 1...		DFARLSJ3	THE SMSVSAM SERVER HAS SUCCESSFULLY INITIALIZED ON THIS SYSTEM. THIS BIT IS USED BY SMS SCHEDULING. ONCE ON, THIS BIT REMAINS ON FOR THE LIFE OF THE IPL. THIS BIT DOES NOT INDICATE THAT THE SMSVSAM SERVER IS CURRENTLY OPERATIONAL.
	.... .1..		DFARECAL	DATA SET RECALL CAPABILITY VIA® THE ARCHRCAL MACRO IS AVAILABLE.
	.... ..1.		DFADEEXT	DESERV EXIT FUNCTION IS AVAILABLE
	1		DFADLL	DFSMS DLL SUPPORT IS AVAILABLE
7(7)	BITSTRING	1	DFAFEAT4	FEATURES BYTE 4
	1...		DFAFDAT	RESERVED
	.1..		DFANSRV	DFP NIP SERVICES CAN BE INVOKED VIA IGGSSRV MACRO
	..1.		DFADYNL	DYNAMIC LINKLIST IS SUPPORTED
	..1.		DFACIR2	THE CATALOG INFORMATION ROUTINE, IKJEHCIR, SUPPORTS A FORMAT 2 WORK AREA, I.E., FULL WORD LENGTH FIELDS
	.... 1...		DFADYLPA	DFSMS SUPPORT FOR DYNAMIC LPA IS AVAILABLE.
	.... .1..		DFAFORK	DFSMS LOADER FORK EXIT IS PRESENT
	.... ..1.		DFASNBK	SOFTWARE SUPPORT PROVIDING “FAST” BACKUP USING THE SNAPSHOT FEATURE OF THE RAMAC VIRTUAL ARRAY (RVA) INSTALLED.

Table 111. DFA Fields (continued)

Offset	Type/Value	Length	Name	Description
	1		DFASNAP	THE API SUPPORT FOR THE SNAPSHOT FEATURE OF THE RAMAC VIRTUAL ARAY (RVA) IS INSTALLED.
8(8)	CHARACTER	4	DFAACRON	ACRONYM FOR THIS CONTROL BLOCK.
12(C)	BITSTRING	1	DFAFEAT5	FEATURES BYTE 5
	1...		DFAUPDSE	UNMANAGED PDSE SUPPORT INSTALLED ON THIS SYSTEM.
	.1..		DFABTSREQ	1 means BLOCKTOKENSIZE=REQUIRE in IGDSMS*xx* member of PARMLIB. Restrictions on opening large format data sets. 0 means BLOCKTOKENSIZE=NOREQUIRE.
	..1.		DFABLDLS	BLDL START= and STOP= parameters are supported.
	...1		DFAUSEAV	System default USEEAV setting for an extended address volume (EAV) when SMS is not active. Initially set on to allow the use of EAV, and changed to the IGDSMSxx PARMLIB specified or defaulted USEEAV value if SMS is active.
	....1...		DFASAMHPF	SAM_USE_HPF, On = yes Note: If DFASAMHPF is set but ZHPF=YES in IECIOSxx is not in effect, BAM does not use zHPF.
	..... .1..			USS PIPES LBI support
	.... ..1.		DFAALVER	VERSION LEVEL AT WHICH NEW TAPE LABELS WILL BE WRITTEN. OFF: ANSI LABEL VERSION 3. ON: ANSI LABEL VERSION 4.
	1		DFAALFOR	DETERMINES WHETHER INSTALLATION ISO/ANSI VERSION LEVEL IS FORCED
13(D)	BITSTRING	1	DFAFEAT6	FEATURES BYTE 6
	1111		DFACPSDB	COPYSDB VALUE IN DEVSUPxx IN PARMLIB. SYSTEM LEVEL DEFAULT FOR THE SDB OPTION OF IEBGENER AND OTHER COPYING PROGRAMS. VAE CONSTANTS BELOW DFACPS**
	0001		DFACPSNO	COPYSDB = NO
	0010		DFACPSYE	COPYSDB = YES
	0010		DFACPSSM	COPYSDB = SMALL (SAME AS COPYSDB = YES)
	0011		DFACPSIN	COPYSDB = INPUT
	0100		DFACPSLA	COPYSDB = LARGE
	.... 1...		DFADCMET	Data class media enforced for all tapes outside of libraries (ENFORCE_DC_MEDIA=ALMEDIATY)
	.... .1..		DFADCMEJ	Data class media enforced for IBM 3592 outside of libraries (ENFORCE_DC_MEDIA=MEDIA5PLUS)
	.... ..1.		DFAMTLAM	When no media preference is expressed, accept all media types for a manual tape library (MTL_NO_DC_WORM_OK)
	1		DFASTIFF	STOW supports the IFF operand
14(E)	UNSIGNED	2	DFABPV	System default break point value (BPV) for an extended address volume (EAV) when the BPV is not specified in the SMS IGDSMSxx PARMLIB or when SMS is not active. Initially set to 10 and changed to the IGDSMSxx SPECIFIED BPV value if SMS is active.
PRODUCT, VERSION, RELEASE, AND MODIFICATION LEVEL INFORMATION. BYTE 0 DEFINES A PRODUCT CODE, BYTES 1-3 DEFINE THE VERSION, RELEASE, AND MODIFICATION LEVELS OF THIS PRODUCT. A PRODUCT BYTE (DFAPROD) OF X'00' INDICATES DFP AS A PRODUCT, BYTES 1-3 OF DFARELS WILL ALSO BE X'00' IN THIS CASE. THE USER MAY CHOOSE TO CHECK DFAREL FOR THE RELEASE LEVEL OF THE DFP PRODUCT IN THIS CASE. IF DFAPROD IS NOT EQUAL TO X'00', DFAREL SHOULD NOT BE CHECKED AS IT WILL BE FROZEN AT THE LAST LEVEL OF DFP PRODUCT SHIPPED. A PRODUCT BYTE (DFAPROD) OF X'01' INDICATES DFSMS AS A PRODUCT BYTES 1-3 OF DFARELS WILL INDICATE THE VERSION, RELEASE AND MODIFICATION LEVELS OF THE DFSMS PRODUCT. A PRODUCT BYTE (DFAPROD) OF X'02' INDICATES OS/390. THIS VALUE INDICATES THIS LEVEL OF DFSMS IS OS/390 EXCLUSIVE. SINCE DFSMS MIGHT NOT BE REFRESHED WITH EACH OS/390 RELEASE, THE VERSION, RELEASE, AND MODIFICATION FIELDS INDICATE THE LEVEL OF OS/390 IN WHICH THIS LEVEL OF DFSMS WAS FIRST SHIPPED. THE VERSION, RELEASE, AND MODIFICATION FIELDS ARE BINARY VALUES. (FOR EXAMPLE TEN WOULD BE X'0A').				
16(10)	BITSTRING	4	DFARELS	4 BYTES = PRODUCT, VERSION, REL, MOD
16(10)	BITSTRING	1	DFAPROD	PRODUCT BYTE

Table 111. DFA Fields (continued)

Offset	Type/Value	Length	Name	Description
	0000 0000		DFADFP	DFP PRODUCT CODE.
	0000 0001		DFADFSMS	PRODUCT CODE FOR DFSMS.
	0000 0002		DFAOS390	PRODUCT CODE FOR OS/390 VERSION 2.
	0000 0003		DFAZOS	PRODUCT CODE FOR z/OS.
17(11)	BITSTRING	1	DFAVER	VERSION BYTE
18(12)	BITSTRING	1	DFARLSE	RELEASE BYTE
19(13)	BITSTRING	1	DFAMOD	MODIFICATION BYTE
20(14)	SIGNED	2	DFAMSMDE	MAXIMUM LENGTH OF THE SMDE IN THIS RELEASE WITH AN 8-BYTE ALIAS NAME
22(16)	UNSIGNED	1	DFAVERBO	FLAGS
	1...		DFAMTPPRC	Multi-Target PPRC
	.1..		DFATVS	tvs flag
	..1.		DFAFCXHS	FlashCopy® across Hyperswap enabled
	...1		DFASYMCF	Reserved: PPRC symmetrical configuration
	.... 1...		DFAREJDEVGRP	SIO Exit Reject IO if
	.... .1..		DFAINCFC	Multiple Incremental FlashCopy
	11		*	Reserved
23(17)	UNSIGNED	1	DFASEFVR	Data set format version for new sequential extended format data sets. This is set by the PS_EXT_VERSION keyword in the IGDSMSxx member of SYS1.PARMLIB.
24(18)	ADDRESS	4	DFACSSVT	CALLABLE SYSTEM SERVICES VECTOR TABLE ADDRESS
28(1C)	ADDRESS	4	DFADCVSO	DATA CONVERSION SERVICES –OPEN.
32(20)	ADDRESS	4	DFADCVSD	DATA CONVERSION SERVICES –CONVERT.
36(24)	ADDRESS	4	DFADCVSC	DATA CONVERSION SERVICES –CLOSE.
40(28)	ADDRESS	4	DFAELNMP	Address of DFSMS element name. Name is mapped by DSECT DFAELNM. See Table 112 on page 443. Valid only on z/OS 1.3 and later.
44(2C)	ADDRESS	4	DFADFVAD	DATA FACILITIES VECTOR TABLE ADDR
End of DFA as it was when it was first shipped in MVS/XA DFP Version 2 Release 1 Modification Level 0. Prior to referencing any field beyond this comment, the user must ensure that DFARELS is greater than or equal to X'02020A00' or DFALEN is big enough. (See DFALEN.)				
48(30)	INTEGER	8	DFABLKSZ	LIMIT ON SYSTEM DETERMINED BLOCK SIZE. DEFAULT IS 32760. OBTAINED FROM DEVSUPxx PARMLIB MEMBER.
56(38)	BITSTRING	1	DFAFEAT7	FEATURES BYTE 7
	1...		DFATADSN	TAPEAUTHDSN=YES
	.1..		DFATADS1	TAPEAUTHF1=YES
	..1.		DFATARC8	TAPEAUTHRC8=WARN
	...1		DFATARC4	TAPEAUTHRC4=FAIL
	.... 1...		DFAXTBAM	THE NON_VSAM_XTIOT OPTION OF THE DEVSUPxx MEMBER OF PARMLIB HAS BEEN SET TO 'YES'.
	.... .1..		DFATPMVA	TAPEMULTIVOLUMEERROR=ALLOW.
	.... ..1.		DFATPMVF	TAPEMULTIVOLUMEERROR=FAIL.
	1		DFA253	SUBSYSTEMS SUPPORTED
57(39)	BITSTRING	1	DFAFEAT8	FEATURES BYTE 8. The following 7 flags indicate VSAM and non-VSAM data set support for the Extended Addressing Space (EAS) on an EAV.
THE FOLLOWING 8 FLAGS INDICATE VSAM AND NON-VSAM DATA SET SUPPORT FOR THE EXTENDED ADDRESSING SPACE (EAS) ON AN EAV				
	1...		DFAVSAMFOREAS	VSAM enabled for EAS
	.1..		DFASEQFOREAS	Basic, large format sequential (QSAM, BSAM, BDAM access) enabled for EAS
	..1.		DFAPDSEFOREAS	PDSE enabled for EAS

Table 111. DFA Fields (continued)

Offset	Type/Value	Length	Name	Description
	...1		DFAPDSFOREAS	PDS enabled for EAS
	.... 1...		DFADIRFOREAS	Direct (BDAM access) enabled for EAS
	.... .1..		DFAEFSEQFOREAS	Extended format sequential enabled for EAS
	.... ..1.		DFAUNDEFFOREAS	Undef DSORGs enabled for EAS
	1		DFAEXPMSG	EXPIRATION_MESSAGE=NEVER
58(3A)	Unsigned	2	DFADDRSZ	Storage size limit allowed in DDR swap (number of megabytes)
60 (3C)	Bit string	1	DFAFEAT9	Features byte 9
	1...		DFAJ3AA	JES3_ALLOC_ASSIST=YES in DEVSUPxx
	.1..		DFAMEMUX	This level of the system supports IEBCOPY member selection user exits
	..1.		DFAPDSEG	PDSE Generation support is installed
	...1		DFAZEDCCMP	zEDC Compression support is installed
	.... 1...		DFASSREN	DADSM extend secondary space reduction enabled
	.... .1..		DFASYSZADRV	DSS full volume dump and restore obtains SYSZADRV/ volser/SYSTEMS resource to avoid lockout
	.... ..1.		DFABYPAUTH	DCBE Bypass Authorization support is installed
	1		DFAENCRYPT	DFSMS support for Data Set Encryption is installed
61 (3D)	CHARACTER	1		Reserved
61 (3D)		3		Reserved
61 (3E)	Bit string	1	DFADEVX1	DEVSUPXX PARMLIB FLAG-1
	1...		DFAEOSV2	Erase-On-Scratch PPRC Version 2
	.1..		DFADDSFVOFF	ICKDSF VERIFY OFFLINE support
	..1.		DFADSFNODS	DSF NODSEXIST status
	...1		DFAREFUCBFA	REFUCB Failure Action
	.... 1...		DFAVTOCZHPF	VTOC_USE_ZHPF keyword
	.... .1..		DFATCTCOMP	TCTCOMPRESSION keyword
62 (3E)	Bit string	1	DFADEVX1	DEVSUPXX PARMLIB FLAG-1
	1...		DFAEOSV2	ERASE-ON-SCRATCH VERSION 2
	.1..		DFADDSFVOFF	Disable ICKDSF VERIFYOFFLINE
	..1.		DFADSFNODS	ICKDSF NODSEXIST status flag
	...1		DFAREFUCBFA	REFUCB Failure action flag
63 (3F)	Bit string	1	DFADEVX2	DEVSUPXX PARMLIB FLAG-2
64 (40)		8		Reserved
72 (48)	Bit string	4	DFAFEATC	HPF/FCX Feature Code bytes
72 (48)	Bit string	1	DFAHPFC1	HPF Features
	.... 1...		DFAFCX_TTEDcw	Transfer TCA Extension
	.... .1..		DFAFCX_REL1	FCX phase 1
	.... ..1.		DFAFCX_ImbeddedLR	Imbedded LR List
	1		DFAFCX_FmtUpdWrt	Format Update Writes
73 (49)	Bit string	1		Second HPF feature code byte
	1...		DFAFCX_FmtWrite	Format Write Enable
74 (4A)	Bit string	1		Third HPF feature code byte
75 (4B)	Bit string	1		Fourth HPF feature code byte
76 (4C)	Unsigned	4	DFAMAXGN	Maximum number of generations allowed in a PDSE
80 (50)	Unsigned	1	DFACMPTYPE	Default compression type

Table 111. DFA Fields (continued)

Offset	Type/Value	Length	Name	Description
			DFACMPTYPEGEN	0 = Generic compression
			DFACMPTYPETLRD	1 = Tailored compression
			DFACMPTYPEzEDCR	2 = zEDC compression required
			DFACMPTYPEzEDCP	3 = zEDC compression preferred
81 (51)	Bit string	1	DFAFEAT10	Features byte 10
	...1		DFADEBLock	DEBCHK Lock function support
	1...		DFAROSEC	Read-Only Secondary support is installed on the system
	.1..		DFADSSBAbove	DSSB Above the Bar for data sets accessed with VSAM or MMSRV
	..1.		DFADEVTYPEXTIOT	DEVTYPE XTIOT=YES/NO support
	...1		DFADEBLock	DEBCHK lock function support
	.... 1...		DFASyncIoWrites	zHyperLink (SyncIO) Write support installed
	.... .1..		DFAPDSEGENCPY	PDSE Generation Copy Support
	.... ..1.		DFAPDSEENCRYPT	PDSE Encryption support
82 (52)	Bit string	1	DFAFEAT11	Features byte 11
	1...		DFASEQENCRYPT	Basic and large format data set encryption support installed and initialized.
	.1..		DFACATALOGINFOVALID	Catalog fields DFACATINFO and DFACatAliasLvl below are valid.
	..1.		DFAMMDUALLOG	Media Manager Dual Log Support
	...1		DFAMMMULTVOL	Media Manager Multivolume Support for zHyperLink writes
	...1		DFAOCEExits	OPEN/CLOSE/EOV and STOW early exits are enabled
	...1 1111			Reserved
83 (53)	Bit string	1	DFACATINFO	Catalog information. These fields are provided for programs that need information about Catalog functions.
	1...		DFACatGDGExt	GDG Extended enabled
	.1..		DFACatGDGFIFO	GDG FIFO enabled
	..1.		DFACatGDGScrD	GDG Scratch Default enabled
	...1		DFACatGDGPrgD	GDG Purge Default enabled
	.... 1...		DFACatSYSPer	Catalog SYSPERCENT enabled
	.... .1..		DFSCatSMF16TS	SMF 16-byte timestamp
	11		*	Reserved
84(54)	Unsigned	1	DFACatAliasLvl	Catalog Alias levels number
85(55)	Unsigned	1	DFAGDGLimitMax	LIMIT maximum for GDG
86(56)		27		Reserved

Table 112. DFA Element Name

Offset	Type/Value	Length	Name	Description
0(0)	DSECT		DFAELNM	DSECT name
0(0)	UNSIGNED	2	DFAELNML	Length of significant characters in next field. Currently, ten.
2(2)	CHARACTER	22	DFAEXTXT	DFSMS element name in EBCDIC. Might contain lowercase characters. Currently, is "z/OS DFSMS".

Appendix B. Maintaining the System Image Library

This information describes how to maintain the system image library, SYS1.IMAGELIB. SYS1.IMAGELIB is a partitioned data set (a PDSE is not supported) containing universal character set (UCS), forms control buffer (FCB), and printer control information for DFSMSdfp–supported IBM printers in the following forms, depending on the type of printer:

- UCS images (also called modules)
- UCS image tables
- FCB images (also called modules)
- Control modules.

The system uses UCS images and image tables to relate a user-requested UCS to the corresponding print band/train. Most IBM standard UCS images are included in SYS1.IMAGELIB during system installation. Note that when you install a new release of DFSMS it replaces images that have the IBM-supplied names. If you modified or replaced any of these you might want to maintain a separate copy of them. The following table shows the standard character set images for the IBM 1403, 3203, and 3211 printers:

Printer	Images
1403 or 3203	AN, HN, PCAN, PCHN, PN, QN, QNC, RN, SN, TN, XN, YN
3211	A11, G11, H11, P11, T11

For detailed, printer-specific information, or to determine which print bands/trains are available, see the publications listed in the following table:

Publication Title	Contents
IBM 2821 Control Unit Component Description	Information on creating a user-designed chain/train for the 1403 Printer.
IBM 3203 Printer Component Description and Operator's Guide	Information on creating a user-designed train for the 3203 Printer.
IBM 3211 Printer, 3216 Interchangeable Train Cartridge, and 3811 Printer Control Unit Component Description and Operator's Guide	Information on creating a user-designed train for the 3211 Printer.
IBM 3262 Model 5 Printer Product Description	Information on band IDs for the 3262 Model 5 Printer.
IBM 3800 Printing Subsystem Programmer's Guide	Information on planning for, conversion to, and use of the IBM 3800 Model 1 Printing Subsystem.
IBM 3800 Printing Subsystem Model 3 Programmer's Guide: Compatibility	Information on planning for, conversion to, and use of the IBM 3800 Model 3, 6, and 8 Printing Subsystem.
IBM 4245 Printer Model 1 Component Description and Operator's Guide	Information on band IDs for the 4245 Printer.
IBM 4248 Printer Model 1 Description	Information on band IDs for the 4248 Printer.
IBM 6262 Printer Print Band Manual	Information on band IDs for the 6262 Printer.
IBM 6262 Printer Model 014 User's Guide	Information on operating and maintaining the 6262 Model 14 Printer.

To use the information in this chapter, you should be familiar with the publications listed in the following table:

Publication Title	Contents
z/OS DFSMS Macro Instructions for Data Sets	Describes the SETPRT macro. You can use it to specify the images (modules) that you want.
z/OS DFSMSdfp Utilities	Describes the IEBIMAGE utility program.
z/OS MVS JCL Reference	Describes the CHARS, MODIFY, UCS, and FCB parameters of the DD statement, that are processed at OPEN.
z/OS JES2 Initialization and Tuning Guide	Contains JES2 reference information.
z/OS JES3 Initialization and Tuning Guide	Contains JES3 reference information.

Table 113 on page 446 provides information about the printer-specific contents of SYS1.IMAGELIB.

Table 113. SYS1.IMAGELIB Contents

Printer Type	UCS Image	UCS Image Table 8	FCB Image	Control Module
1403	X			
3203 [1]	X		X	
3211	X		X	
3262 Model 5 [2, 4, 5]		X	X	
4245 [3]		X[6]	X	
4248 [3]		X[6]	X	
6262 Model 14 [2, 4, 5]		X	X	
3800				X [7]

Notes to Table 113 on page 446:

1. The IBM 3203 Model 5 Printer is treated as a 3211 Printer by JES, except that the 3203 Model 5 does not support the 3211 indexing feature and ignores any indexing commands from JES. The 3203 Model 5 uses its own unique UCS images, but uses 3211 FCB images.
2. You can operate this printer as if it were a 4248 operating in 4248 native mode. In this case, use the 4248-related UCS sections of this chapter. The UCS information is contained in image tables.
3. You can operate this printer in native mode, in which case the UCS information is contained in image tables. You can also operate these printers in 3211 compatibility mode, in which case the UCS information is contained in UCS images; you should then use the 3211-related UCS sections of this chapter.
4. This printer uses the same image table as the 4248 printer. However, it does not support variable printer speeds or the horizontal copy feature and host stacker controls of the 4248 printer.
5. This printer uses the same FCB image as the 4248 Printer.
6. The image table is supplied by IBM. The contents of the 4245 image table are shown in Table 114 on page 452. The contents of the 4248 image table are shown in Table 115 on page 453.
7. 3800 Printing Subsystem control modules exist for:
   - Character arrangement tables
   - Graphics character modification tables
   - Copy modification tables
   - Library character sets
   - FCB images.

You can use the IEBIMAGE utility program to create and maintain these control modules. See *z/OS DFSMSdfp Utilities* for details about IEBIMAGE. IEBIMAGE and SYS1.IMAGELIB are not used when the printer is running in page mode. In that case, PSF for z/O uses other types of modules from other libraries. See *3900 Product Description* for more information.

8. Figure 45 on page 452 defines and describes the structure of a UCS image table entry.

UCS Images in SYS1.IMAGELIB

This information applies to the IBM 1403, 3203, and 3211 printers. SYS1.IMAGELIB contains UCS images for these printers. You can use the assembler and linkage editor to add a UCS image to those that reside in SYS1.IMAGELIB. The assembler does not generate executable code. It merely prepares DC statements, and the linkage editor puts them into SYS1.IMAGELIB. Observe the following rules when creating a new UCS image:

1. The member name must be 5 to 8 characters long; the first 4 characters must be the appropriate UCS prefix, as follows:

Prefix	Meaning
UCS1	1403 printer
UCS2	3211 printer (or 3211-compatible printer)
UCS3	3203 printer

These first four characters must be followed by a character set code, one to four characters long. Any valid combination of letters and numbers under assembler language rules is acceptable. However, do not use the single letters U or C, because they are symbols for special conditions recognized by the system. Specify the assigned character set code on the DD statement or SETPRT macro to load the image into the UCS buffer.

You can supply an alias name for a new image with the ALIAS statement. (For more information on the ALIAS statement, see *z/OS MVS Program Management: User's Guide and Reference*.)

2. The first byte of the character set image load module specifies whether the image is a default. If a program issues an OPEN macro to a printer in which the UCS buffer has been loaded with a default image and the JCL does not specify a UCS name, the system uses the image in the buffer. If the buffer has not been loaded with a default image the system directs the operator to take action.

Specify the following in the first byte for JES2:

Value	Meaning
X'00'	Indicates that the image is not to be used as a default.
X'40'	Indicates that the output is to be folded.
X'80'	Indicates a default image.
X'C0'	Indicates default image and folding.

For non-JES2, specify:

Value	Meaning
X'00'	Indicates that the image is not to be used as a default.
X'80'	Indicates a default image.

3. The second byte of the load module indicates the number of lines (n) to be printed for image verification. See "Verifying the UCS Image" on page 458 for more information on image verification.
4. Each byte of the next n bytes indicates the number of characters to be printed on each verification line. For the 3211 Printer, the maximum number of characters printed per line is 48; the bytes of associative bits (see Rule "5" on page 448, which follows) do not print during verification.

5. The UCS image itself must follow the previously described fields. The image must fill the number of bytes required by the printer; see the following table for image lengths. Note that, because of Assembler language syntax, you must code two apostrophes or two ampersands to represent a single apostrophe or a single ampersand, respectively, within a character set image.

Printer	Image Length
1403	240 bytes
3203	304 bytes (240 characters followed by 64 bytes of associative bits)
3211	512 bytes (432 characters followed by 15 bytes of X'00', 64 bytes of associative bits, and 1 reserved byte of X'00')

   You must code associative bits to prevent data checks when adding a UCS image to SYS1.IMAGELIB. See the appropriate printer publication for more information on coding associative bits.

See the following information:

- "Examples of UCS Image Coding" on page 448
- "UCS Image Alias Names" on page 451
- "UCS Image Tables in SYS1.IMAGELIB" on page 451
- "Alias Names in UCS Image Tables" on page 451
- "Adding or Modifying a UCS Image Table Entry" on page 455
- "Verifying the UCS Image" on page 458

Examples of UCS Image Coding

Figure 42 on page 448 shows an example of the JCL to add a 1403 UCS image, YN, to SYS1.IMAGELIB. Notes, which apply to all examples, follow Figure 44 on page 450.

```
//ADDYN     JOB  MSGLEVEL=1
//STEP      EXEC PROC=ASMHCL,PARM.ASM='NODECK,LOAD',
//               PARM.LKED='LIST,OL,REFR,RENT,XREF'
//ASM.SYSIN DD *
UCS1YN   CSECT
         DC   X'80'        (THIS IS A DEFAULT IMAGE)
         DC   AL1(6)       (NUMBER OF LINES TO BE PRINTED)
         DC   AL1(39)      (39 CHARACTERS TO BE PRINTED ON LINE 1)
         DC   AL1(42)      (42 CHARACTERS TO BE PRINTED ON LINE 2)
         DC   AL1(39)      (39 CHARACTERS TO BE PRINTED ON LINE 3)
         DC   AL1(39)      (39 CHARACTERS TO BE PRINTED ON LINE 4)
         DC   AL1(42)      (42 CHARACTERS TO BE PRINTED ON LINE 5)
         DC   AL1(39)      (39 CHARACTERS TO BE PRINTED ON LINE 6)
*    THE FOLLOWING SIX LINES REPRESENT THE TRAIN IMAGE
         DC   C'1234567890STABCDEFGHIJKLMNOPQRUVWXYZ*,.'
         DC   C'1234567890STABCDEFGHIJKLMNOPQRUVWXYZ*,.#-$'
         DC   C'1234567890STABCDEFGHIJKLMNOPQRUVWXYZ*,.'
         DC   C'1234567890STABCDEFGHIJKLMNOPQRUVWXYZ*,.'
         DC   C'1234567890STABCDEFGHIJKLMNOPQRUVWXYZ*,.#-$'
         DC   C'1234567890STABCDEFGHIJKLMNOPQRUVWXYZ*,.'
         END
/*
//LKED.SYSLMOD DD DSNAME=SYS1.IMAGELIB(UCS1YN),DISP=OLD,
//             SPACE=        (OVERRIDE SECONDARY ALLOCATION)
```

Figure 42. Code to add a 1403 UCS image to SYS1.IMAGELIB

See Figure 44 on page 450 for the notes to this figure.

Figure 43 on page 449 shows an example of the JCL to add a 3203 UCS image, YN, to SYS1.IMAGELIB. Notes to this figure follow Figure 44 on page 450.

```
//ADYN3203  JOB  MSGLEVEL=1
//STEP      EXEC PROC=ASMHCL,PARM.ASM='NODECK,LOAD',
//             PARM.LKED='LIST,OL,REFR,RENT,XREF'
//ASM.SYSIN DD *
UCS3YN   CSECT
         DC   X'80'        (THIS IS A DEFAULT IMAGE)
         DC   AL1(6)       (NUMBER OF LINES TO BE PRINTED)
         DC   AL1(39)      (39 CHARACTERS TO BE PRINTED ON LINE 1)
         DC   AL1(42)      (42 CHARACTERS TO BE PRINTED ON LINE 2)
         DC   AL1(39)      (39 CHARACTERS TO BE PRINTED ON LINE 3)
         DC   AL1(39)      (39 CHARACTERS TO BE PRINTED ON LINE 4)
         DC   AL1(42)      (42 CHARACTERS TO BE PRINTED ON LINE 5)
         DC   AL1(39)      (39 CHARACTERS TO BE PRINTED ON LINE 6)
*    THE FOLLOWING SIX LINES REPRESENT THE TRAIN IMAGE
         DC   C'1234567890STABCDEFGHIJKLMNOPQRUVWXYZ*,.'
         DC   C'1234567890STABCDEFGHIJKLMNOPQRUVWXYZ*,.#-$'
         DC   C'1234567890STABCDEFGHIJKLMNOPQRUVWXYZ*,.'
         DC   C'1234567890STABCDEFGHIJKLMNOPQRUVWXYZ*,.'
         DC   C'1234567890STABCDEFGHIJKLMNOPQRUVWXYZ*,.#-$'
         DC   C'1234567890STABCDEFGHIJKLMNOPQRUVWXYZ*,.'
*    THE FOLLOWING FOUR DC INSTRUCTIONS DEFINE THE ASSOCIATIVE BITS,
*    UCSB BYTE POSITIONS 241-304
         DC   X'C0101010101010101010004000000000010'
         DC   X'1010101010101010004040000000040001010'
         DC   X'1010101010100040000000001010101010'
         DC   X'10101010004000000'
         END
/*
//LKED.SYSLMOD DD DSNAME=SYS1.IMAGELIB(UCS3YN),DISP=OLD,
//             SPACE=         (OVERRIDE SECONDARY ALLOCATION)
```

Figure 43. Code to add a 3203 UCS image to SYS1.IMAGELIB

Notes: See Figure 44 on page 450 for the notes to this figure.

Figure 44 on page 450 shows an example of the JCL to add a 3211 UCS image, A11, to SYS1.IMAGELIB.

```
//ADDA11    JOB   MSGLEVEL=1
//STEP      EXEC PROC=ASMHCL,PARM.ASM='NODECK,LOAD',
//             PARM.LKED='LIST,OL,REFR,RENT,XREF'
//ASM.SYSIN DD *
UCS2A11  CSECT
         DC   X'80'         (THIS IS A DEFAULT IMAGE)
         DC   AL1(9)        (NUMBER OF LINES TO BE PRINTED)
         DC   AL1(48)       (48 CHARACTERS TO BE PRINTED ON LINE 1)
         DC   AL1(48)       (48 CHARACTERS TO BE PRINTED ON LINE 2)
         DC   AL1(48)       (48 CHARACTERS TO BE PRINTED ON LINE 3)
         DC   AL1(48)       (48 CHARACTERS TO BE PRINTED ON LINE 4)
         DC   AL1(48)       (48 CHARACTERS TO BE PRINTED ON LINE 5)
         DC   AL1(48)       (48 CHARACTERS TO BE PRINTED ON LINE 6)
         DC   AL1(48)       (48 CHARACTERS TO BE PRINTED ON LINE 7)
         DC   AL1(48)       (48 CHARACTERS TO BE PRINTED ON LINE 8)
         DC   AL1(48)       (48 CHARACTERS TO BE PRINTED ON LINE 9)
*     THE FOLLOWING NINE LINES REPRESENT THE TRAIN IMAGE
*     NOTE 2 AMPERSANDS MUST BE CODED TO GET 1 IN ASSEMBLER SYNTAX
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   C'1<.+IHGFEDCBA*$-RQPONMLKJ%,&&ZYXWVUTS/@#098765432'
         DC   15X'00'         (RESERVED FIELD, BYTES 433-447)
*     THE FOLLOWING FOUR DC INSTRUCTIONS DEFINE THE ASSOCIATIVE BITS,
*     UCSB BYTE POSITIONS 448-5111
         DC   X'C0101010101010101010000404042400004010'
         DC   X'1010101010101010000404041000040401010'
         DC   X'1010101010100004040000000101010101010'
         DC   X'101010100040404444800'
         DC   X'00'         (RESERVED FIELD, BYTE 512)
         END
/*
//LKED.SYSLMOD DD DSNAME=SYS1.IMAGELIB(UCS2A11"DISP=OLD,
//          SPACE=       (OVERRIDE SECONDARY ALLOCATION)
```

Figure 44. Sample code to Add a 3211 UCS image to SYS1.IMAGELIB

In the sample code in the Figure 42 on page 448, Figure 43 on page 449, and Figure 44 on page 450 be aware of the following:

- The RENT linkage editor attribute is required.
- For the 3203 and 3211 printers, to avoid data checks code the 64 bytes of associative bits. To determine how to code these bits for a particular image, see *IBM 3203 Printer Component Description and Operator's Guide* or *IBM 3211 Printer, 3216 Interchangeable Train Cartridge, and 3811 Printer Control Unit Component Description and Operator's Guide.*
- Executing the ASMHCL procedure does not generate executable code. The assembler/linkage editor merely places the UCS image into SYS1.IMAGELIB.
- The SPACE parameter is overridden here because the ASMHCL cataloged procedure has secondary allocation specified. You can specify use of the original secondary allocation amount by deleting the override.

UCS Image Alias Names

Alias names are provided for many of the IBM-supplied print bands and trains. For example, if the data set were printed on a 3211, a request for the 1403 TN train would assign the T11 train. The assigned alias names that follow the naming conventions currently used in SYS1.IMAGELIB are:

Image	Alias
UCS1AN	UCS1A11
UCS1HN	UCS1H11
UCS1PN	UCS1P11
UCS1TN	UCS1T11
UCS2A11	UCS2AN
UCS2H11	UCS2HN
UCS2P11	UCS2PN, UCS2RN, UCS2QN
UCS2T11	UCS2TN

The image and alias names are included in SYS1.IMAGELIB at system installation.

Some bands/trains, such as SN and G11, do not have aliases because neither has an equivalent band/train on the other printer. An installation can assign an alias, if it chooses. (For details about the ALIAS statement, see *z/OS MVS Program Management: User's Guide and Reference*.) If you do not specify a name (or alias), you must specify an installation-defined SYSOUT class or a printer routing code to assign the data set to the correct printer. If JES is directed to print a data set on a printer for which the specified image does not exist, JES notifies the operator. The operator can then cause the data set to be printed with a valid band/train or redirect the data set to the proper printer. If an installation defines a new band/train, it can supply an alias name for it through the ALIAS statement when including the image in SYS1.IMAGELIB.

UCS Image Tables in SYS1.IMAGELIB

This section applies only to the IBM 3262 Model 5, 4245 , 4248, and 6262 Model 14 printers. SYS1.IMAGELIB does not contain UCS images for these printers, but, instead, contains image tables. If you are running the printer in 3211 compatibility mode, UCS information is contained in image tables. Use the 3211-related sections of this chapter. The UCS image for each band is stored within the printer and is automatically loaded into the UCS buffer when you turn on machine power or install a new band. See Figure 45 on page 452 for the format of image table entries, and “Adding or Modifying a UCS Image Table Entry” on page 455 for information on how to add or modify an image table entry.

SYS1.IMAGELIB contains one UCS image table for each type of printer that supports image tables. An image table contains an entry for most installation-standard IBM-supplied bands. The 4245 image table is named UCS5. The shared 4248, 3262 Model 5, and 6262 Model 14 image table is named UCS6.

Alias Names in UCS Image Tables

The image tables also define alias names for most installation-standard print bands used on the IBM 4245 and 4248 printers. The IBM-supplied image tables do not provide alias names for the IBM 3262 Model 5 or 6262 Model 14 printers.

Some print bands, such as SN and KA22, do not have alias names because there is no equivalent band on other printers. You can add an alias name by adding or modifying an entry in the appropriate UCS image table. See “Adding or Modifying a UCS Image Table Entry” on page 455. A typical UCS image table entry is shown in Figure 45 on page 452.

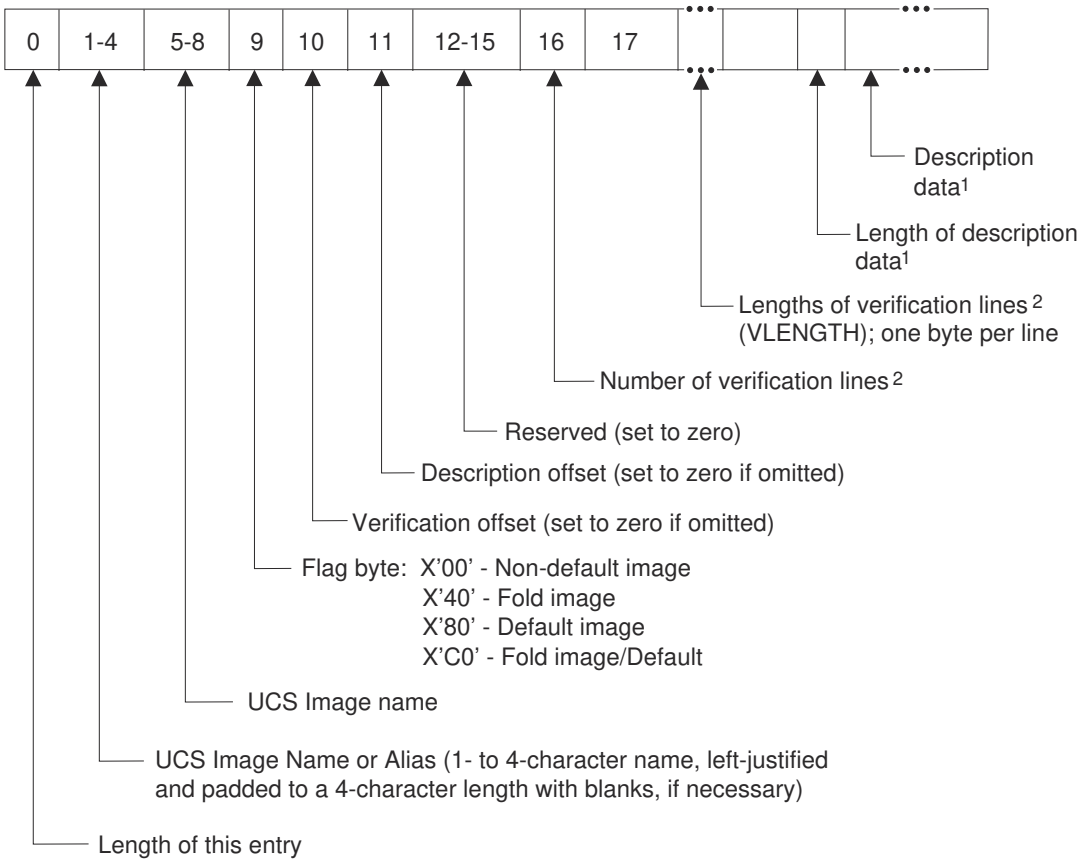


Figure 45. UCS image table entry format

Notes to Figure 45 on page 452:

1. This field is optional. The description data field is of variable length, up to a maximum of 32 bytes.
2. This field is optional for the 4245 Printer. For the 3262 Model 5, 4248, and 6262 Model 14, this field does not apply and is set to X'00'.

The contents of the UCS image table UCS5 (IGGUCS5 macro) for the 4245 Printer are shown in Table 114 on page 452.

Table 114. UCS5 Image Table Contents

Name	Alias	Default	Description
AN21	AN21	YES	Default UCS image
AN21	AN	NO	1403/3203 AN image
AN21	A11	NO	3211 A11 image
AN21	40E1	NO	4248 40E1 image
HN21	HN21	NO	Nondefault UCS image
HN21	HN	NO	1403/3203 HN image
HN21	H11	NO	3211 H11 image
HN21	4101	NO	4248 4101 image
PL21	PL21	NO	Nondefault UCS image
PL21	PN	NO	1403/3203 PN image

Table 114. UCS5 Image Table Contents (continued)

Name	Alias	Default	Description
PL21	P11	NO	3211 P11 image
PL21	4121	NO	4248 4121 image
SN21	SN21	NO	Nondefault UCS image
SN21	4201	NO	4248 4201 image
TN21	TN21	NO	Nondefault UCS image
TN21	TN	NO	1403/3203 TN image
TN21	T11	NO	3211 T11 image
TN21	4181	NO	4248 4181 image
GN21	GN21	NO	Nondefault UCS image
GN21	G11	NO	3211 G11 image
GN21	41C1	NO	4248 41C1 image
RN21	RN21	NO	Nondefault UCS image
RN21	RN	NO	1403/3203 RN image
KA21	KA21	NO	Nondefault UCS image
KA21	4041	NO	4248 4041 image
KA22	KA22	NO	Nondefault UCS image
FC21	FC21	NO	Nondefault UCS image
FC21	4161	NO	4248 4161 image

The contents of the UCS image table UCS6 (IGGUCS6 macro), for the 4248 printer, are shown in Table 115 on page 453.

Table 115. UCS6 Image Table Contents

Name	Alias	Default	Description
40E1	40E1	YES	Default UCS image
40E1	AN21	NO	4245 AN21 image
40E1	AN	NO	1403/3203 AN image
40E1	A11	NO	3211 A11 image
4101	4101	NO	Nondefault UCS image
4101	HN21	NO	4245 HN21 image
4101	HN	NO	1403/3203 HN image
4101	H11	NO	3211 H11 image
41C1	41C1	NO	Nondefault UCS image
41C1	GN21	NO	4245 GN21 image
41C1	G11	NO	3211 G11 image
4121	4121	NO	Nondefault UCS image
4121	PL21	NO	4245 PL21 image
4121	PN	NO	1403/3203 PN image

Table 115. UCS6 Image Table Contents (continued)

Name	Alias	Default	Description
4121	P11	NO	3211 P11 image
4181	4181	NO	Nondefault UCS image
4181	TN21	NO	4245 TN21 image
4181	TN	NO	1403/3203 TN image
4181	T11	NO	3211 T11 image
4061	4061	NO	Nondefault UCS image
40C1	40C1	NO	Nondefault UCS image
4161	4161	NO	Nondefault UCS image
4161	FC21	NO	4245 FC21 image
4201	4201	NO	Nondefault UCS image
4201	SN21	NO	4245 SN21 image
4041	4041	NO	Nondefault UCS image
4041	KA21	NO	4245 KA21 image

Tip: The image tables for the 4245 and 4248 printers include USA and Canada band IDs only. To support other national band IDs, modify the UCS image table. See Table 116 on page 454 and “Adding or Modifying a UCS Image Table Entry” on page 455.

The 3262 Model 5 and 6262 Model 14 printers use the 4248 UCS image table, UCS6. However, IBM does not provide band names or aliases for either the 3262 Model 5 or 6262 Model 14 printer. To use 3262 Model 5 or 6262 Model 14 UCS images, add the names and aliases to UCS6. “Adding or Modifying a UCS Image Table Entry” on page 455 describes how to add entries to the UCS image table. For a list of the bands available for the 3262 Model 5, see Table 116 on page 454. For a list of the bands available for the 6262 Model 14, see *IBM 6262 Printer Print Band Manual* and *IBM 6262 Printer Model 014 User's Guide.*

For Table 116 on page 454, be aware of the following:

- You can define any 1- to 4-character name as aliases of the UCS6 Band names.
- *xx* designates switch settings that are ignored. Switch number 3 must be on if you are using a special order (RPQ) band.
- Note that *bb* here represents a space.

Table 116. 3262 Model 5 Print Bands

Character Set Name	UCS6 Band Name	3262 Model 5 Band Image Select Switch Settings (positions 1-8)
U.S./International		
48 char EBCDIC	00*bb*	*xx*00 0000
63 char EBCDIC	01*bb*	*xx*00 0001
64 char EBCDIC	02*bb*	*xx*00 0010
96 char EBCDIC	03*bb*	*xx*00 0011
48 char AON OCR	04*bb*	*xx*00 0100
48 char BON OCR	05*bb*	*xx*00 0101
Austria/Germany		
52 char EBCDIC	06*bb*	*xx*00 0110
63 char EBCDIC	07*bb*	*xx*00 0111

Table 116. 3262 Model 5 Print Bands (continued)

Character Set Name	UCS6 Band Name	3262 Model 5 Band Image Select Switch Settings (positions 1-8)
64 char EBCDIC	08*bb*	*xx*00 1000
96 char EBCDIC	09*bb*	*xx*00 1001
52 char AON OCR	0A*bb*	*xx*00 1010
52 char BON OCR	0B*bb*	*xx*00 1011
Canada/French		
116 char EBCDIC	0C*bb*	*xx*00 1100
Katakana		
96 char EBCDIC	0D*bb*	*xx*00 1101
128 char EBCDIC	0E*bb*	*xx*00 1110
UINN		
128 char U.S. text	0F*bb*	*xx*00 1111
WTNN		
128 char World Trade text	10*bb*	*xx*01 0000

Adding or Modifying a UCS Image Table Entry

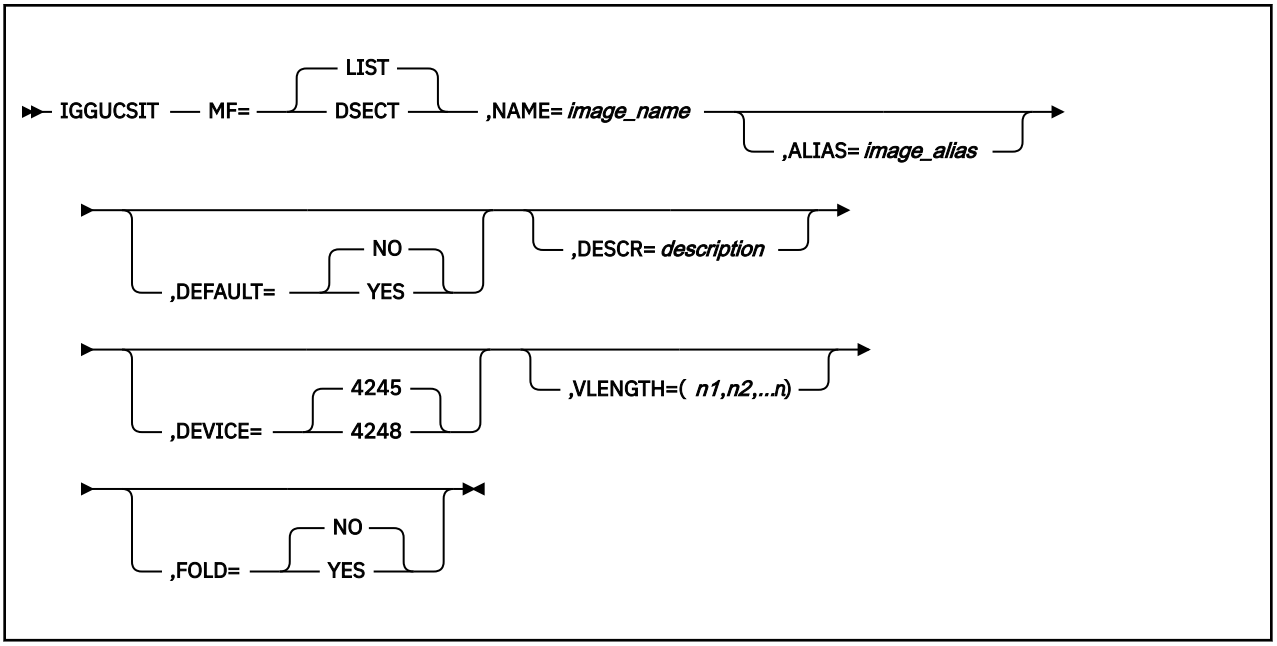
To use a new UCS image name/alias with the 3262 Model 5, 4245, 4248, or 6262 Model 14 printers, add an entry for that image name/alias to the UCS image table. Use the assembler to create the image table object module, then link-edit the object module into SYS1.IMAGELIB, as shown in the following procedure. Similarly, to specify other images as defaults or change the description on an old image, change the image table.

To build new UCS table entries, or to change the format of old entries, use the following procedure. For examples of coding the IGGUCSIT macro, see Figure 46 on page 457 and Figure 47 on page 458.

1. To build a new UCS image table entry issue the IGGUCSIT macro, as described in the following text. If you are updating the image table as shown in the following examples, the linkage editor builds a new entry at the start of the table, even if you intended to replace an existing entry. When the system subsequently uses the table, it encounters the new entry first, thus the old one is effectively replaced.
2. Include the UCS image table source, using the IGGUCS5 or IGGUCS6 macro, both of which reside in SYS1.MODGEN.
3. Assemble the image table module (UCS5 or UCS6).
4. Link-edit the assembled module into SYS1.IMAGELIB.

   Requirement: The RENT linkage editor attribute is required.

The format of the IGGUCSIT macro is:



MF=LIST or DSECT
Specifies the form of the macro instruction.

LIST
Produces a UCS image table entry based on the information supplied in other IGGUCSIT parameters. If LIST is selected or allowed to default, the NAME parameter must also be coded.

DSECT
Produces a DSECT for a single UCS image table entry, similar to the sample entry shown in Figure 45 on page 452. If you code DSECT, all other parameters of IGGUCSIT are ignored.

LIST is the default.

NAME=*image_name*
Specifies the one to four character UCS image name.

ALIAS=*image_alias*
Specifies a one to four character alias name for the UCS image. If ALIAS is not specified, the image name coded in the NAME parameter will be entered in the UCS image table.

Exception: The 3262, 4248, and 6262 printers have no band IDs in common. Because these devices all use the UCS6 image table, select a unique alias name for each of these printer types. The alias must not appear in the image table more than once.

DEFAULT=YES or NO
Indicates whether the new UCS image is to be used as a default value.

YES
Indicates that this UCS image is a default. Default images are used by the system for jobs that do not request a specific image.

NO
Indicates that this UCS image should not be used as a default.

If the DEFAULT parameter is not specified, the new UCS image is not used as a default.

DESCR=*description*
Specifies descriptive information about the new UCS image. *description* can be up to 32 EBCDIC or hexadecimal characters long. You cannot use EBCDIC and hexadecimal characters in combination.

Descriptive information is placed in the header line of the verification display, following the real UCS image name. If you omit the DESCR parameter, no description appears in the display. For more information on the verification display, see "Verifying the UCS Image" on page 458.

If VLENGTH is not specified for the 4245 Printer, the DESCR parameter is ignored.

DEVICE=4245 or 4248
Specifies the type of device for which an image table entry is to be created.

If you specify MF=LIST on the first invocation of the IGGUCSIT macro, DEVICE defaults to 4245. The default for subsequent invocations is the printer type that you specified (or the default) on the first invocation. Table entries with different DEVICE specifications are not allowed.

For the 3262 Model 5 or 6262 Model 14 printers, DEVICE=4248 should be specified to create the appropriate form of the image table entry.

VLENGTH=(*n1*,*n2*,. . . *n*)
Specifies the lengths of each line in the UCS verification display. The length of each line must be specified separately, even if all lines are of the same length.

n1 is the length of print line 1; *n2* is the length of print line 2; *n* is the length of the last print line. To display the complete image, the sum of the verification line lengths should equal 350.

For details on the verification report, see "Verifying the UCS Image" on page 458.

The VLENGTH parameter is not valid for the 3262 Model 5, 4248, or 6262 Model 14 printers.

FOLD=YES or NO
Indicates whether the UCS image is to be folded.

YES
Indicates that the UCS image is to be folded. Allows printing only uppercase characters from either upper- or lowercase data codes. Folding continues until an UNFOLD command is received.

NO
Indicates that the UCS image is not to be folded. This is the default.

Adding to the UCS Image Table

In Figure 46 on page 457, the band name RPQ1 with description "RPQ BAND" is added to UCS5. In the UCS verification display, 7 lines of 50 characters each are printed. Macro IGGUCS5 causes the UCS image table source (as distributed by IBM) to be included in the table entry.

```
                                                                      72
//UCS5   JOB  . . .
//       EXEC ASMHCL,
//            PARM.ASM='NODECK,LOAD',
//            PARM.LKED='OL,RENT,REUS'
//SYSPRINT  DD SYSOUT=A
//ASM.SYSIN DD *
         TITLE 'UPDATED UCS5 IMAGE TABLE'
UCS5     CSECT
         IGGUCSIT NAME=RPQ1,                                          X
               VLENGTH=(50,50,50,50,50,50,50),                        X
               DESCR='RPQ BAND'
         IGGUCS5
         END
/*
//LKED.SYSLMOD DD DSN=SYS1.IMAGELIB(UCS5),DISP=OLD,
//             SPACE=  (OVERRIDE SECONDARY ALLOCATION)
```

Figure 46. Adding a New Band ID to the 4245 UCS Image Table (UCS5)

When adding a new band ID to the 4245 UCS image table as shown in Figure 46 on page 457, be aware of the following:

- The RENT linkage editor attribute is required.

- Executing the ASMHCL procedure does not generate executable code. The assembler/linkage editor places the updated UCS image table into SYS1.IMAGELIB.
- The SPACE parameter is overridden here because the ASMHCL cataloged procedure has a secondary allocation specified. Eliminating the override causes the original secondary allocation amount to be used.

In Figure 47 on page 458 the band name 40E1 DEFAULT BAND has been added to UCS6 and defined as a default band. An alias name, HN21, is also defined for band 40E1. Macro IGGUCS6 causes the UCS image table source (as distributed by IBM) to be included in the table entry.

```
                                                                    72
//UCS6   JOB  . . .
//       EXEC ASMHCL,
//            PARM.ASM='NODECK,LOAD',
//            PARM.LKED='OL,RENT,REUS'
//SYSLIB    DD
//          DD DSN=SYS1.AMODGEN,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//ASM.SYSIN DD *
         TITLE 'UPDATED UCS6 IMAGE TABLE'
UCS6     CSECT
         IGGUCSIT NAME=40E1,                                       X
               DEVICE=4248,                                        X
               ALIAS=HN21,                                         X
               DEFAULT=YES,                                        X
               DESCR='40E1 DEFAULT BAND'
         IGGUCS6
         END
/*
//LKED.SYSLMOD DD DSN=SYS1.IMAGELIB(UCS6),DISP=OLD,
//             SPACE=  (OVERRIDE SECONDARY ALLOCATION)
```

Figure 47. Adding a New Default Entry to the 4248 UCS Image Table (UCS6).

For Figure 47 on page 458, be aware of the following:

- This method creates a duplicate entry for 40E1 that becomes the first entry in the table. Because the table is searched sequentially, the new entry is always found before the old entry, thus replacing the old entry.
- The RENT linkage editor attributes is required.
- Executing the ASMHCL procedure does not generate executable code. The assembler/linkage editor places the updated UCS image table into SYS1.IMAGELIB.
- The SPACE parameter is overridden because the ASMHCL cataloged procedure has a secondary allocation specified. Eliminating the override causes the original secondary allocation amount to be used.

Verifying the UCS Image

For the 1403 (with the UCS feature), 3203, 3211, 3262 Model 5, 4245, 4248, and 6262 Model 14 printers, you can print the UCS image for visual verification using either of the following parameters:

- In JCL: UCS=(*character set code*,,VERIFY)
- In the SETPRT macro: UCS=(*character set code*,,V).

These parameters have no effect for SYSOUT data sets.

You can also use these parameters for the 3262 Model 5, 4248, and 6262 Model 14 printers. However, because the UCS image cannot be read directly from the 3262 Model 5, 4248, or 6262 Model 14, only the header information is printed. The verification display header appears on the printer as follows:

```
UCS IMAGE VERIFICATION image_id [,FOLD] [description]
```

image_id
A one to four character name of the UCS image.

description
The descriptive information supplied for this UCS image in the UCS image table.

For more information about the UCS VERIFY parameters, see *z/OS MVS JCL Reference* and *z/OS DFSMS Macro Instructions for Data Sets*.

FCB Images in SYS1.IMAGELIB

Two standard FCB images, STD1 and STD2, are included in SYS1.IMAGELIB during system installation. The names of these standard images begin with the characters “FCB2”. You can define FCB images whose names begin with “FCB4” for the 4248, 3262 Model 5, and 6262 Model 14 Printers. These printers can use both the “FCB2”-prefixed images (referred to as 3211 format FCBs) and the “FCB4”-prefixed images. All other printers, except the 3800, can use only “FCB2”-prefixed images.

This section describes how you can create or replace “FCB2”-prefixed images. *z/OS DFSMSdfp Utilities* describes how you can use the IEBIMAGE utility program to create, update, or replace an “FCB4”-prefixed image for the 3262 Model 5, 4248, or 6262 Model 14 Printers. For details about the “FCB4”-prefixed images, see either *IBM 6262 Printer Model 014 Product Description* or *IBM 4248 Printer Model 1 Description.*

The 3262 Model 5, 4245, 4248, and 6262 Model 14 printers each load a default FCB image into the buffer when they are powered on. The 3262 Model 5 default FCB image is an 11-inch form with 6 lines per inch, a Channel 1 on the third print line, and a Channel 12 on line 64. The 4245 default FCB image is an 11-inch form with 6 lines per inch and a Channel 1 on the first print line. The 4248 default FCB image is the last FCB image loaded. The 6262 Model 14 default FCB image is either the last FCB image loaded and saved or the default shipped with the printer.

STD1 sets line spacing at 6 lines per inch for an 8½ inch form; STD2 is a default FCB image that sets line spacing at 6 lines per inch for an 11-inch form. Channels for both images are evenly spaced, with Channel 1 on the fourth line and Channel 9 on the last line. See Figure 48 on page 460 and Figure 49 on page 460 for the format of the standard STD1 and STD2 images.

The standard FCB image for the 3800 Printing Subsystem, STD3, is included in SYS1.IMAGELIB during system installation. All models of the 3800 use FCB images whose names begin with "FCB3". Use the IEBIMAGE utility to create and modify FCB modules for the 3800 Printing Subsystem.

```
FCB2STD1  CSECT
          DC    X'80'             DEFAULT
          DC    AL1(48)           FCB IMAGE LENGTH = 48
          DC    X'000000'         LINE 1, 2, 3
          DC    X'01'             LINE 4, CHANNEL 1
          DC    X'000000'         LINE 5, 6, 7
          DC    X'02'             LINE 8, CHANNEL 2
          DC    X'000000'         LINE 9, 10, 11
          DC    X'03'             LINE 12, CHANNEL 3
          DC    X'000000'         LINE 13, 14, 15
          DC    X'04'             LINE 16, CHANNEL 4
          DC    X'000000'         LINE 17, 18, 19
          DC    X'05'             LINE 20, CHANNEL 5
          DC    X'000000'         LINE 21, 22, 23
          DC    X'06'             LINE 24, CHANNEL 6
          DC    X'000000'         LINE 25, 26, 27
          DC    X'07'             LINE 28, CHANNEL 7
          DC    X'000000'         LINE 29, 30, 31
          DC    X'08'             LINE 32, CHANNEL 8
          DC    X'000000'         LINE 33, 34, 35
          DC    X'0A'             LINE 36, CHANNEL 10
          DC    X'000000'         LINE 37, 38, 39
          DC    X'0B'             LINE 40, CHANNEL 11
          DC    X'000000'         LINE 41, 42, 43
          DC    X'0C'             LINE 44, CHANNEL 12
          DC    X'000000'         LINE 45, 46, 47
          DC    X'19'             LINE 48, CHANNEL 9-END OF FCB IMAGE
          END
```

Figure 48. Format of the standard STD1 FCB image

```
FCB2STD2  CSECT
          DC    X'80'             DEFAULT
          DC    AL1(66)           FCB IMAGE LENGTH = 66
          DC    X'000000'         LINE 1, 2, 3
          DC    X'01'             LINE 4, CHANNEL 1
          DC    X'0000000000'     LINE 5, 6, 7, 8, 9
          DC    X'02'             LINE 10, CHANNEL 2
          DC    X'0000000000'     LINE 11, 12, 13, 14, 15
          DC    X'03'             LINE 16, CHANNEL 3
          DC    X'0000000000'     LINE 17, 18, 19, 20, 21
          DC    X'04'             LINE 22, CHANNEL 4
          DC    X'0000000000'     LINE 23, 24, 25, 26, 27
          DC    X'05'             LINE 28, CHANNEL 5
          DC    X'0000000000'     LINE 29, 30, 31, 32, 33
          DC    X'06'             LINE 34, CHANNEL 6
          DC    X'0000000000'     LINE 35, 36, 37, 38, 39
          DC    X'07'             LINE 40, CHANNEL 7
          DC    X'0000000000'     LINE 41, 42, 43, 44, 45
          DC    X'08'             LINE 46, CHANNEL 8
          DC    X'0000000000'     LINE 47, 48, 49, 50, 51
          DC    X'0A'             LINE 52, CHANNEL 10
          DC    X'0000000000'     LINE 53, 54, 55, 56, 57
          DC    X'0B'             LINE 58, CHANNEL 11
          DC    X'0000000000'     LINE 59, 60, 61, 62, 63
          DC    X'0C'             LINE 64, CHANNEL 12
          DC    X'00'             LINE 65
          DC    X'19'             LINE 66, CHANNEL 9-END OF FCB IMAGE
          END
```

Figure 49. Format of the standard STD2 FCB image

See the following information:

- "Adding an FCB Image to the Image Library" on page 461
- "Modifying an FCB Image" on page 462

Adding an FCB Image to the Image Library

You can add a 3211-format FCB image to those that reside in SYS1.IMAGELIB, using the assembler and linkage editor. No executable code is generated; the assembler prepares DCs, and the linkage editor links them into SYS1.IMAGELIB. The new FCB image must be structured according to the following rules:

1. The member name cannot exceed 8 bytes and must begin with the prefix FCB2. The characters that follow identify the FCB image and are referred to as the image identifier (ID). Any combination of valid assembler language characters can be used, with the exception of a single 'C' or 'U', because these are used by the system to recognize special conditions. To load the image into the FCB buffer, the image identifier must be specified in the FCB keyword of a DD statement or in the SETPRT macro.
2. The first byte of the FCB load module specifies whether the image is the default. (Default images are used by the system for jobs that do not request a specific image.) Specify the following in the first byte:

Value	Meaning
X'80'	Indicates a default image
X'00'	Indicates a nondefault image

3. The second byte of the load module indicates the number of bytes to be transferred to the control unit to load the FCB image. This count includes the byte, if used, for the print position indexing feature.
4. The third byte of the load module (the first byte of the FCB image) is either the print position indexing byte, or the lines-per-inch byte. The print position indexing byte is optional and, when used, precedes the lines-per-inch byte. The 3203 Model 5, 3262 Model 5, 4245, 4248, and 6262 Model 14 printers accept and discard the index byte if it is present, because they do not support the indexing feature. A description of the print position indexing feature and its use can be found in the publication *IBM 3211 Printer, 3216 Interchangeable Train Cartridge, and 3811 Printer Control Unit Component Description and Operator's Guide.*

   The special index flag in the third byte contains X'80' plus a binary index value, from 1 to 32 (the default is 1). This index value sets the left margin: 1 indicates flush-left; any other value indicates a line indented the specified number of spaces.

   The form image begins with the lines-per-inch (LPI) byte. The LPI byte defines the number of lines per inch (6 or 8) and also represents the first line of the page.

   Requirement: Printers controlled by JES2 require a channel 1 identifier here.

   Typically, the length of an FCB image is consistent with the length of the form it represents. For example, an 8½ inch form to be printed at 6 LPI has an FCB image that is 51 bytes long (8½ inches times 6 LPI).

   The LPI byte appears as follows:

Value	Meaning
X'1n'	Sets 8 LPI
X'0n'	Sets 6 LPI

5. All remaining bytes (lines) must contain X'0n', except the last byte, which must be X'1n'. The letter n can be a hexadecimal value from 1 to C, representing a channel (one to 12), or it can be 0, which means no channel is indicated.

In Figure 50 on page 462, an FCB load module is assembled and added to SYS1.IMAGELIB. The image defines a print density of 8 lines per inch on an 11-inch form, with a right shift of 15 line character positions (1½ inches).

```
//ADDFCB  JOB   MSGLEVEL=1
//STEP    EXEC  PROC=ASMHCL,PARM.ASM='NODECK,LOAD',
//              PARM.LKED='LIST,OL,REFR,RENT,XREF'
//ASM.SYSIN DD  *
FCB2ID1   CSECT
*THIS EXAMPLE IS FOR A FORM LENGTH OF 11 INCHES WITH 8 LPI (88 LINES)
          DC    X'80'     THIS IS A DEFAULT IMAGE
          DC    AL1(89)   LENGTH OF FCB IMAGE AND INDEXING BYTE
          DC    X'8F'     OFFSET 15 CHARACTERS TO THE RIGHT
          DC    X'10'     8 LINES PER INCH-NO CHANNEL FOR LINE 1
          DC    XL4'0'    4 LINES NO CHANNEL
          DC    X'01'     CHANNEL 1 IN LINE 6
          DC    XL6'0'    6 LINES NO CHANNEL
          DC    X'02'     CHANNEL 2 IN LINE 13
          DC    XL6'0'    6 LINES NO CHANNEL
          DC    X'03'     CHANNEL 3 IN LINE 20
          DC    XL6'0'    6 LINES NO CHANNEL
          DC    X'04'     CHANNEL 4 IN LINE 27
          DC    XL6'0'    6 LINES NO CHANNEL
          DC    X'05'     CHANNEL 5 IN LINE 34
          DC    XL6'0'    6 LINES NO CHANNEL
          DC    X'06'     CHANNEL 6 IN LINE 41
          DC    XL6'0'    6 LINES NO CHANNEL
          DC    X'07'     CHANNEL 7 IN LINE 48
          DC    XL6'0'    6 LINES NO CHANNEL
          DC    X'08'     CHANNEL 8 IN LINE 55
          DC    XL6'0'    6 LINES NO CHANNEL
          DC    X'09'     CHANNEL 9 IN LINE 62
          DC    XL6'0'    6 LINES NO CHANNEL
          DC    X'0A'     CHANNEL 10 IN LINE 69
          DC    XL6'0'    6 LINES NO CHANNEL
          DC    X'0B'     CHANNEL 11 IN LINE 76
          DC    XL6'0'    6 LINES NO CHANNEL
          DC    X'0C'     CHANNEL 12 IN LINE 83
          DC    XL4'0'    4 LINES NO CHANNEL
          DC    X'10'     POSITION 88 LAST LINE IN IMAGE
          END
/*
//LKED.SYSLMOD DD DSNAME=SYS1.IMAGELIB(FCB2ID1),DISP=OLD,
//              SPACE=    (OVERRIDE SECONDARY ALLOCATION)
```

Figure 50. Sample code to assemble and add an FCB load module to SYS1.IMAGELIB

For Figure 50 on page 462, be aware of the following:

- The RENT linkage editor attribute is required.
- Executing the ASMHCL procedure does not generate executable code. The assembler/linkage editor is used to place the FCB image into SYS1.IMAGELIB.
- The SPACE parameter is overridden here because the ASMHCL cataloged procedure has a secondary allocation specified. Eliminating the override causes the original secondary allocation amount to be used.

Modifying an FCB Image

To modify an FCB image in virtual storage before loading it into a forms control buffer, use the following sequence of macro instructions to read the FCB image into virtual storage.

1. An IMGLIB macro instruction, along with the OPEN parameter.
2. A BLDL macro instruction to determine if the FCB image is in the image library.
3. A LOAD macro instruction to load the image into virtual storage.
4. After the image has been read in, issue the IMGLIB macro instruction with the CLOSE. parameter and the address of the DCB built by the first IMGLIB macro.

A SETPRT macro instruction can be used to load the forms control buffer with the modified image. Printers other than the 3800 require the use of an FCB entry in an exit list, as described in *z/OS DFSMS Using Data Sets*.

The formats of the BLDL and SETPRT macros are given in *z/OS DFSMS Macro Instructions for Data Sets*; the format of the LOAD macro is given in *z/OS MVS Programming: Assembler Services Guide* and *z/OS MVS Programming: Assembler Services Reference ABE-HSP*.

The format of the IMGLIB macro is:

OPEN
Opens SYS1.IMAGELIB and specifies that a DCB is to be built for it. The address of the DCB is returned in register 1.

CLOSE
Closes SYS1.IMAGELIB.

addr
Specifies the RX-type address of a word that points to the DCB. If coded in the form (*reg*), the register in parentheses contains the address of the DCB, not the address of the word.

Return Codes from IMGLIB

Return codes from the IMGLIB macro with the OPEN parameter are as follows:

Return Code	Meaning
0 (X'00')	Operation successful.
4 (X'04')	Either the volume containing SYS1.IMAGELIB is not mounted or a required catalog volume is not mounted.
8 (X'08')	Either SYS1.IMAGELIB does not exist on the volume to which the catalog points, or SYS1.IMAGELIB is not cataloged.
12 (X'0C')	An error occurred in reading the catalog or VTOC.

BLDL and LOAD are the only macros that can refer to the DCB built by the IMGLIB macro.

JES Support for the 3211 Indexing Feature

JES2 supports the 3211 indexing feature in two ways:

- Specification of the INDEX parameter on the /*OUTPUT card.
- The extended FCB image.

The system supplies two special FCBs: FCB26 for 6 lines per inch and FCB28 for 8 lines per inch (specified as FCB=6 and FCB=8, respectively). These FCBs contain a channel 1 indication in position 1, a special index flag in the third byte, and the number of lines per inch in the fourth byte of the image.

The special index flag in the third byte of FCB26 and FCB28 contains X'80' plus a binary index value, in the range 1 to 32 (default=1). The index value sets the left margin (1 indicates flush-left; other values cause indentation of the print line by N-1 positions).

If JES2 is to use any other FCB images, the images must specify channel 1 in position 1; otherwise, JES2 positions the forms in the printer incorrectly. (Because STD1 and STD2 do not specify channel 1 in position 1, they must be modified for JES2.)

If the third byte of any other FCB image (specifying the number of lines per inch) contains a data character other than X'80', JES2 uses that specification and supplies an index value of 1.

JES3 does not support the 3211 indexing feature, and any indexing commands from JES3 are ignored by the 3203 Model 5.

Appendix C. Using the extended address volume (EAV) migration assistance tracker

The EAV migration assistance tracker can help you find programs that you might need to change if you want to support extended address volumes (EAV). The EAV migration assistance tracker is an extension of the console ID tracking facility. It helps you:

- Identify select systems services by job and program name, where the invoking programs might require analysis for changes to use new services. The program calls are identified as informational instances for possible migration actions. They are not considered errors, because the services return valid information.
- Identify possible instances of improper use of returned information in programs, like parsing 28-bit cylinder numbers in output as 16-bit cylinder numbers. These instances are identified as warnings.
- Identify instances of programs that will either fail or run with an informational message if they run on an EAV. These are identified as programs in error. The migration assistance tracker flags programs with the following functions, when the target volume of the operation is non-EAV, and the function invoked did not specify the EADSCB=OK keyword:
  - OBTAIN
  - CVAFDIR
  - CVAFSEQ
  - CVAFDSM
  - CVAFFILT
  - CVAFVSM - Note that the CVAFVSM interface is an internal system function that is not documented externally for general use.
  - OPEN of VTOC
  - DCB OPEN of an EAS eligible data set

  This allows the system programmer to identify programs in error by job and program name, without failing the programs. It also allows you to exclude programs that are not yet ready for evaluation.

  Programs identified in this phase of migration assistance tracking will continue to fail if the system service is issued for an EAV if you do not specify the EADSCB=OK keyword for them.

The EAV migration assistance tracker can be manipulated with the following commands:

- The SETGTZ operator command, which is used to activate and deactivate the tracking facility.
- The DISPLAY GTZ[,STATUS] operator command, which is used to display the current status of the tracking facility.
- The DISPLAY GTZ,TRACKDATA operator command, which is used to display any recorded instances of violations.
- The GTZPRMxx dynamic parmlib member, which is used to list violations that have already been identified in order to prevent them from being recorded again.

See The generic tracker facility in *z/OS MVS Diagnosis: Tools and Service Aids* for more information.

The following describe the instances identified and recorded by the EAV migration assistance tracker:

- “Information conventions for the EAV migration assistance tracker” on page 466
- “DFSMS instances tracked by the EAV migration assistance tracker” on page 467

Information conventions for the EAV migration assistance tracker

The information returned by the EAV migration assistance tracker describes the occurrence of an instance in text. Like the console ID tracking facility, the EAV migration assistance tracker returns tracking information and a tracking value. The tracking information can be from 1 to 28 characters in length and the system can set any EBCDIC value. The tracking value is four bytes of binary data associated with this instance. For DFSMS, these values include data to associate an instance to a specific DFSMS function and to define the reason for the instance being recorded. This standard allows for maximum flexibility in defining exclusion records that apply to DFSMS records.

The conventions for the tracking information and tracking value for a DFSMS instance follow.

Tracking information

The tracking information for a DFSMS instance might look as follows:

```
'SMS-I:3 LSPACE MSG= '
```

It can be broken down into several parts:

1. The first portion of the tracking information will be set to 'SMS-' to identify this as a DFSMS instance.
2. Appended to this is an error category, of "E" for error, "W" for warning, or "I" for informational, followed by a colon.
3. Appended to the colon is a numeric value that will identify the reason for the recorded instance. These values are
   - **1** - EAV migration: EADSCB=OK keyword was not specified on an invoking program where the target volume was non-EAV. The invoking program fails if the target volume is an EAV.
   - The invoking program would fail if the target volume was an EAV. The following section of the tracking information indicates the error that would have occurred. This instance is recorded in the tracker as an error message.
   - **2** - EAV migration. Formatted output display may contain 28-bit cylinder numbers. Program usage of these track addresses may need to be changed. Use macro TRKADDR for the comparison and manipulation of 28-bit cylinder numbers. This instance is recorded in the tracker as a warning message.
   - **3** - EAV migration. The new function is available on the invoking program. The identified program may want to exploit the available new function. This instance is recorded in the tracker as an informational message.
4. The remaining tracking information must be an EBCDIC value that describes the function executing when the tracker recorded the instance.

In the example above, the `'SMS-I:3 LSPACE MSG= '` tracking information describes a DFSMS instance as an informational instance where new function is available that the invoking program might want to exploit. The function running when the tracker recorded this instance was LSPACE MSG=.

Tracking value

For DFSMS instances, the EAV migration assistance tracker sets the tracking value as follows:

- The system sets the low order byte of the track value to the same numeric value that identifies the reason for the instance.
- The remaining high order 3-bytes are left for the function recording the tracked instance. These 3-bytes are optional. For example, these values could be set to return and reason codes or parameter list flags.

Note that the tracking value must be set to a non-zero value in order for the SETCON TRACKING=ONWITHABEND to be applicable when an instance is recorded in the tracker. If the track value is 0 or 128 no ABEND will be issued when you specify tracking ONWITHABEND.

DFSMS instances tracked by the EAV migration assistance tracker

LSPACE (SVC 78)

An LSPACE request with the DATA=, MSG=, or EXPMSG= keywords was issued. Additional data from track-managed space is available with the EXPDATA= and XEXPMSG= keywords. When this instance occurs for any volume type, it will be recorded in the tracker as an informational message.

These five keywords are mutually exclusive.

LSPACE processing (IGC0007H) will set the following tracking information:

```
TRPL_TRACK_INFO =

   'SMS-I:3 LSPACE reqtype '
    where,
       reqtype = DATA= or MSG= or EXPMSG=

TRPL_TRACK_DATA =

    Byte 0-1
       Set to zero, not used.

    Byte 2
       Set to the LSPACE parameter flag byte
          BIT 2 ON INDICATES THAT LSPACE WITH THE DATA= KEYWORD WAS
          SPECIFIED
          BIT 3 ON INDICATES THAT LSPACE WITH THE MSG= KEYWORD WAS SPECIFIED
          BIT 4 ON INDICATES THAT LSPACE WITH THE EXPMSG=KEYWORD WAS
          SPECIFIED

    Byte 3
       Set to 03.
       DFSMS Tracking category 3: EAV Migration. Informational Message. New
       function is available. Additional data from track-managed space is
       available with the EXPDATA= and XEXPMSG= keywords.

TRPL_VIOLATORS_ADDR =

    SET TO THE RESUME PSW OF THE RB THAT ISSUED SVC 78. TRACKER CODE WILL
    DETERMINE JOB AND PROGRAM NAMES FROM THIS ADDRESS.

SAMPLE OUTPUT =

15.00.00 SYSTEM1           d opdata,tracking
15.00.00 SYSTEM1           CNZ1001I 15.00.00 TRACKING DISPLAY 631
STATUS=ON       NUM=3    MAX=1000 MEM=7T  EXCL=0     REJECT=0
----TRACKING INFORMATION---- -VALUE-- JOBNAME  PROGNAME+OFF-- ASID NUM
SMS-I:3 LSPACE MSG=              5003 ALLOCAS  IEFW21SD 4CE5C   11   1
SMS-I:3 LSPACE DATA=             2003 VTDS0IS1 VTDS0IS2   118   28   2
SMS-I:3 LSPACE EXPMSG=           8803 VTDS0IS1 VTDS0IS2   118   28   2
SMS-I:3 LSPACE MSG=              9003 *MASTER* IEE70110  52F6   01  46
------------------------------------------------------------------------

SAMPLE EXCLUSION LIST =

*                            Jobname  Pgmname                        *
* Tracking Information Mask   Mask     Mask     Comments (ignored)    *
*-----------------------------+--------+--------+----------------------+
|SMS-I:3 LSPACE*              |*MASTER*|IEE70110| I SMF CALLS TO LSPACE|
|SMS-I:3 LSPACE*              |ALLOCAS |IEFW21SD| VARY DEVICE OFFLINE  |
|SMS-I:3 LSPACE*              |*       |VTDSOIS2| VTDSOIS2 PROG CALLS  |
|SMS-I:3 LSPACE*              |VTDSOIS1|*       | VTDSOIS1 JOB CALLS   |
```

DEVTYPE (SVC 24)

A DEVTYPE request with DEVTAB or UCBLIST without INFOLIST, returns the number of cylinders on the volume. This is in a two-byte field at offset 8, which is too small if the volume has more than 65 520 cylinders. Consider using INFO=DASD which returns the number of cylinders in a four-byte field. When INFO=DASD is specified additional fields are now provided. They include (see mapping macro IHADVA):

```
DVAIXVLD BIT           DVACYLMG, DVAEADSCB, DVAVIRSZ valid
DVACYLMG BIT             Cylinder-managed space exists on
*                        this volume and begins at DVALCYL
```

```
*                        in multicylinder units of DVAMCU.
*                        DVAEADSCB is also set with this
*                        flag on. Valid when DVAIXVLD is set.
DVAEADSCB BIT            Extended attribute DSCBs, Format 8
*                        and 9 DSCBs, are allowed on this
*                        volume.  Valid when DVAIXVLD is set.
DVAMCU 8-BIT UNSIGNED INTEGER
*                        Minimum allocation size in
*                        cylinders for cylinder-managed
*                        space.  Each extent in this space
*                        must be a multiple of this value.
*                        space.  Also referred to as the
*                        multicylinder unit (MCU). This is
*                        the smallest unit of disk space in
*                        cylinders that can be allocated
*                        in cylinder-managed space.
*                        Valid when DVACYLMG is set.
*                        This field is zero on releases
*                        before z/OS 1.10 or if the status
*                        is not yet known.  In these two
*                        cases DVAIXVLD is not set.
DVALCYL 16-BIT UNSIGNED INTEGER
*                        First cylinder address divided by
*                        4095 where space is managed in
*                        multicylinder units. Cyl-managed
*                        space begins at this address.
*                        Valid when DVACYLMG is set. This
*                        field is zero on releases before
*                        z/OS 1.10 or if the status is not
*                        yet known.  In these two cases
*                        DVAIXVLD is not set.
DVAVIRSZ 16-BIT UNSIGNED INTEGER
*                        Block size of the index data set.
*                        Valid when DAVIXVLD is set on.
*                        When valid and zero the volume
*                        has no working VTOC index. This
*                        field is zero on releases before
*                        z/OS 1.10 or if the status is not
*                        yet known.  In these cases
*                        DVAIXVLD is not set.
```

When this instance occurs for any volume type, it will be recorded in the tracker as an informational message.

DEVTYPE processing (IGC0002D) will set the following tracking information:

```
TRPL_TRACK_INFO =

    ‘SMS-I:3 DEVTYPE ‘

TRPL_TRACK_DATA =

    Byte 0-2
       Set to zero, not used.

    Byte 3
       Set to 03.

       DFSMS Tracking category 3: EAV Migration. Informational Message. New
       function is available. Additional data from DEVTYPE INFO=DASD
       invocation is available. See mapping macro IHADVA.

TRPL_VIOLATORS_ADDR =
    SET TO THE RESUME PSW OF THE RB THAT ISSUED SVC 24. TRACKER CODE WILL
    DETERMINE JOB AND PROGRAM NAMES FROM THIS ADDRESS.

SAMPLE OUTPUT =

15.00.00 SYSTEM1           d opdata,tracking
15.00.00 SYSTEM1           CNZ1001I 15.00.00 TRACKING DISPLAY 631
STATUS=ON       NUM=1    MAX=1000 MEM=7T  EXCL=0     REJECT=0
----TRACKING INFORMATION---- -VALUE-- JOBNAME  PROGNAME+OFF-- ASID NUM
SMS-I:3 DEVTYPE                    03 DEVTJOB  DEVTPROG 4CE5C   11   1
------------------------------------------------------------------------


SAMPLE EXCLUSION LIST =

*                            Jobname  Pgmname                        *
* Tracking Information Mask   Mask     Mask     Comments (ignored)    *
```

```
*----------------------------+--------+--------+----------------------+
|SMS*DEVTYPE*                |*       |*       | ALL DEVTYPE          |
```

IDCAMS LISTDATA PINNED

An IDCAMS LISTDATA PINNED request was processed. The track addresses for the PINNED tracks may contain 28-bit cylinder numbers.

When this instance occurs for any volume type, it will be recorded in the tracker as a warning message.

IDCAMS LISTDATA PINNED processing (IDCSS05) will set the following tracking information:

```
TRPL_TRACK_INFO =

    'SMS-W:2 IDCAMS LISTDATA PINN'


TRPL_TRACK_DATA =

      Byte 0-2
              Set to zero, not used.

      Byte 3
               Set to 02.

            DFSMS Tracking category 2: EAV Migration. Warning Message.
            An IDCAMS LISTDATA PINNED request was processed.  The track addresses for the
            PINNED tracks may contain 28-bit cylinder numbers.
```

```
TRPL_VIOLATORS_ADDR =

SET TO THE RESUME PSW OF THE PRB WITH A VALID POINTER (RBCDE1) TO
THE 'IDCAMS' CDNAME.  USE THE PREVIOUS RB (RBLINKB) IF THIS PRB
IS NOT THE FIRST RB.  TRACKER CODE WILL DETERMINE JOB AND PROGRAM
NAMES FROM THIS ADDRESS.

SAMPLE OUTPUT =


07.34.08 SYSTEM1           d opdata,tracking
07.34.08 SYSTEM1           CNZ1001I 07.34.08 TRACKING DISPLAY 673
STATUS=ON,ABEND NUM=1    MAX=1000 MEM=7T  EXCL=5     REJECT=0
----TRACKING INFORMATION---- -VALUE-- JOBNAME  PROGNAME+OFF-- ASID NUM
SMS-W:2 IDCAMS LISTDATA PINN       02 LISTDATA IDCAMS    E48E   28   4



SAMPLE EXCLUSION LIST =

*                            Jobname  Pgmname                       *
* Tracking Information Mask   Mask     Mask     Comments (ignored)    *
*----------------------------+--------+--------+----------------------+
|SMS*LISTDATA PINNED*        |*       |IDCAMS  | All IDCAMS PGM CALLS |
```

IEHLIST LISTVTOC

An IEHLIST LISTVTOC request was processed. Extent descriptors may contain cylinder addresses 65520 or larger. Free space descriptors may contain track addresses 982800 or larger and/or full cylinders 65520 or larger. The generated report will display the information in different columns as compared to reports generated on releases prior to z/OS V1.10.

When this instance occurs for any volume type, it will be recorded in the tracker as a warning message.

IEHLIST LISTVTOC processing will set the following tracking information:

```
TRPL_TRACK_INFO =

'SMS-W:2 IEHLIST LISTVTOC   '


TRPL_TRACK_DATA =

      Byte 0-2

        Set to zero, not used.

      Byte 3

      Set to 02.
```

```
      DFSMS Tracking category 2: EAV Migration. Warning Message.
      An IEHLIST LISTVTOC request was processed. Extent descriptors may
      contain cylinder addresses 65520 or larger.  Free space descriptors
      may contain track addresses 982800 or larger and/or full cylinders
      65520 or larger.   The generated report will display the information
      in different columns as compared to reports generated on releases
      prior to z/OS V1.10.

TRPL_VIOLATORS_ADDR =

SET TO THE RESUME PSW OF THE PRB WITH A VALID POINTER (RBCDE1) TO THE 'IEHLIST' CDNAME.
USE THE PREVIOUS RB (RBLINKB) IF THIS PRB IS NOT THE FIRST RB.  TRACKER CODE WILL DETERMINE
JOB AND PROGRAM NAMES FROM THIS ADDRESS.

SAMPLE OUTPUT =


07.34.08 SYSTEM1           d opdata,tracking
07.34.08 SYSTEM1           CNZ1001I 07.34.08 TRACKING DISPLAY 673
STATUS=ON,ABEND NUM=1    MAX=1000 MEM=7T  EXCL=5     REJECT=0
----TRACKING INFORMATION---- -VALUE-- JOBNAME  PROGNAME+OFF-- ASID NUM
SMS-W:2 IEHLIST LISTVTOC           02 LNKLST   LNKLST      2A   29   2
SMS-W:2 IEHLIST LISTVTOC           02 LST004   IEHLIST   2304   29   1
SMS-W:2 IEHLIST LISTVTOC           02 LST004   IEHLIST   34A6   29   1




SAMPLE EXCLUSION LIST =

*                            Jobname  Pgmname                        *
* Tracking Information Mask   Mask     Mask     Comments (ignored)    *
*--------------------------+--------+--------+---------------------+
| SMS*IEHLIST LISTVTOC       |*       |IEHLIST | IEHLIST PGM CALLS    |
```

IDCAMS DCOLLECT

An IDCAMS DCOLLECT request for 'V' (Volume Record Field) and 'VL' (SMS Volume Definition Field) records was processed. Additional data for track-managed space was recorded.

When this instance occurs for any volume type, it will be recorded in the tracker an informational message.

IDCAMS DCOLLECT processing will set the following tracking information:

```
TRPL_TRACK_INFO =

'SMS-I:3 IDCAMS DCOLLECT      '


TRPL_TRACK_DATA =

      Byte 0-2

        Set to zero, not used.

      Byte 3
            Set to 03.

         DFSMS Tracking category 3: EAV Migration. Informational Message. An
         IDCAMS DCOLLECT request for 'V' (Volume Record Field) and 'VL' (SMS
         Volume Definition Field) records was processed. Additional data
         for track-managed space was recorded.

TRPL_VIOLATORS_ADDR =

    SET TO THE RESUME PSW OF THE PRB WITH A VALID POINTER (RBCDE1) TO THE
    'IDCAMS' CDNAME.  USE THE PREVIOUS RB (RBLINKB) IF THIS PRB IS NOT THE
    FIRST RB.  TRACKER CODE WILL DETERMINE JOB AND PROGRAM NAMES FROM THIS
    ADDRESS.

SAMPLE OUTPUT =

07.34.08 SYSTEM1           d opdata,tracking
07.34.08 SYSTEM1           CNZ1001I 07.34.08 TRACKING DISPLAY 673
STATUS=ON,ABEND NUM=1    MAX=1000 MEM=7T  EXCL=5     REJECT=0
```

```
----TRACKING INFORMATION---- -VALUE-- JOBNAME  PROGNAME+OFF-- ASID NUM
SMS-I:3 IDCAMS DCOLLECT            03 DCOLLECT IDCAMS     xxx   28   4

SAMPLE EXCLUSION LIST =

*                             Jobname  Pgmname                        *
* Tracking Information Mask   Mask     Mask     Comments (ignored)    *
*-----------------------------+--------+--------+---------------------+
|SMS*DCOLLECT*               |*       |IDCAMS  | IDCAMS DCOLLECT PGM  |
```

IDCAMS LISTCAT

An IDCAMS LISTCAT request was processed that printed extent descriptors for one or more EAS eligible data set (VSAM in z/OS V1R10). The returned extent descriptors may contain 28-bit cylinder numbers.

When this instance occurs for any volume type, it will be recorded in the tracker as a warning message.

IDCAMS LISTCAT processing will set the following tracking information:

```
TRPL_TRACK_INFO =
      'SMS-W:2 IDCAMS LISTCAT   '


TRPL_TRACK_DATA =

      Byte 0-2

       Set to zero, not used.

      Byte 3

       Set to 02.

      DFSMS Tracking category 2: EAV Migration.  Warning Message.
      An IDCAMS LISTCAT request was processed that printed extent
      descriptors for one or more EAS eligible data set (VSAM in z/OS V1R10).
      The returned extent descriptors may contain 28-bit cylinder numbers.
      This instance will be recorded for both EAS and non-EAS capable volumes.
      Please note thatAMS Listcat output format may change as a result of
      service and new function support. IBM recommends applications processing
      LISTCAT output be updated to obtain results directly from the Catalog
      Search Interface (CSI). For more information on CSI, see
      z/OS DFSMS Managing Catalogs and HLASM Programmer's Guide.


TRPL_VIOLATORS_ADDR =

      SET TO THE RESUME PSW OF THE PRB WITH A VALID POINTER (RBCDE1) TO THE
      'IDCAMS' CDNAME.  USE THE PREVIOUS RB (RBLINKB) IF THIS PRB IS NOT THE
       FIRST RB.  TRACKER CODE WILL DETERMINE JOB AND PROGRAM NAMES FROM THIS
       ADDRESS.

SAMPLE OUTPUT =


07.34.08 SYSTEM1           d opdata,tracking
07.34.08 SYSTEM1           CNZ1001I 07.34.08 TRACKING DISPLAY 673
STATUS=ON,ABEND NUM=1    MAX=1000 MEM=7T  EXCL=5     REJECT=0
----TRACKING INFORMATION---- -VALUE-- JOBNAME  PROGNAME+OFF-- ASID NUM
SMS-W:2 IDCAMS LISTCAT             02 LISTCAT  IDCAMS     xxx   28   4

SAMPLE EXCLUSION LIST =

*                             Jobname  Pgmname                        *
* Tracking Information Mask   Mask     Mask     Comments (ignored)    *
*-----------------------------+--------+--------+---------------------+
|SMS*IDCAMS LISTCAT          |*       |IDCAMS  | IDCAMS LISTCAT PGM   |
```

OBTAIN (SVC 27)

OBTAIN was issued with the search or seek option to a non-EAV volume. The caller did not specify with EADSCB=OK That it supports the extended attribute DSCBs and the target data set is EAS eligible.

When this instance occurs for a non-EAV volume type, it will be recorded in the tracker as an error message.

OBTAIN processing will set the following tracking information:

```
TRPL_TRACK_INFO =

      'SMS-E:1 DADSM OBTAIN       '


TRPL_TRACK_DATA =

      Byte 0

        Set to zero, not used.

       Byte 1-2

      Operation code.
                X'C100' SEARCH for DSNAME.
                  X'C080' SEEK for track address.

      Byte 3

      Set to 01.

      DFSMS Tracking category 1: EAV Migration. Error Message. DADSM
      OBTAIN was issued with the search or seek option to a non-EAV
      volume. The caller did not specify with EADSCB=OK That it supports
      the extended attribute DSCBs and the target data set is 'EAS
      eligible'.

TRPL_VIOLATORS_ADDR =

      SET TO THE RESUME PSW OF THE RB THAT ISSUED SVC 27.  TRACKER CODE WILL
      DETERMINE JOB AND PROGRAM NAMES FROM THIS ADDRESS.

SAMPLE OUTPUT =


07.34.08 SYSTEM1           d opdata,tracking
07.34.08 SYSTEM1           CNZ1001I 07.34.08 TRACKING DISPLAY 673
STATUS=ON,ABEND NUM=1    MAX=1000 MEM=7T  EXCL=5     REJECT=0
----TRACKING INFORMATION---- -VALUE-- JOBNAME  PROGNAME+OFF-- ASID NUM
SMS-E:1 DADSM OBTAIN           C08001 OBTJBN   OBTPGM     xxx   28   4

SAMPLE EXCLUSION LIST =

*                             Jobname  Pgmname                        *
* Tracking Information Mask   Mask     Mask     Comments (ignored)    *
*-----------------------------+--------+--------+---------------------+
|SMS*OBTAIN*                 |*       |*       | DADSM OBTAIN         |
```

CVAFDIR

CVAFDIR was issued with the search or seek option to a volume that does not support extended attribute DSCBs. The caller did not specify with EADSCB=OK That it supports the extended attribute DSCBs and the target data set is 'EAS eligible'. CVAF return code 4 and CVSTAT of X'52' would have been set if issued to a volume that supports extended attribute DSCBs.

When this instance occurs for a volume that does not support extended attribute DSCBs, it will be recorded in the tracker as an error message.

CVAFDIR processing will set the following tracking information:

```
TRPL_TRACK_INFO =

      'SMS-E:1 CVAFDIR STAT082     '


TRPL_TRACK_DATA =

      Byte 0

        Set to zero, not used.

       Byte 1

       CVAF Return Code = 4
```

```
      Byte 2

       CVAF Status Code = STAT082 (X'52')

      Byte 3

       Set to 01.

       DFSMS Tracking category 1: EAV Migration. Error Message. CVAFDIR was
       issued with the search or seek option to a volume that does not
       support extended attribute DSCBs. The caller did not specify with
       EADSCB=OK That it supports the extended attribute DSCBs and the
       target data set is 'EAS eligible'.  CVAF return code 4 and CVSTAT of
       X'52' would have been set if issued to a device that supports
       extended attribute DSCBs.

TRPL_VIOLATORS_ADDR =

    FOR BRANCH ENTRY CALLERS, SET TO THE CALLER OF CVAF RETURN ADDRESS.
    TRACKER CODE WILL DETERMINE JOB AND PROGRAM NAMES FROM THIS ADDRESS.

    FOR SVC CALLS, SET TO THE RESUME PSW OF THE RB THAT ISSUED SVC 139.
    TRACKER CODE WILL DETERMINE JOB AND PROGRAM NAMES FROM THIS ADDRESS.

SAMPLE OUTPUT =


07.34.08 SYSTEM1           d opdata,tracking
07.34.08 SYSTEM1           CNZ1001I 07.34.08 TRACKING DISPLAY 673
STATUS=ON,ABEND NUM=1    MAX=1000 MEM=7T  EXCL=5     REJECT=0
----TRACKING INFORMATION---- -VALUE-- JOBNAME  PROGNAME+OFF-- ASID NUM
SMS-E:1 CVAFDIR STAT082        045201 CVAFJBN  CVAFPGM    xxx   28   4

SAMPLE EXCLUSION LIST =

*                             Jobname  Pgmname                        *
* Tracking Information Mask   Mask     Mask     Comments (ignored)    *
*-----------------------------+--------+--------+---------------------+
|SMS*CVAFDIR STAT082*        |*       |*       | CVAFDIR              |
```

CVAFSEQ

CVAFSEQ was issued for physical sequential or index order to a volume that does not support extended attribute DSCBs. The caller did not specify with EADSCB=OK That it supports the extended attribute DSCBs and the target data set is 'EAS eligible'. CVAF return code 4 and CVSTAT of X'52' would have been set if issued to a volume that supports extended attribute DSCBs.

When this instance occurs for a volume that does not support extended attribute DSCBs, it will be recorded in the tracker as an error message.

CVAFSEQ processing will set the following tracking information:

```
TRPL_TRACK_INFO =

      'SMS-E:1 CVAFSEQ STAT082    '

TRPL_TRACK_DATA =

      Byte 0

         Set to zero, not used.

       Byte 1

        CVAF Return Code = 4

      Byte 2

        CVAF Status Code = STAT082 (X'52')

      Byte 3

        Set to 01.

        DFSMS Tracking category 1: EAV Migration. Error Message. CVAFSEQ was
        issued for physical sequential or index order to a volume that does
```

```
        not support extended attribute DSCBs.   The caller did not specify
        with EADSCB=OK That it supports the extended attribute DSCBs and the
        target data set is 'EAS eligible'. CVAF return code 4 and CVSTAT of
         X'52' would have been set if issued to a volume that supports
        extended attribute DSCBs.

TRPL_VIOLATORS_ADDR =

      FOR BRANCH ENTRY CALLERS, SET TO THE CALLER OF CVAF RETURN ADDRESS.
      TRACKER CODE WILL DETERMINE JOB AND PROGRAM NAMES FROM THIS ADDRESS.

      FOR SVC CALLS, SET TO THE RESUME PSW OF THE RB THAT ISSUED SVC 139.
      TRACKER CODE WILL DETERMINE JOB AND PROGRAM NAMES FROM THIS ADDRESS.

SAMPLE OUTPUT =


07.34.08 SYSTEM1           d opdata,tracking
07.34.08 SYSTEM1           CNZ1001I 07.34.08 TRACKING DISPLAY 673
STATUS=ON,ABEND NUM=1    MAX=1000 MEM=7T  EXCL=5     REJECT=0
----TRACKING INFORMATION---- -VALUE-- JOBNAME  PROGNAME+OFF-- ASID NUM
SMS-E:1 CVAFSEQ STAT082        045201 CVAFJBN  CVAFPGM    xxx   28   4


SAMPLE EXCLUSION LIST =

*                             Jobname  Pgmname                        *
* Tracking Information Mask   Mask     Mask     Comments (ignored)    *
*-----------------------------+--------+--------+---------------------+
|SMS*CVAFSEQ STAT082*        |*       |*       | CVAFSEQ              |
```

CVAFDSM

CVAFDSM was issued to retrieve unallocated space on a volume that does not support extended attribute DSCBs. The caller did not specify with EADSCB=OK That it supports the extended attribute DSCBs. CVAF return code 4 and CVSTAT of X'52' would have been set if issued to a volume that supports extended attribute DSCBs.

When this instance occurs for a volume that does not support extended attribute DSCBs, it will be recorded in the tracker as an error message.

CVAFDSM processing will set the following tracking information:

```
TRPL_TRACK_INFO =

      'SMS-E:1 CVAFDSM STAT082     '


TRPL_TRACK_DATA =

      Byte 0

        Set to zero, not used.

      Byte 1

       CVAF Return Code = 4

      Byte 2

       CVAF Status Code = STAT082 (X'52')

      Byte 3

       Set to 01.

       DFSMS Tracking category 1: EAV Migration. Error Message. CVAFDSM was
       issued to retrieve unallocated space on a volume (CVAFDSM
       ACCESS=MAPDATA, MAP=VOLUME, RTA4BYTE=YES) that does not support
       extended attribute DSCBs.   The caller did not specify with
       EADSCB=OK That it supports the extended attribute DSCBs.  CVAF
       return code 4 and CVSTAT         of X'52' would have been set if issued to a
       volume that supports extended attribute DSCBs.
       CVAFVSM interface is an internal system function that is not documented
       externally for general use.

TRPL_VIOLATORS_ADDR =
```

```
      FOR BRANCH ENTRY CALLERS, SET TO THE CALLER OF CVAF RETURN ADDRESS.
      TRACKER CODE WILL DETERMINE JOB AND PROGRAM NAMES FROM THIS ADDRESS.

      FOR SVC CALLS, SET TO THE RESUME PSW OF THE RB THAT ISSUED SVC 139.
      TRACKER CODE WILL DETERMINE JOB AND PROGRAM NAMES FROM THIS ADDRESS.

SAMPLE OUTPUT =


07.34.08 SYSTEM1           d opdata,tracking
07.34.08 SYSTEM1           CNZ1001I 07.34.08 TRACKING DISPLAY 673
STATUS=ON,ABEND NUM=1    MAX=1000 MEM=7T  EXCL=5     REJECT=0
----TRACKING INFORMATION---- -VALUE-- JOBNAME  PROGNAME+OFF-- ASID NUM
SMS-E:1 CVAFDSM STAT082        045201 CVAFJBN  CVAFPGM    xxx   28   4


SAMPLE EXCLUSION LIST =

*                             Jobname  Pgmname                        *
* Tracking Information Mask   Mask     Mask     Comments (ignored)    *
*---------------------------+--------+--------+---------------------+
|SMS*CVAFDSM STAT082*        |*       |*       | CVAFDSM              |
```

CVAFFILT

CVAFFILT was issued to obtain DSCB information for fully or partially qualified data set names on a volume that does not support extended attribute DSCBs. The caller did not specify with EADSCB=OK That it supports the extended attribute DSCBs and the qualified data set is ‘EAS eligible’. CVAF return code 4 and CVSTAT of X’56’ along with data set name status in the FCL (FCLDSNST) of X’06’ would have been set if the request was issued to a volume that supports extended attribute DSCBs.

When this instance occurs for a volume that does not support extended attribute DSCBs, it will be recorded in the tracker as an error message.

CVAFFILT processing will set the following tracking information:

```
TRPL_TRACK_INFO =

    ‘SMS-E:1 CVAFFILT STAT086        ‘


TRPL_TRACK_DATA =


       Byte 0

         CVAF Return Code = 4

      Byte 1

        CVAF Status Code = STAT086 (X’56’)

      Byte 2

        FCL data set status code = X’06’


      Byte 3

        Set to 01.

        DFSMS Tracking category 1: EAV Migration. Error Message. CVAFFILT
        was issued to obtain DSCB information for fully or partially
        qualified data set names on a volume that does not support extended
        attribute DSCBs.   The caller did not specify with EADSCB=OK that is
        supports  the extended attribute DSCBs and the qualified data set is
        ‘EAS eligible’.   CVAF return code 4 and CVSTAT of X’56’ along with
        data set name status in the FCL (FCLDSNST) of X’06’ would have been
        set if the request was issued to a volume that supports extended
        attribute DSCBs.


TRPL_VIOLATORS_ADDR =

      FOR BRANCH ENTRY CALLERS, SET TO THE CALLER OF CVAF RETURN ADDRESS.
      TRACKER CODE WILL DETERMINE JOB AND PROGRAM NAMES FROM THIS ADDRESS.
```

```
      FOR SVC CALLS, SET TO THE RESUME PSW OF THE RB THAT ISSUED SVC 139.
      TRACKER CODE WILL DETERMINE JOB AND PROGRAM NAMES FROM THIS ADDRESS.

SAMPLE OUTPUT =


07.34.08 SYSTEM1           d opdata,tracking
07.34.08 SYSTEM1           CNZ1001I 07.34.08 TRACKING DISPLAY 673
STATUS=ON,ABEND NUM=1    MAX=1000 MEM=7T  EXCL=5     REJECT=0
----TRACKING INFORMATION---- -VALUE-- JOBNAME  PROGNAME+OFF-- ASID NUM
SMS-E:1 CVAFFILT STAT086     04560601 CVAFJBN  CVAFPGM    xxx   28   4



SAMPLE EXCLUSION LIST =

*                             Jobname  Pgmname                       *
* Tracking Information Mask   Mask     Mask     Comments (ignored)    *
*---------------------------+--------+--------+---------------------+
|SMS*CVAFFILT STAT086*       |*       |*       | CVAFFILT             |
```

CVAFVSM

CVAFVSM was issued to allocate space for a volume that is not an EAV. The caller did not specify with EADSCB=OK That it supports an EAV. CVAF return code 4 and CVSTAT of X'52' would have been set if issued to an EAV.

When this instance occurs for a non EAV, it will be recorded in the tracker as an error message.

Note that the CVAFVSM interface is an internal system function that is not documented externally for general use.

CVAFVSM processing will set the following tracking information:

```
TRPL_TRACK_INFO =

      'SMS-E:1 CVAFVSM STAT082        '


TRPL_TRACK_DATA =

      Byte 0

           Set to zero, not used.

       Byte 1

        CVAF Return Code = 4

      Byte 2

        CVAF Status Code = STAT082 (X'52')

      Byte 3

        Set to 01.

        DFSMS Tracking category 1: EAV Migration. Error message. CVAFVSM was
        issued to allocate space for a volume that is not an EAV. The
        caller did not specify with EADSCB=OK That it supports an EAV. CVAF
        return code 4 and CVSTAT of X'52' would have been set if issued to
        an EAV. CVAFVSM interface is an internal system function that is not
        documented externally for general use.

        When this instance occurs for a non EAV, it will be recorded in the
        tracker as an error message.

TRPL_VIOLATORS_ADDR =

      FOR BRANCH ENTRY CALLERS, SET TO THE CALLER OF CVAF RETURN ADDRESS.
      TRACKER CODE WILL DETERMINE JOB AND PROGRAM NAMES FROM THIS ADDRESS.

      FOR SVC CALLS, SET TO THE RESUME PSW OF THE RB THAT ISSUED SVC 139.
      TRACKER CODE WILL DETERMINE JOB AND PROGRAM NAMES FROM THIS
```

```
SAMPLE OUTPUT =


07.34.08 SYSTEM1           d opdata,tracking
07.34.08 SYSTEM1           CNZ1001I 07.34.08 TRACKING DISPLAY 673
STATUS=ON,ABEND NUM=1    MAX=1000 MEM=7T  EXCL=5     REJECT=0
----TRACKING INFORMATION---- -VALUE-- JOBNAME  PROGNAME+OFF-- ASID NUM
SMS-E:1 CVAFVSM STAT082        045201 CVAFJBN  CVAFPGM    xxx   28   4



SAMPLE EXCLUSION LIST =

*                             Jobname  Pgmname                        *
* Tracking Information Mask   Mask     Mask     Comments (ignored)    *
*---------------------------+--------+--------+---------------------+
|SMS*CVAFVSM STAT085*        |*       |*       | CVAFVSM              |
```

DCB Open of a VTOC

A DCB Open of a VTOC was issued to a volume that does not support extended attribute DSCBs. The caller did not specify EADSCB=OK on the DCBE macro indicating that it supports the extended attribute DSCBs in the VTOC. Open would have issued an ABEND, MSGIEC142I 113-48 if an attempt was made to open the VTOC of a volume that supported extended attribute DSCBs.

When this instance occurs for a volume that does not support extended attribute DSCBs, it will be recorded in the tracker as an error message.

OPEN processing will set the following tracking information:

```
TRPL_TRACK_INFO =

      'SMS-E:1 DCB OPEN VTOC 113-48'


TRPL_TRACK_DATA =


      Byte 0-2

         Set to zero, not used.


      Byte 3

        Set to 01.

        DFSMS Tracking category 1: EAV Migration. Error Message. A DCB Open
        of a VTOC was issued to a volume that does not support extended
        attribute DSCBs.   The caller did not specify EADSCB=OK on the
        DCBE macro indicating that it supports the extended attribute DSCBs in the
        VTOC.  Open would have issued an ABEND, MSGIEC142I 113-48 if an
        attempt was made to open the VTOC of a volume that supported
        extended attribute DSCBs.

TRPL_VIOLATORS_ADDR =

        SET TO THE RESUME PSW OF THE RB THAT ISSUED THE OPEN SVC.  TRACKER CODE
        WILL DETERMINE JOB AND PROGRAM NAMES FROM THIS ADDRESS.

SAMPLE OUTPUT =


07.34.08 SYSTEM1           d opdata,tracking
07.34.08 SYSTEM1           CNZ1001I 07.34.08 TRACKING DISPLAY 673
STATUS=ON,ABEND NUM=1    MAX=1000 MEM=7T  EXCL=5     REJECT=0
----TRACKING INFORMATION---- -VALUE-- JOBNAME  PROGNAME+OFF-- ASID NUM
SMS-E:1 DCB OPEN VTOC 113-48       01 OPENJBN  OPENPGM    xxx   28   4



SAMPLE EXCLUSION LIST =

*                             Jobname  Pgmname                        *
* Tracking Information Mask   Mask     Mask     Comments (ignored)    *
```

```
*----------------------------+--------+--------+----------------------+
|SMS*DCB OPEN VTOC*          |*       |*       | MSGIEC142I 113-48    |
```

DCB Open of EAS eligible data set

A DCB Open (MACRF = E for EXCP) of an EAS eligible data set was issued to a volume that does not support extended attribute DSCBs. The caller did not specify with EADSCB=OK on the DCBE macro to indicate that it supports the extended attribute DSCBs for an EAS eligible data set. Open would have issued an ABEND, MSGIEC142I 113-44 if an attempt was made to open the EAS eligible data set on a volume that supported extended attribute DSCBs.

When this instance occurs for a volume that does not support extended attribute DSCBs, it will be recorded in the tracker as an error message.

OPEN processing will set the following tracking information:

```
TRPL_TRACK_INFO =


      'SMS-E:1 DCB OPEN EAS 113-44'


TRPL_TRACK_DATA =

      Byte 0-2

         Set to zero, not used.

      Byte 3

        Set to 01.

        DFSMS Tracking category 1: EAV Migration. Error Message.
        A DCB Open (MACRF = E for EXCP) of an EAS eligible data
        set was issued to a volume that does not support extended
        attribute DSCBs. The caller did not specify with EADSCB=OK on the
        DCBE macro to indicate that it supports the extended attribute
        DSCBs for an EAS eligible data set. Open would have issued
        an ABEND, MSGIEC142I 113-44 if an attempt was made to open the
        EAS eligible data set on a volume that supported extended attribute
        DSCBs.

TRPL_VIOLATORS_ADDR =

      SET TO THE RESUME PSW OF THE RB THAT ISSUED THE OPEN SVC.  TRACKER CODE
      WILL DETERMINE JOB AND PROGRAM NAMES FROM THIS ADDRESS.

SAMPLE OUTPUT =


07.34.08 SYSTEM1           d opdata,tracking
07.34.08 SYSTEM1           CNZ1001I 07.34.08 TRACKING DISPLAY 673
STATUS=ON,ABEND NUM=1    MAX=1000 MEM=7T  EXCL=5     REJECT=0
----TRACKING INFORMATION---- -VALUE-- JOBNAME  PROGNAME+OFF-- ASID NUM
SMS-E:1 DCB OPEN EAS 113-44        01 OPENJBN  OPENPGM    xxx   28   4



SAMPLE EXCLUSION LIST =

*                            Jobname  Pgmname                        *
* Tracking Information Mask   Mask     Mask     Comments (ignored)    *
*----------------------------+--------+--------+----------------------+
|SMS*DCB OPEN VSAM*          |*       |*       | MSGIEC142I 113-44    |
```

Other Sample exclusion list

```
SAMPLE EXCLUSION LIST =

*                            Jobname  Pgmname                        *
* Tracking Information Mask   Mask     Mask     Comments (ignored)    *
*----------------------------+--------+--------+----------------------+
|SMS*                        |*       |*       | SMS all instances    |
|SMS*E*                      |*       |*       | SMS Errors           |
```

```
|SMS*W*                      |*       |*       | SMS Warnings          |
|SMS*I*                      |*       |*       | SMS Informational     |
|SMS*:1*                     |*       |*       | EAV EADSCB=OK missing|
|SMS*:2*                     |*       |*       | EAV 28-bitcyls output|
|SMS*:3*                     |*       |*       | EAV new function      |
```

Recommend exclusion list

For the recommended exclusion lists for DFSMS, see z/OS downloads (www.ibm.com/systems/z/os/zos/downloads).

Summary of DFSMS instances

```
07.34.08 SYSTEM1           d opdata,tracking
07.34.08 SYSTEM1           CNZ1001I 07.34.08 TRACKING DISPLAY 673
STATUS=ON,ABEND NUM=1    MAX=1000 MEM=7T  EXCL=5     REJECT=0
----TRACKING INFORMATION---- -VALUE-- JOBNAME  PROGNAME+OFF-- ASID NUM
SMS-E:1 CVAFDIR STAT082        045201 CVAFJBN  CVAFPGM    123   28   4
SMS-E:1 CVAFDSM STAT082        045201 CVAFJBN  CVAFPGM    456   28   4
SMS-E:1 CVAFFILT STAT086     04560601 CVAFJBN  CVAFPGM    789   28   1
SMS-E:1 CVAFSEQ STAT082        045201 CVAFJBN  CVAFPGM    123   28   4
SMS-E:1 CVAFVSM STAT082        045201 CVAFJBN  CVAFPGM    456   28   4
SMS-E:1 DADSM OBTAIN           C08001 OBTJBN   OBTPGM     789   28   4
SMS-E:1 DCB OPEN EAS  113-44       01 OPENJBN  OPENPGM    123   28   1
SMS-E:1 DCB OPEN VTOC 113-48       01 OPENJBN  OPENPGM    456   28   1
SMS-I:3 DEVTYPE                    03 DEVTJOB  DEVTPROG 4CE5C   11   1
SMS-I:3 LSPACE MSG=              5003 ALLOCAS  IEFW21SD 4CE5C   11   1
SMS-I:3 LSPACE DATA=             2003 VTDS0IS1 VTDS0IS2   118   28   2
SMS-I:3 LSPACE EXPMSG=           8803 VTDS0IS1 VTDS0IS2   118   28   2
SMS-I:3 IDCAMS DCOLLECT            03 DCOLLECT IDCAMS     123   28   4
SMS-W:2 IDCAMS LISTCAT             02 LISTCAT  IDCAMS     456   28   4
SMS-W:2 IDCAMS LISTDATA PINN       02 LISTDATA IDCAMS    E48E   28   4
SMS-W:2 IEHLIST LISTVTOC           02 LISTVTOC IEHLIST    123   28   4
```

Appendix D. 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.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and Trademark information (www.ibm.com/legal/copytrade.shtml).

Index

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Z

Product Number: 5650-ZOS

SC23-6861-50