z/OS

DFSMShsm Diagnosis

Version 2 Release 2

GC52-1387-01

Note
Before using this information and the product it supports, read the information in "Notices" on page 141.

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

Contents

Figures

Tables

About this document

This document helps you diagnose IBM® z/OS® DFSMShsm problems and, if needed, report them to the IBM Support Center.

For information about the accessibility features of z/OS, for users who have a physical disability, see "Accessibility," on page 137.

Who should read this document

This document is intended for system programmers and storage administrators responsible for diagnosing DFSMShsm errors and, if needed, reporting them to the IBM Support Center.

Major divisions of this document

This document is divided into chapters, as follows:

- **Chapter 1, "Introduction," on page 1** provides an overview of the steps used to diagnose errors.
- **Chapter 2, "Using keywords to identify the problem," on page 3** explains how to build and use a complete keyword string to specify program failures.
- **Chapter 3, "Using the IBM Support Center," on page 13** explains how to use the keyword string developed from the diagnostic procedures as a search argument in the Software Support Facility (SSF) or IBMLink/ServiceLink.
- **Chapter 4, "Basic documentation requirements," on page 15** lists the minimum documentation that will be needed by the IBM support group to diagnose customer problems or to document an authorized program analysis report (APAR).
- **Chapter 5, "Using the problem determination aid facility," on page 19** explains the use of this diagnostic tool.
- **Chapter 6, "Copying data sets to tape," on page 27** shows JCL examples that demonstrate ways to copy these data sets to tape.
- **Chapter 7, "Locating modules and control blocks in a dump," on page 31** shows by example how to obtain important information from a dump.
- **Chapter 8, "Diagnosing from return codes and reason codes," on page 37** explains how to diagnose problems from return codes and reason codes.
- **Chapter 9, "Using patches for problem determination," on page 91** describes the DFSMShsm-supported problem determination patches.
- **Chapter 10, "Using DFSMShsm maintenance commands," on page 95** explains how to use DISPLAY, FIXCDS, PATCH, and TRAP commands in problem diagnosis.
- **Chapter 11, "Introduction to data areas and control blocks," on page 135** provides a reference to descriptions of the DFSMShsm data areas and control blocks.

Required product knowledge

You should be familiar with the IBM Support Center, basic dump analysis, and diagnostic techniques. You are presumed to have a background in programming, using TSO, and z/OS concepts and terms and to understand the information in *z/OS DFSMS Introduction*.

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 Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

The z/OS Basic Skills Information Center

The z/OS Basic Skills Information Center is a Web-based information resource intended to help users learn the basic concepts of z/OS, the operating system that runs most of the IBM mainframe computers in use today. The Information Center is designed to introduce a new generation of Information Technology professionals to basic concepts and help them prepare for a career as a z/OS professional, such as a z/OS system programmer.

Specifically, the z/OS Basic Skills Information Center is intended to achieve the following objectives:

- Provide basic education and information about z/OS without charge
- Shorten the time it takes for people to become productive on the mainframe
- Make it easier for new people to learn z/OS.

To access the z/OS Basic Skills Information Center, open your Web browser to the following Web site, which is available to all users (no login required): z/OS Basic Skills in IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zbasics/homepage.html)

How to read syntax diagrams

This section describes how to read syntax diagrams. It defines syntax diagram symbols, items that may be contained within the diagrams (keywords, variables, delimiters, operators, fragment references, operands) and provides syntax examples that contain these items.

Syntax diagrams pictorially display the order and parts (options and arguments) that comprise a command statement. They are read from left to right and from top to bottom, following the main path of the horizontal line.

For users accessing the Information Center using a screen reader, syntax diagrams are provided in dotted decimal format.

Symbols

The following symbols may be displayed in syntax diagrams:

Symbol
Definition

►►— Indicates the beginning of the syntax diagram.

—► Indicates that the syntax diagram is continued to the next line.

►— Indicates that the syntax is continued from the previous line.

—►◄ Indicates the end of the syntax diagram.

Syntax items

Syntax diagrams contain many different items. Syntax items include:

- Keywords - a command name or any other literal information.
- Variables - variables are italicized, appear in lowercase, and represent the name of values you can supply.
- Delimiters - delimiters indicate the start or end of keywords, variables, or operators. For example, a left parenthesis is a delimiter.
- Operators - operators include add (+), subtract (-), multiply (*), divide (/), equal (=), and other mathematical operations that may need to be performed.
- Fragment references - a part of a syntax diagram, separated from the diagram to show greater detail.
- Separators - a separator separates keywords, variables or operators. For example, a comma (,) is a separator.

Note: If a syntax diagram shows a character that is not alphanumeric (for example, parentheses, periods, commas, equal signs, a blank space), enter the character as part of the syntax.

Keywords, variables, and operators may be displayed as required, optional, or default. Fragments, separators, and delimiters may be displayed as required or optional.

Item type
Definition

Required
Required items are displayed on the main path of the horizontal line.

Optional
Optional items are displayed below the main path of the horizontal line.

Default
Default items are displayed above the main path of the horizontal line.

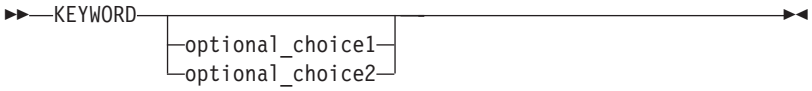
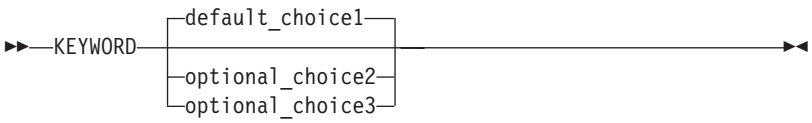
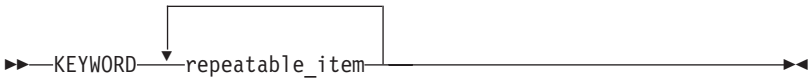
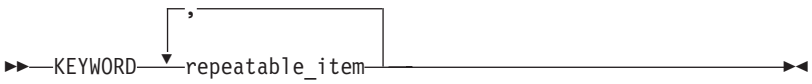
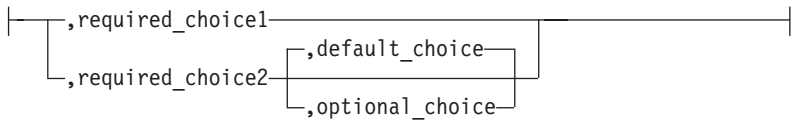
Syntax examples

The following table provides syntax examples.

Table 1. Syntax examples

Item	Syntax example
Required item. Required items appear on the main path of the horizontal line. You must specify these items.	►►—KEYWORD—required_item—►◄

Table 1. Syntax examples (continued)

Item	Syntax example
Required choice. A required choice (two or more items) appears in a vertical stack on the main path of the horizontal line. You must choose one of the items in the stack.	
Optional item. Optional items appear below the main path of the horizontal line.	
Optional choice. An optional choice (two or more items) appears in a vertical stack below the main path of the horizontal line. You may choose one of the items in the stack.	
Default. Default items appear above the main path of the horizontal line. The remaining items (required or optional) appear on (required) or below (optional) the main path of the horizontal line. The following example displays a default with optional items.	
Variable. Variables appear in lowercase italics. They represent names or values.	
Repeatable item. An arrow returning to the left above the main path of the horizontal line indicates an item that can be repeated. A character within the arrow means you must separate repeated items with that character. An arrow returning to the left above a group of repeatable items indicates that one of the items can be selected,or a single item can be repeated.	 
Fragment. The fragment symbol indicates that a labelled group is described below the main syntax diagram. Syntax is occasionally broken into fragments if the inclusion of the fragment would overly complicate the main syntax diagram.	 **fragment:** 

How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or provide any other feedback that you have.

Use one of the following methods to send your comments:

1. Send an email to mhvrcfs@us.ibm.com.
2. Send an email from the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html).

Include the following information:

- Your name and address.
- Your email address.
- Your telephone or fax number.
- The publication title and order number:
  z/OS V2R2 DFSMShsm Diagnosis
  GC52-1387-01
- The topic and page number that is related to your comment.
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive right 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

Do not use the feedback methods that are listed for sending comments. Instead, take one of the following actions:

- Contact your IBM service representative.
- Call IBM technical support.
- Visit the IBM Support Portal at z/OS Support Portal (http://www-947.ibm.com/systems/support/z/zos/).

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.

Summary of changes for z/OS Version 2 Release 2 (V2R2)

The following changes are made for z/OS Version 2 Release 2 (V2R2).

Changed

- New error codes added to "Return code processing" on page 37.

Summary of changes for z/OS Version 2 Release 1

See the following publications for all enhancements to z/OS Version 2 Release 1 (V2R1):

- *z/OS Migration*
- *z/OS Planning for Installation*
- *z/OS Summary of Message and Interface Changes*
- *z/OS Introduction and Release Guide*

Chapter 1. Introduction

If you experience a problem with the operation of DFSMShsm, it will be necessary to accurately describe the problem to your IBM Level 2 support group so that they may quickly help you solve the problem. This section explains how to describe DFSMShsm program failures through the use of keywords. A ***keyword*** is an agreed-upon word or abbreviation used to describe a single aspect of a program failure. This section shows you how to systematically develop a ***set of keywords*** that describes a program failure.

After you have selected a set of keywords, use it to search the ServiceLink function within IBMLink. You might determine whether an authorized program analysis report (APAR) has already been recorded for the failure. An APAR is a record of a product operation discrepancy. The IBM Software Support Facility (SSF), an online database available only to IBM personnel, contains APAR resolution information. If ServiceLink identifies a program failure with the same set of keywords, your search will yield a description of the problem and usually a solution. If the failure is not on record, use the keywords to describe the failure when you contact IBM for assistance.

Diagnosing Errors: Use the following steps to diagnose program failures:

1. For a description of system messages, use LookAt or see *z/OS MVS System Messages, Vol 2 (ARC-ASA)*. You can access IBM message explanations directly from the LookAt Web site at http://www.ibm.com/systems/z/os/zos/bkserv/lookat/.
2. Look up the commands in the DFSMShsm section of *z/OS DFSMSdfp Storage Administration*.
3. Examine the parameters specified by each command to verify that they are specified correctly.
4. If you notice any messages that indicate that an command that is not valid or a parse error was received due to a missing or parameter that is not valid, correct the error and resubmit the command.
5. If all parameters appear to be correctly specified, you can build a set of keywords that describes the error, and then contact IBM for assistance.

Using Keywords: When you contact IBM, you will be asked to identify your problem with a full set of keywords. Each keyword describes an aspect of a program failure. A full set of keywords for DFSMShsm is made up of the following:

- The component identification number
- The release and modification level
- The type of failure
- The command involved
- The DFSMShsm function involved
- The load module or the control section (CSECT), or both

The more precisely the keyword describes the failure, the more selective the resulting search can be, thus increasing the chance of finding an APAR that already addresses a similar failure.

Use program temporary fix (PTF) numbers as a keyword only if you feel that the PTF has caused the problem. The match you are looking for might have been found for a program with an earlier or later PTF level than yours.

A search of ServiceLink using the DFSMShsm component identifier (5695DF170) by itself detects all reported problems for the entire program product. However, each keyword added to the search argument makes the search more specific, thereby reducing the number of problem descriptions needing consideration.

If you are doing your own search, use the following guidelines for building search arguments that are generic in nature. The arguments may need to be varied slightly for the particular search facility that you are using. Most search facilities have logical operators (AND, OR, and NOT) as well as wild card characters (.), which enhance the ability to search the database and limit the number of matches. For the purposes of this section, we will try to keep the suggested search arguments as generic as possible. However, where required, the conventions used are those available through ServiceLink:

- A search argument may contain no more than 14 words or 122 characters.
- It cannot contain a question mark (?).
- No word can be longer than 15 characters. Words longer than 15 characters are truncated at 15 characters and treated as abbreviations.
- Each word must be separated from the next by at least one blank.

The logical operator for AND is the blank; for OR the vertical bar (|); and for NOT the not symbol (¬) or single quote ('). The character for searching for an abbreviation is the asterisk (*).

Understanding DFSMShsm data areas and control blocks: In this document, the term *data area* refers to the DFSMShsm control data sets (CDSs), the resources that DFSMShsm uses to manage the storage environment. The three DFSMShsm control data sets are the backup control data set (BCDS), the migration control data set (MCDS), and the offline control data set (OCDS). These control data sets contain information about DFSMShsm settings and describe in-storage information that is used by DFSMShsm for internal processing.

For descriptions of the DFSMShsm control data set records, see *z/OS DFSMShsm Data Areas*, which is available online at the z/OS Internet Library at http://www.ibm.com/systems/z/os/zos/bkserv/.

Using DFSMShsm health checks to prevent problems: IBM Health Checker for z/OS includes the following checks, which are designed to help you determine whether DFSMShsm is configured correctly:

- HSM_CDSB_BACKUP_COPIES. This check determines whether DFSMShsm is configured to maintain a critical level of control data set (CDS) backups.
- HSM_CDSB_DASD_BACKUPS. If DFSMShsm control data set backups are created on DASD, this check will ensure that all required DASD backup data sets are in place.
- HSM_CDSB_VALID_BACKUPS. This check determines whether the number of valid control data set (CDS) backups has fallen below a critical level.

If a check finds a potential problem, it issues a detailed message. For more information, see *IBM Health Checker for z/OS User's Guide*.

Chapter 2. Using keywords to identify the problem

This section explains individual keywords and their relation to the full set of keywords used in describing a DFSMShsm program failure. There are six types of keywords.

The following table shows each type of keyword and the associated procedures for using the keywords to identify the problem.

You can now perform the steps for the decision you have made.

If the keyword type is . . .	Then see the associated procedure . . .
component identification	"Using the component identification keyword"
release	"Using the release keyword" on page 4
type of failure	"Using the type of failure keyword" on page 4
command	"Using the command keyword" on page 10
function	"Using the function keyword" on page 10
load module or control section (CSECT), or both	"Using the load module and CSECT keywords" on page 10

The following example displays a full set of keywords:

```
5695DF170 Rnnn  type command function module
```

Where Represents

5695DF170
Component identification keyword
R*nnn* Release-level keyword
type Type-of-failure keyword
command
Command keyword
function
Function keyword
module
Module keyword

Using the component identification keyword

Use the component identification number whenever DFSMShsm is suspected of being the failed component. Combine the component identification keyword with other keywords to search SSF or IBMLink and ServiceLink. Used alone, this keyword produces a full listing of APARs against DFSMShsm.

The component identification number for the DFSMShsm functional component is:

Use of the release keyword with the component identification number narrows the symptom search to the specified version and release. The release keyword is explained in "Using the release keyword."

Using the release keyword

The release keyword, which is required on APAR forms, identifies the release of DFSMShsm.

Table 2 shows the release keyword values for the current releases of DFSMShsm.

Table 2. Release keyword values for current releases of DFSMShsm

Release	Keyword	Description
z/OS 1.11.0	RB10	Base
	RB10	English/ISMF panels
	RB1K	Japan/ISMF panels
z/OS 1.12.0	RC10	Base
	RC10	English/ISMF panels
	RC1K	Japan/ISMF panels
z/OS 1.13.0	RD10	Base
	RD10	English/ISMF panels
	RD1K	Japan/ISMF panels

Because several keywords are associated with each release, you might not find a specific fix unless you search all keywords for the problem description. If you do not find a matching APAR problem description, omit the release level keyword from the search argument. Doing so widens the search to include similar failures on other releases.

Using the type of failure keyword

The following table shows each type of failure keyword and the associated procedures for using the keywords to identify the problem.

If the failure keyword type is . . .	Then see the associated procedure . . .
ABEND*xxx*	"ABENDxxx" on page 5
INCORROUT	"INCORROUT" on page 7
LOOP	"LOOP" on page 7
MSG*xxxxxxxx*	"MSGxxxxxxxx" on page 8
WAIT	"WAIT" on page 8
xxxxxxxxxx (Documentation)	"xxxxxxxxxx (documentation)" on page 9
PERFM	"PERFM" on page 9

ABENDxxx

Use this keyword when the system, DFSMShsm, or any program that services DFSMShsm ends abnormally. When specifying this keyword, you should have a PDA trace at the time the abnormal end occurred, and one of the following types of dumps:

- An unformatted stand-alone dump
- An unformatted SYS1.DUMP
- A SYSMDUMP

Do not use this keyword if the abnormal end has been forced by the system or the operator because of a prolonged wait state or an endless loop. For those situations, refer to the WAIT and LOOP keywords.

Steps for using the ABENDxxx failure keyword

The format of the keyword string is:

```
5695DF170 Rnnn ABENDxxxx module name
```

R*nnn* is the release keyword, and its optional use can reduce the number of matches in the search.

1. Add the abend code (xxxx) to the ABEND keyword. DFSMShsm issues either system or user abend codes; therefore, you should use both of the following search methods:

   System abend code
   Entered in hexadecimal and has a fixed length of 3 hexadecimal characters. For example, use ABEND0C6 if an 0C6 abend occurred. Other search words common to a DFSMShsm system abend include: ABEND0C4, ABEND878, and ABENDE37.

   User abend code
   Entered in decimal and varies in length. It is entered without any leading zeros. For example, a 3EC abend would be entered as ABEND1004. Note that there are no zeroes between the abend prefix and the first nonzero character in the decimal value of the user abend code.

2. If the abnormal end has occurred in a DFSMShsm primary address space, message ARC0003I indicates the module name and offset of the abnormal end. An alternate method can be used, however, if message ARC0003I is not received or if the message states that the module and offset are unknown. The program status word (PSW) and the register contents at the time of the abend can be found in the RTM2WA summary obtained by using the SUMMARY command in the Interactive Problem Control System (IPCS). Proceed to the instructions in Step 4 after you have determined the storage location at which the abend occurred.
3. If the abnormal end has occurred in the ABARS secondary address space, message ARC6035E indicates the module name and offset of the abnormal end. The PSW address and the register contents at the time of the abend are saved in the ARCWABNP control block. To find this control block, issue a FIND command for *WABNP250* from the IPCS browse function. Look at the WABPSW2 field to find the PSW address at the time of the abend. If message ARC6035E is not received, the data in ARCWABNP is not reliable and you must use the information stored in the RTM2WA summary. The program status word (PSW) and the register contents at the time of the abend can be found in

the RTM2WA by using the SUMMARY command in IPCS. Proceed to the instructions in Step 4 after you have determined the storage location at which the abend occurred.

4. If the module corresponding to the address in the PSW is not identified by either message ARC0003I or ARC6035E, you can identify the failing module using IPCS. Issue a LOCATE command for the PSW address from the browse function. If the address is not found, issue the WHERE command to identify the failing load module. If the WHERE command indicates that the load module is in LPA, issue the IPCS LPAMAP command to identify the failing CSECT. If the LOCATE did find the PSW address, read backward through "eye catcher area" in the dump, until you find the name of the module that abnormally ended. See the following example:

```
00DDCFD0.  00000000  00000000  00000000  00000000  | ................ |
00DDBFE0.  00000000  00000000  00000000  00000000  | ................ |
00DDBFF0.  00000000  00000000  00000000  00000000  | ................ |
00DDC000.  47F0F00C  001847F0  F006011E  05C041B0  | .00....00....».. |
00DDC010.  CFFF4AF0  F00407FF  47F0C080  C9C7C7F0  | ...00....0».IGG0 |
00DDC020.  F1F9C2E9  F0F261F1  F961F8F8  C8C4D7F2  | 19BZ02/19/88HDP2 |
00DDC030.  F2F3F040  E4E8F1F8  F6F4F840  F5F6F6F5  | 230 UY18648 5665 |
00DDC040.  60E7C1F2  404DC35D  40C3D6D7  E8D9C9C7  | -X2A (C) COPYRIG |
00DDC050.  C8E340C9  C2D440C3  D6D9D74B  40F1F9F8  | HT IBM CORP. 198 |
00DDC060.  F26B40F1  F9F8F240  D3C9C3C5  D5E2C5C4  | 2, 1982 LICENSED |
00DDC070.  40D4C1E3  C5D9C9C1  D3E24060  40D7D9D6  |  MATERIALS - PRO |
00DDC080.  D7C5D9E3  E840D6C6  40C9C2D4  40009180  | PERTY OF IBM .j. |
00DDC090.  10254780  C114900F  D0004120  20005F20  | ....A...Œ.....¬. |
```

Note: This example was obtained by using Interactive Problem Control System (IPCS).

The preceding example reflects a typical module "eye catcher" in the dump. The "eye catcher" area varies from component to component but generally contains the module name (shown highlighted) the compile date, the function modification identifier (FMID), and the service level in the form of a program temporary fix (PTF) or APAR number. In this example, the module name is IGG019BZ, the compile date is 02/19/88, the FMID is HDP2230, and the service level is PTF UY18648.

To find the "eye catcher" area at the abend location, use the instruction address from the "PSW on entry to ABEND" or the "PSW at time of error". Scroll backward in the dump scanning the EBCDIC columns until you locate the associated module name in the "eye catcher" area.

In this example, module IGG019BZ is a part of DFP and not DFSMShsm. Because the abend occurred in the DFSMShsm address space but in a DFP module, the error could be in either DFSMShsm or DFSMSdfp.

After determining the module name, the format of the keyword string is *ABENDxxx module name*, where *xxx* is the abend code. If the failing module is a non-DFSMShsm module, do not use the DFSMShsm component identification number (5695DF170) or the R*nnn* keyword.

Installation exit abends

The system issues message ARC0004I when an installation exit abends. The message identifies the exit code and the abend code.

Note: Because IBM does not support installation exits, it is the user's responsibility to debug installation exit abnormal ends.

If the system fails in an installation exit, you can force DFSMShsm to generate a dump. You can prevent DFSMShsm from continuing its normal ESTAE recovery process by using the DFSMShsm PATCH command to set the MCVTFDMP flag on. For example:

```
PATCH .MCVT.+2D BITS(.......1)
```

See Chapter 10, "Using DFSMShsm maintenance commands," on page 95 for more information about the PATCH and DISPLAY commands.

INCORROUT

Use this keyword when the expected output is missing (not received) or when the output is different from what is expected (incorrect).

Procedure for using the INCORROUT failure keyword

The keyword string is:

```
5695DF170 Rnnn INCORROUT keyword
```

where ***keyword*** is a word or words relating to what you are doing at the time incorrect output is generated. ***R****nnn* is the release keyword, and its optional use reduces the number of matches in the search.

1. If you are receiving incorrect output and using a command such as HLIST, put the keyword *HLIST* (or whatever command you are using) in the search string.

LOOP

Use this keyword if a part of the program code runs endlessly; include situations in which a part of the output repeats endlessly.

Do not use this keyword for an intentional loop used to wait for a resource. For this condition, refer to the WAIT keyword.

Steps for using the LOOP failure keyword

The format of the keyword string is:

```
5695DF170  Rnnn LOOP modulename
```

R*nnn* is the release keyword, and its optional use reduces the number of matches in the search.

1. If a DFSMShsm program suspends activity for no apparent reason, the program may be in a loop or wait state. A loop can be evident from a never-ending request found by the QUERY ACTIVE command or from some other symptom such as a page of output being printed repeatedly. If a loop is not evident, use the WAIT keyword.
2. If you have a complete storage dump, locate the address where the LOOP occurs. DFSMShsm modules contain the module name near the beginning of the module. See the following example:

```
00007C30.  00000000  00000000  00000000  00000000  | ................ |
00007C40.  00000000  00000000  00000000  00000000  | ................ |
00007C50.  00000000  00000000  00000000  00000000  | ................ |
00007C60.  47F0F022  1DC1D9C3  C1C2D4E2  C74BF8F7  | .00..ARCABMSG.87 |
00007C70.  F0F4F44B  F1F0F0F0  F3F14BC8  C8D4F2F3  | 044.100031.JDZ11 |
00007C80.  F0F290EC  D00C18CF  1FFF43F0  C5081F00  | BC..Œ.....0E...  |
00007C90.  BF07C509  47F0C03C  00000200  89F00008  | ..E..0Œ.....i0.. |
```

Note:

a. This example has been obtained by using IPCS.

b. See *z/OS MVS Diagnosis: Reference* and *z/OS MVS Diagnosis: Tools and Service Aids* for information about locating modules within a loop.

3. Locate the system trace table to determine if a DFSMShsm task is being dispatched repeatedly in the same module. If the loop involves more than one DFSMShsm module, the repetitive invoking of these modules by DFSMShsm can be identified by the PDA trace facility. See Chapter 5, "Using the problem determination aid facility," on page 19 for further information.

MSGxxxxxxxx

Use this keyword when any of the following conditions occurs:

- DFSMShsm issues a message indicating an internal program error.
- A message is not issued in a set of conditions that should cause it to be issued.
- A message is issued in a set of conditions that should not cause it to be issued.
- A message contains non-valid or missing data.

A message keyword is a commonly used search argument. Most APARS will indicate the specific messages received as a result of a failure.

Procedure for using the MSGxxxxxxxx failure keyword

The format of the keyword string is:

```
5695DF170  Rnnn MSGxxxxxxxx
```

***R**nnn* is the release keyword, and its optional use can reduce the number of matches in the search.

1. Replace the *xxxxxxxx* in the MSG*xxxxxxxx* keyword with the message identifier. For example, if the message number is ARC0025I, the MSG*xxxxxxxx* keyword is *MSGARC0025I*.

When reporting a problem with a message symptom to the IBM Support Center, please have available the message IDs and complete message text from all related messages. This includes all variable text, return codes, reason codes, and so on. You may also want to search the DFSMShsm activity logs, X and Y logs, and the PDA trace for additional messages. In addition, the support center may need the PDA trace to determine the origin of the message.

WAIT

Use this keyword while waiting for a specific condition to be satisfied, the system, DFSMShsm, or some program that services DFSMShsm has suspended activity without issuing a message.

Do not use this keyword if the wait occurs after an abnormal end, as the result of an unanswered message, or because of an endless loop in DFSMShsm. Use the ABEND or LOOP keyword.

Procedure for using the WAIT failure keyword

The format of the keyword string is:

```
5695DF170 Rnnn WAIT module name resource name
```

***R**nnn* is the release keyword, and its optional use can reduce the number of matches in the search. The module name and resource name are optional keywords.

The wait state can have many causes; here are some of the most common causes:

- The system resource manager has marked DFSMShsm nondispatchable because of excessive paging.
- DFSMShsm has serialized on a resource and is locked out by another task in this system or by a task in another system.
- A backup task is waiting for a backup volume to become available. These backup volumes have been made unavailable because another processing unit stopped before completing backup.
- The WAIT may be a coded WAIT.

xxxxxxxxxx (documentation)

Use the documentation keyword when a programming problem appears to be caused by incorrect, missing, or ambiguous information in one of the DFSMShsm books.

Steps for using the xxxxxxxxxx (documentation) failure keyword

The format of the keyword string is:

```
5695DF170 xxxxxxxxxx  keyword
```

keyword is an optional word, such as the command or DFSMShsm function involved. For example, if the DFSMShsm function is migration, use the keyword *migration*.

1. Enter the order number of the publication in place of *xxxxxxxxxx*, the documentation keyword. Include the letter prefix and version number, but omit all hyphens. For example, if the order number is SH35-0083-02, the documentation keyword is *SH35008302*.
2. Locate the page in the book on which the error or omission occurs and prepare a description of the problem. Also include this information in the error description when submitting documentation for an APAR.

If you feel the documentation is unclear, please bring your concerns to our attention by submitting the Reader's Comment form located in the back of the manual.

PERFM

Use this keyword when some part of DFSMShsm performs below explicitly stated expectations and the performance problem cannot be corrected by system tuning.

Most performance problems are related to system tuning and should be handled by system engineers and system programmers.

Procedure for using the PERFM failure keyword

The format of the keyword string is:

```
5695DF170 Rnnn PERFM
```

Rnnn is the release keyword, and its optional use can reduce the number of matches in the search.

Using the command keyword

Build keywords based on DFSMShsm commands. Keep in mind that a search keyword can have a maximum of 15 characters. Words longer than 14 characters are truncated at 15 characters and treated as abbreviations. Refer to Chapter 1, "Introduction," on page 1 for guidelines on building search arguments.

The format of the keyword string is:

```
5695DF170 Rnnn TAPEHARDWARECOMPACT RECYCLE | TAPECOPY
```

Through ServiceLink, this sample argument lists as matches any APAR in the DFSMShsm release specified by "*nnn*" that contains the word TAPEHARDWARECOMPACT and either the word RECYCLE or the word TAPECOPY. Note that because TAPEHARDWARECOMPACT contains more than 15 characters, it is treated by ServiceLink as if entered as TAPEHARDWARECOM*; that is, it is treated as an abbreviation search and matches TAPEHARDWARECOMPRESS, TAPEHARDWARECOMPARE, and so on.

Using the function keyword

Function keywords refer to basic DFSMShsm functions. Use keywords such as backup, recall, or other basic functions. Because of the numerous matches you can get, use of the function keyword by itself is not recommended. For example, if the DFSMShsm function is recycle:

The format of the keyword string is:

```
5695DF170 Rnnn RECYCLE
```

R*nnn* is the release keyword, and its optional use can reduce the number of matches in the search.

Using the load module and CSECT keywords

Use module names that appear in messages when you build load module and CSECT keywords. Examples of load modules for DFSMShsm include:

ARCCTL, ARCWCTL, and **ARCLISTM**

Perform a search on a particular module to find the most recent APARS.

The format of the keyword string is:

5695DF170 R*nnn* **ARCCTL**

R*nnn* is the release keyword, and its optional use can reduce the number of matches in the search.

Chapter 3. Using the IBM Support Center

IBM Support Center personnel have access to several software support databases and are responsible for using the set of keywords you provide as a search argument to help solve the program failure. Support Center personnel may help you improve the effectiveness of your search argument, and if the problem has previously been reported, they can provide records describing the failure and the corrective action.

The types of software support databases available to the IBM Support Center personnel include:

- Software support facility (SSF)
- IBMLink and ServiceLink
- Info/System.

Using the software support facility

The software support facility (SSF) is an IBM online database containing information about all current APARs and PTFs. IBM Support Center personnel have access to SSF and are responsible for using the set of keywords you provide as a search argument. Support Center personnel may help you improve the effectiveness of your search argument. If the problem has previously been reported, they can retrieve the records describing the failure and the corrective action.

Using IBMLink and ServiceLink

IBMLink and ServiceLink are a set of online electronic services available to customers. Some of these services are available to you free of charge as a part of the SoftwareXcel basic contract. Some of these services are available as part of the optional SoftwareXcel Extended contract for an additional fee. Contact your local IBM marketing branch office for more information on SoftwareXcel contracts and services.

The following services are available to you under one of these contracts:

SRCHSERVICE
Online database of current authorized program analysis report (APAR) and program temporary fix (PTF) information with extensive search capability.

PSP Preventive Service Planning information database. This data contains the latest information concerning the installation of IBM products including the latest service recommendations.

SRD Service Request and Delivery facility. This facility provides a means for election ordering and delivery of corrective services including PTFs and APARS.

ASAP Automatic Software Alert Process. This facility allows the user to be alerted when critical service information becomes available on a list of products selected by the user.

ETR Electronic Technical Response. Through this facility, the user may electronically report problems and ask appropriate technical questions about IBM products. Problem reports and questions are answered

electronically. Optionally, problem reports may be answered through voice contact at the request of the user. Submit nondefect-related, nontechnical questions to the question and answer (Q&A); queue in Canada on a severity 3, priority 3 basis.

AST Automatic Status Tracking. This facility allows the user to request notification when the status of a user-selected APAR or PTF changes, or both.

VPL View Program Listings. Online database of module listings for non-OCO modules distributed via PTFs.

Info/System

Info/System, an interactive online database information retrieval program product, is available primarily for use by customers with the companion database feature, Info/MVS. The database divides itself into several logical files of related or similar information, such as IBM ServiceLink.

Chapter 4. Basic documentation requirements

This topic discusses the basic documentation you will need for problem diagnosis or if IBM opens an authorized problem analysis report (APAR) addressing your problem.

Documentation needed for problem diagnosis

After the proper diagnostic procedures have been followed, user specifications have been checked for accuracy, and the keyword search has proven unsuccessful, contact the IBM Support Center for further assistance.

If the Support Center representatives cannot immediately solve the problem, they may pass the problem call to the Level 2 support group. When the support group representatives call you, they will request that you have certain documentation at hand to aid in the diagnosis of the problem.

When you speak with the representative from the Level 2 support group, you will need to have immediate access to the following documentation:

- The PDA trace, preferably in a form that can be browsed online.
- An unformatted dump, obtained as a SYS1.DUMP or SYSMDUMP. A SYS1.DUMP is preferred because it supports the dump analysis elimination function, which prevents duplicate dumps occurring for the same problem. For a description of how to use the SETSYS SYS1DUMP command, refer to the DFSMShsm section of *z/OS DFSMSdfp Storage Administration*.

The dump is available for browsing online by the Interactive Problem Control System (IPCS). In general, the Level 2 support can work with other forms of dumps including hardcopy, but their use of other forms of dumps may seriously impact the effectiveness of diagnostic efforts. Softcopy, raw format dumps, traces, and logs are generally more useful for the support group when it becomes necessary to open an APAR.

Because it is possible that the support group may need additional documentation, either to pursue problem diagnosis or to open an APAR, you should keep the following documentation for possible future use:

- The system console log from around the time the problem first occurred.
- The portion of the journal pertaining to the problem.
- The related function statistics records (FSRs), daily statistics records (DSRs), and volume statistics records (VSRs) from System Management Facility (SMF). You can use the SETSYS SMF command to extract these records.
- The command activity log, if appropriate.
- The function-specific activity log, if appropriate.

Retain these logs and journals until you are sure that they are no longer needed.

Documentation needed for an authorized program analysis report

If IBM opens an authorized program analysis report (APAR) addressing your problem, be prepared to supply the following information:

- Customer number
- Version
- Release and modification level
- Keyword string used to search IBMLink/ServiceLink
- Current maintenance level, obtained with the MODLEVEL parameter of the DISPLAY command described in "DISPLAY: Displaying DFSMShsm storage locations" on page 95.

You will also be asked to supply various types of information that describe the:

- DFSMShsm functions used
- Database used
- Environment
- Activities.

Applicable items of information from the following list may also be requested:

- JCL listings (for a failure related to a BATCH job).
- System console log.
- The related FSRs, DSR, and VSRs from SMF. You can use the SETSYS SMF command to extract these records.
- The command activity log, if appropriate.
- The specific activity log, if appropriate.
- PDA trace.
- A copy of any traces or SYS1.DUMPs taken.
- A DFSMSdss logical dump of the data set on tape, if the problem relates to a specific data set.
- For a wait failure, a complete description of the resource being waited for and the program module that is waiting.

  An MVS™ DUMP command must be issued prior to canceling DFSMShsm. Specify the DUMP options as follows:

  jobname—
  specifies the name of the job associated with the address space you want to dump (for example, *DFHSM* or *DFHSMABR*).

  sdata—
  specifies the specific storage areas you want to dump. The options are CSA, GRSQ, LSQA, PSA, RGN, SQA, SUM, SWA, and TRT.

  For information about DUMP options and their definitions, see the section titled "DUMP Commands" in *z/OS MVS System Commands*.
- For LOOP failures, an indication of the location of the loop. This information can be taken from console trace addresses or it can be one or more module names taken from the PDA trace. Try to get at least a partial trace of the loop.
- For documentation failures, the location of the error in the manual and a description of the problem it caused.
- For PERFM failures, a description of the actual performance, the expected performance, and the source of the performance specification.

Note: When you submit any of the requested documentation on tape, write it to a **standard** label tape. A hardcopy of the data control block (DCB) information for each data set and the **JCL** used to create the tape is also required.

Chapter 5. Using the problem determination aid facility

During DFSMShsm processing, the problem determination aid (PDA) facility gathers diagnostic information about DFSMShsm processing, stores this information in a circular file within storage, and periodically writes it to a circular file on DASD. This type of file appends data until full; then, starting at the beginning of the file, subsequent incoming data overwrites the data already there. For each DFSMShsm host, the circular DASD file consists of two data sets: ARCPDOX and ARCPDOY. ARCPDOX is the active data set. You can make ARCPDOY become the active data set and ARCPDOX become the inactive data set by issuing the SWAPLOG PDA command. This allows you to take the active data set and make it inactive for the purpose of examining the diagnostic information that has been gathered.

For detailed information on using the problem determination aid or calculating the PDA log data set sizes, refer to *z/OS DFSMShsm Implementation and Customization Guide*.

There are several reasons why you will at times want to collect and save PDA trace data. The following reasons are most common:

- A trace showing DFSMShsm's operating history can pinpoint the activity at the time the problem first occurs. This information can be helpful due to the time difference between when a problem first occurs and when it is first detected.
- A trace can locate points of contention when two separate tasks conflict with one another.
- A trace can help you to determine if a suspected DFSMShsm problem really exists, or if it is an operational error.
- A trace may help locate a "missing" data set. You can scan for the last occurrence of the data set name.
- Traces are needed to supplement dumps when you contact your IBM support.

Two very useful functions can help you to extract the exact information you need from the PDA files. By using either or both the ISPF browse function and ARCPRPDO (PDA trace formatter) program, you may be able to solve your problem without assistance from IBM support.

The DFSMShsm trace formatter facility takes the raw trace data and organizes, reduces, and prints user-selected trace information. This is especially useful when you need specific data.

Use the **trace formatter facility** to:

- Translate trace records into a readable format
- Edit raw trace data
- Select records based on your specific criteria

Conditional tracing is available to allow users to turn off some of the problem determination aid (PDA) tracing that is normally performed. Reducing the number of trace points can improve the performance of PDA tracing, as well as reducing the amount of data to analyze. However, when using conditional tracing, there is the possibility that the system may not capture needed data on the first failure. Users may have to turn off conditional tracing and recreate the problem.

The default for tracing is to trace everything. You may use conditional tracing to turn off some of the tracing functions. You use the PATCH command to turn off the functions. See *z/OS DFSMShsm Implementation and Customization Guide*, Chapter 16, "Tuning DFSMShsm," for examples of the PATCH commands that DFSMShsm uses to implement conditional tracing.

Using the ARCPRPDO (PDA trace formatter) program

The ARCPRPDO (PDA trace formatter) program has a number of options that can assist you in collecting data from ARCPDOY, ARCPDOX, or a copy of either. To process the most recent PDA log entries, issue the SWAPLOG PDA command and then process the data that was placed in the ARCPDOY data set.

ARCPDOX can be browsed while DFSMShsm is running (disposition of the data set must be SHR).

If you need to format ARCPDOX while DFSMShsm is running, turn PDA tracing off with the SETSYS PDA(OFF) command. The SETSYS PDA(ON) command will restore PDA tracing.

The formatter can be used to collect and print data based on your criteria. The options useful for debugging are outlined in the following topics.

Command syntax

Formatting Options	Selection Options
COPY | EXTRACT COMPACT | FORMAT | NOPRINT	START(*date[,time]*) END(*date[,time]*) TCB(*aaaaaa[,bbbbbb,cccccc,...]*) MODULE(*aaaaa[,bbbbb,ccccc,...]*) LOGIC(*aaaa[,bbbb,cccc,...]*) SCAN(*data*) RECYCLE

Formatting options

The trace formatting options tell DFSMShsm what to do with the collected data. A brief description of each of the formatting options is given here. Examples using these options appear later in this section.

COPY: This option routes trace data that match your selection criteria to the data set specified for your ARCOUT DD. This new data set may now act as the raw data for subsequent search runs.

COMPACT:
This option formats the trace entries into single-line output, if possible. The compact option is the default option and does not need to be specified.

EXTRACT:
This option routes trace data that matches your selection criteria to the data set specified for your ARCOUT DD. This new data set has a prefix that contains date, time, and host identification that you can use for sorting or merging with other extract files. Extract data sets may act as the raw data for subsequent ARCPRPDO formatting runs.

FORMAT:
This option formats each trace keyword on a separate line. The keyword itself is printed at the start of each line.

NOPRINT:
This option prevents any selected records from being written to the ARCPRINT DD. NOPRINT is intended to be used with the copy option. An ARCPRINT DD statement is still required when you select the NOPRINT option, as it contains messages and other information.

Selection options

The various selection options listed here are useful for narrowing and defining your trace data search. These options are used within examples later in this section.

The **START(*yyddd[,hhmmss]*)** option allows you to select records for output starting from the specified date until the end of the data, or until an end date is reached, as specified by the END option. A start time can also be specified and is separated from the date by a comma. If you do not specify the START option, the system selects records that begin at the start of the ARCPDOX or ARCPDOY data set. If a start time is not specified, the default is 000000.

The **END(*yyddd[,hhmmss]*)** option selects records until the date, and optionally time, are reached. A comma separates the date and time. The end date must be the same as, or after, the start date. If the start and end dates are the same, the end time must be the same as, or after, the start time. If you do not specify the END option, the system selects records through the end of the file. If an end time is not specified, the default is 235959.

With both **START** and **END** options:

yy	Must be a number from 00 through 99
ddd	Must be a number from 001 through 366
hh	Must be a number from 00 though 23
mm	Must be a number from 00 through 59
ss	Must be a number from 00 through 59

The **TCB(*aaaaaa[,bbbbbb,cccccc,....]*)** option selects records that match any of the TCB addresses you select. Up to 10 TCB addresses may be specified in this option. The TCB addresses are identified under the AS/TCB column in the trace output display.

Each address must contain six characters, and each address must be separated by a comma. Actually, only bytes two and three of the actual TCB address are used in the trace address field, and these are placed in byte positions one and two. Byte zero of the TCB trace entry is used for address space identification. For activity that is recorded in the ABARS secondary address space, use the last two digits (in hex) of the ABARS address space identifier (ASID). The following example helps illustrate this:

```
ACTUAL TCB ADDRESS = 00F823C0
TRACED TCB ADDRESS = xxF823  (xx is an address space identifier)
```

The **MODULE(*aaaaa[,bbbbb,ccccc.....]*)** option selects all trace records that have been requested by the module you specify. For example, if you specify MODULE(CTL), all the trace records requested by ARCCTL are selected. You may specify up to ten module names for this option. Each name can contain up to five

characters, with each name separated by a comma. Whenever you specify a module name of less than five characters, DFSMShsm pads the name to equal five characters. The module name DFSMShsm traces actually starts on the fourth character of the true name. The following examples help illustrate this:

```
ACTUAL MODULE NAME = ARCSELTV
TRACED MODULE NAME = SELTV

ACTUAL MODULE NAME = ARCCTL
TRACED MODULE NAME = CTL
```

The **LOGIC(*aaaa[,bbbb,cccc,....]*)** option looks at the logic fields of the trace entries and selects only those that match one of the logic types you select. Up to 10 logic types may be specified in this option. Each type must contain four characters, with each type separated by a comma in the case of multiple entries. Three especially useful ones are MESG, TIME, and ENTR. Logic types are displayed under the LOGIC column in the trace output display.

The **SCAN(*data*)** option selects records that contain any reference to data you specify in the data field. The data may match parameters, such as data set names or a key to a data set record. For example, to select all records in which a particular data set name appears, specify the following parameter: SCAN(*your.data.set.name*).

The **RECYCLE** option selects records related only to RECYCLE activity.

Examples of PDA trace formatter options

Examples: These are examples of trace options and output:

- The following examples show different trace options:

```
//PDACOPY JOB MSGCLASS=A
  //COPY    EXEC PGM=ARCPRPDO
  //SYSPRINT DD SYSOUT=*
  //ARCMSG   DD SYSOUT=*
  //ARCPDO   DD DSN=HSM.PDOY,DISP=SHR
  //ARCOUT   DD DSN=PDOY.COPY,DISP=OLD
  //ARCPRINT DD SYSOUT=*
  //SYSIN    DD  *
    COPY
    NOPRINT
    START(00229,031148)
    END(00229,031527)
```

The above job copies those PDA trace entries from the HSM.PDOY data set that have a time stamp between 03:11:48 a.m. and 03:15:27 a.m. The Julian date is 00.229, which is August 16, 2000. No trace data is printed.

```
 //PRPDO2  JOB
  //PRINT   EXEC PGM=ARCPRPDO
  //SYSPRINT DD SYSOUT=*
  //ARCMSG   DD SYSOUT=*
  //ARCPDO   DD DSN=HSM.PDOY,DISP=SHR
  //ARCPRINT DD SYSOUT=*
  //SYSIN    DD  *
    COMPACT
    LOGIC(MESG)
```

The above job displays all PDA trace entries for messages, with one output line per entry.

```
//PRPDO3  JOB
  //PRINT   EXEC PGM=ARCPRPDO
  //SYSPRINT DD SYSOUT=*
  //ARCMSG   DD SYSOUT=*
  //ARCPDO   DD DSN=HSM.PDOY,DISP=SHR
  //ARCPRINT DD SYSOUT=*
  //SYSIN    DD  *
    COMPACT
    START(00207,081000)
    END(00208,170000)
    MODULE(RACF)
    SCAN(TEST.RECALL.RACF.FAILURE)
```

The preceding job formats all trace entries within ARCPDOY that were created by ARCRACF for data set TEST.RECALL.RACF.FAILURE between 8:10 a.m. on July 26, 2000 and 5:00 p.m. on July 27, 2000. Resource Access Control Facility (RACF®) is a component of IBM Security Server for z/OS.

- The following is an example of output using the COMPACT option:

```
 TIME   USECS   ID AS/TCB  HOST    MOD   LOGIC CALLER  ARCPRPDO
 00104------------------------------------------------------------
 104252.627968  00 009DA4  HOST=A  ZREAD ENTR   MCTL. *......MCT
 104252.632384  00 009DDE  HOST=A  ZREAD ENTR   ..?.. *........?
 104252.638000  00 009DD8  HOST=A  ZREAD ENTR   ZBDST *......ZBD
 104252.641376  00 009DD1  HOST=A  ZREAD ENTR   ..?.. *...1....?
 104252.727120  01 009DD1  HOST=A  ZREAD GET          *..BCR1...
```

- The following is an example of output with the FORMAT option:

```
TIME   USECS   ID AS/TCB   HOST    MOD   LOGIC CALLER  ARCPRPDO
 00104------------------------------------------------------------
 104252.627968  00 009DA4  HOST=A  ZREAD ENTR   MCTL.
   R13ADDR=       001F4B5800D4C3E3D340                    *....
 00104------------------------------------------------------------
 104252.632384  00 009DDE  HOST=A  ZREAD ENTR   ..?..
   R13ADDR=       001F1AA00F000E6F06                      *....
 00104------------------------------------------------------------
 104252.638000  00 009DD8  HOST=A  ZREAD ENTR   ZBDST
   R13ADDR=       001FA99000E9C2C4E2E3                    *....
```

Recommendations for using PDA formatter

Recommendations: These are recommendations that you may find helpful when you use the trace formatter to collect data from trace data sets:

- Use **Compact Logic(MESG)** to get an initial sense of the messages issued at the time of the error. This action can help you to locate TCB or module information for subsequent searches. Logic types are listed in the output display under the LOGIC column.
- Use **Format TCB(*aaaaaa*)** for task-specific errors such as migration or backup.
- Use **Compact Scan(*dsn*)** with the actual data set name for errors related to single data set processing, such as a missing data set.

Browsing the PDA data set

Examples: These examples, obtained by using the ISPF browse function and the HEX DATA command, aid you in identifying the trace entries when you are discussing the PDA trace data set with the IBM Level 2 support group:

- The following example shows a portion (four entries) of a PDA trace data set as it would appear on your terminal screen, just before you issue the HEX DATA command.

  Note: The file name (in this example, HSM.PDOX) is set by you in your startup procedure.

```
 BROWSE -- HSM.PDOX ------------------------------ LINE
 COMMAND ===> HEX DATA

LOCK .ENQ  ..D....ARCGPA  ..BAKQ..E..L.....................
UNLK .DEQ  ..D....ARCGPA  ..BAKQ..L.......................
MCTL .ENTR ..D.N.........Y.
IRCB .ENTR ..D.U.....H.MCTL
```

- The following example shows the same data set as in the previous example, after issuing the HEX DATA command. Note that the display has expanded and each entry is now shown in EBCDIC and hexadecimal notation.

```
 BROWSE -- HSM.PDOX ------------------------------ LINE
 COMMAND ===>

LOCK .ENQ  ..D....ARCGPA  ..BAKQ..E..L.....................
D3D6C3D24005C5D5D840008CEEC470040A29C1D9C3C7D7C1404006FFC2C1D2D803FFC503FFD300
00000000000000000000000000000000000000000000
UNLK .DEQ  ..D....ARCGPA  ..BAKQ..L.......................
E4D5D3D24008C4C5D840008CEEC4700B0A28C1D9C3C7D7C1404006FFC2C1D2D803FFD300000000
00000000000000000000000000000000000000000000
MCTL .ENTR ..D.N.........Y.
D4C3E3D34000C5D5E3D9008D3EC471D50BFC00302F700E00FDBDA800
00000000000000000000000000000000000000000000
IRCB .ENTR ..D.U.....H.MCTL
C9D9C3C24001C5D5E3D9008D3EC471E40CFC00302EC800D4C3E3D340
00000000000000000000000000000000000000000000
```

For simplicity, the remaining examples show only the first entry.

Module name identifier

Example: In this example, the highlighted five bytes are the last five bytes of the DFSMShsm module that has generated the entry. The highlighting is done in both the EBCDIC and hexadecimal portions of the examples whenever it is significant to do so.

```
 BROWSE -- HSM.PDOX ------------------------------ LINE
 COMMAND ===>

 LOCK  .ENQ  ..D....ARCGPA  ..BAKQ..E..L.....................
 D3D6C3D240 05C5D5D840008CEEC470040A29C1D9C3C7D7C1404006FFC2C1D2D803FFD30000
00000000000000000000000000000000000000000000
```

Trace point identifier

Example: The highlighted byte in this example is the individual trace point identifier:

```
 BROWSE -- HSM.PDOX ------------------------------ LINE
 COMMAND ===>

LOCK .ENQ  ..D....ARCGPA  ..BAKQ..E..L.....................
D3D6C3D240 05 C5D5D840008CEEC470040A29C1D9C3C7D7C1404006FFC2C1D2D803FFD30000
00000000000000000000000000000000000000000000
```

Logic type identifier

Example: The highlighted four bytes in this example show the logic type identifier:

```
 BROWSE -- HSM.PDOX ------------------------------ LINE
 COMMAND ===>

LOCK . ENQ  ..D....ARCGPA  ..BAKQ..E..L.....................
D3D6C3D24005 C5D5D840 008CEEC470040A29C1D9C3C7D7C1404006FFC2C1D2D803FFD30000
00000000000000000000000000000000000000000000
```

Task control block identifier

Example: The highlighted three bytes in this example show the unique task control block (TCB) identifier. The first byte is the address space identifier. If this byte is zeroed, the TCB resides in the DFSMShsm primary address space; otherwise, it resides in a secondary address space. The next two bytes are the second and third bytes (middle two bytes) of the TCB address.

```
 BROWSE -- HSM.PDOX ------------------------------ LINE
 COMMAND ===>

LOCK .ENQ  ..D....ARCGPA  ..BAKQ..E..L.....................
D3D6C3D24005C5D5D840 008CEE C470040A29C1D9C3C7D7C1404006FFC2C1D2D803FFD30000
00000000000000000000000000000000000000000000
```

Time of day clock entry

Example: The highlighted three bytes in this example show the fourth, fifth, and sixth bytes of the eight-byte processing unit time-of-day (TOD) clock entry:

```
 BROWSE -- HSM.PDOX ------------------------------ LINE
 COMMAND ===>

LOCK .ENQ  ..D....ARCGPA  ..BAKQ..E..L.....................
D3D6C3D24005C5D5D840008CEE C47004 0A29C1D9C3C7D7C1404006FFC2C1D2D803FFD30000
00000000000000000000000000000000000000000000
```

Data segment length

Example: A number of variable length data segments can follow the TOD clock entry, one for each trace value recorded. In this example, there are four data segments. The highlighted byte shows the length of the first segment (0A or ten bytes), and the bracketed area is the segment. Note that the length byte is included in the total length.

```
 BROWSE -- HSM.PDOX ------------------------------ LINE
 COMMAND ===>

LOCK .ENQ  ..D....ARCGPA  ..BAKQ..E..L.....................
D3D6C3D24005C5D5D840008CEEC47004 0A 29C1D9C3C7D7C1404006FFC2C1D2D803FFD30000
00000000000000000000000000000000 0000000
```

Parameter keyword identifier

Example: The highlighted byte in the first data segment is the parameter keyword identifier, as shown in this example:

```
  BROWSE -- HSM.PDOX ------------------------------ LINE
  COMMAND ===>

 LOCK .ENQ  ..D....ARCGPA  ..BAKQ..E..L.....................
 D3D6C3D24005C5D5D840008CEEC470040A 29 C1D9C3C7D7C1404006FFC2C1D2D803FFD30000
 00000000000000000000000000000000 0000000
```

Parameter data

Example: The highlighted bytes in this example show the variable data for this entry:

```
  BROWSE -- HSM.PDOX ------------------------------ LINE
  COMMAND ===>

 LOCK .ENQ  ..D.... ARCGPA  ..BAKQ..E..L.....................
 D3D6C3D24005C5D5D840008CEEC470040A29 C1D9C3C7D7C14040 06FFC2C1D2D803FFD30000
 00000000000000000000000000000000 0000000
```

Example: This bracketed area is the second data segment of this entry and the highlighted byte is the length of the segment (six bytes):

```
  BROWSE -- HSM.PDOX ------------------------------ LINE
  COMMAND ===>

 LOCK .ENQ  ..D....ARCGPA  ..BAKQ..E..L.....................
 D3D6C3D24005C5D5D840008CEEC470040A29C1D9C3C7D7C14040 06 FFC2C1D2D803FFD30000
 00000000000000000000000000000000000000000000000000
```

Padding

Example: The highlighted area in this example shows that the last entry is optionally padded with binary zeros:

```
  BROWSE -- HSM.PDOX ------------------------------ LINE
  COMMAND ===>

LOCK .ENQ  ..D....ARCGPA  ..BAKQ..E..L.....................
D3D6C3D24005C5D5D840008CEEC470040A29C1D9C3C7D7C1404006FFC2C1D2D803FFC503FFD3 0000
 00000000000000000000000000000000000000000000000000
```

Chapter 6. Copying data sets to tape

The JCL examples in this section show how to copy certain data sets onto a tape so that they can be submitted to IBM as APAR documentation. The tapes that you use for this purpose **must be standard label tapes.**

Because of numerous variations in hardware and software, these JCL jobs are shown only as examples; they might not work in all environments. Also note that you must choose the data set names, relative file numbers, tape volume serial number, and unit type (shown highlighted) that are consistent with your naming conventions.

When you create your JCL, do not use the high-level qualifiers SYS1, HSM, or DFHSM in the data set names specified in the SYSOUT2 DD cards because these may be reserved names in the system where your tape will be read. It is recommended that you use the APAR number as the high-level qualifier, followed by a qualifier that describes the content of the data set.

Copying the PDA trace data set to tape

This JCL shows a method of copying the PDA trace data set to tape.

```
//PDACOPY JOB MSGCLASS=A
//S1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT1 DD DSN=HSM.PDOY,DISP=SHR
//SYSUT2 DD DSN=OW99999.PDOY,LABEL=(1,SL),VOL=SER=TAPE01,
//          DISP=(NEW,KEEP),UNIT=TAPE
```

Frequently, only several minutes or an hour of the PDA trace is required. To reduce the amount of data to be copied, use the DFSMShsm trace formatter program, ARCPRPDO, to copy all trace entries created during the time span of interest. The example below copies the trace records created from Julian date 02.277 at 23:50:05 (50 minutes and 5 seconds past 11 p.m.) through 02.278 at 00:10:00 (10 minutes past midnight). Also, if the time span required is included in several PDA data sets, the data sets may be concatenated in chronological order in the JCL.

```
//PDACOPY  JOB  MSGCLASS=A
//S1       EXEC PGM=ARCPRPDO
//ARCMSG   DD   SYSOUT=*
//ARCPRINT DD   SYSOUT=*
//ARCPDO   DD   DSN=HSM.PDOY,DISP=SHR
//ARCOUT   DD   DSN=OW99999,PDOY,LABEL=(1,SL),VOL=SER=TAPE01,
//     DISP=(NEW,KEEP),UNIT=TAPE
//SYSIN    DD   *
   COPY
   NOPRINT
   START(02277,235005)
   END(02278,001000)
```

Copying the dump data set to tape

This JCL shows a method of copying a dump data set to tape.

```
//DMPCOPY JOB MSGCLASS=A
//S1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT1 DD DSN=SYS1.DUMP00,DISP=SHR
//SYSUT2 DD DSN=OW99999.DUMP,LABEL=(2,SL),VOL=SER=TAPE01,
//          DISP=(NEW,KEEP),UNIT=TAPE
```

Copying the journal data set to tape

If you are running SETSYS CDSVERSIONBACKUP(TAPE), then copy the appropriate journal data set or sets to tape. Be sure to include a **hardcopy of the JCL used** when you send the data sets to IBM.

If you have allocated the CDSs in your startup procedure, use the following JCL to copy the online journal data sets to tape.

```
                                              Note: if DFSMShsm
//JRNCOPY JOB MSGCLASS=A                     ┌─is in operation, the
//S1 EXEC PGM=IEBGENER                       │ disposition for DFSMShsm
//SYSPRINT DD SYSOUT=A                       │ must also be SHR.
//SYSIN DD DUMMY                             ▼
//SYSUT1 DD DSN=HSM.JRNL.BACKUP.V0000002,DISP=SHR
//SYSUT2 DD DSN=OW99999.JRNL,LABEL=(3,SL),VOL=SER=TAPE01,
//          DISP=(NEW,KEEP),UNIT=TAPE
```

Copying the DFSMShsm log data set to tape

This JCL shows a method of copying the DFSMShsm log Y data set to tape.

If the needed data is on the active DFSMShsm log (log X), run the JCL job, issue the SWAPLOG LOG command, and run the JCL job again.

```
//LOGCOPY JOB MSGCLASS=A
//S1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT1 DD DSN=HSM.LOGY,DISP=SHR
//SYSUT2 DD DSN=OW99999.LOGY,LABEL=(4,SL),VOL=SER=TAPE01,
//          DISP=(NEW,KEEP),UNIT=TAPE
```

Copying the activity log data set to tape

This JCL shows a method of copying the activity log data sets to tape. If you are running ACTLOGTYPE(DASD) and you wish to process the current activity logs, issue the following commands:

- SETSYS ACTLOGTYPE(SYSOUT)

  This command closes and deallocates the current activity logs.
- SETSYS ACTLOGTYPE(DASD)

  This command allocates new activity logs.

Using the following JCL example, the old logs can now be browsed or copied to tape:

```
//ALOGCOPY JOB MSGCLASS=A
//S1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT1 DD DSN=HSMACT.H1.BAKLOG.D89012.T085721,DISP=SHR
//SYSUT2 DD DSN=OW99999.ACTLOG,LABEL=(5,SL),VOL=SER=TAPE01,
//          DISP=(NEW,KEEP),UNIT=TAPE
```

Copying FSR, DSR, and VSR records to tape

This JCL shows a method of copying FSR, DSR, and VSR records from the SMF data set to tape.

If you are running SETSYS SMF (SMFID), then the records that are identified with SMFID contain daily statistics (DSR) and volume statistics (VSR). Records that are identified with SMFID+1 contain function statistics (FSR) and ABARS function statistics (WWFSR).

```
//SMFCOPY  JOB  MSGCLASS=A
//S1  EXEC  PGM=IFASMFDP
//SYSPRINT DD    SYSOUT=A
//SYSUT1   DD    DSN=SYS1.MANX,DISP=SHR
//SYSUT2   DD    DSN=0W99999.SMF,LABEL=(6,SL),
//               VOL=SER=TAPE01,DISP=(NEW,KEEP),UNIT=TAPE
//SYSIN DD *
 INDD(SYSUT1,OPTIONS(DUMP))
 OUTDD(SYSUT2,TYPE(SMFID,SMFID+1))
/*
```

Chapter 7. Locating modules and control blocks in a dump

Many vital DFSMShsm data areas are located by their addresses in the management communication vector table (MCVT). Figure 1 shows how two base items (MCVT, ARCESD) can be located when the dump includes low storage.

Note: The following information and examples refer only to the DFSMShsm primary address space. For dumps occurring in the ABARS secondary address space, contact the IBM Support Group for further assistance.

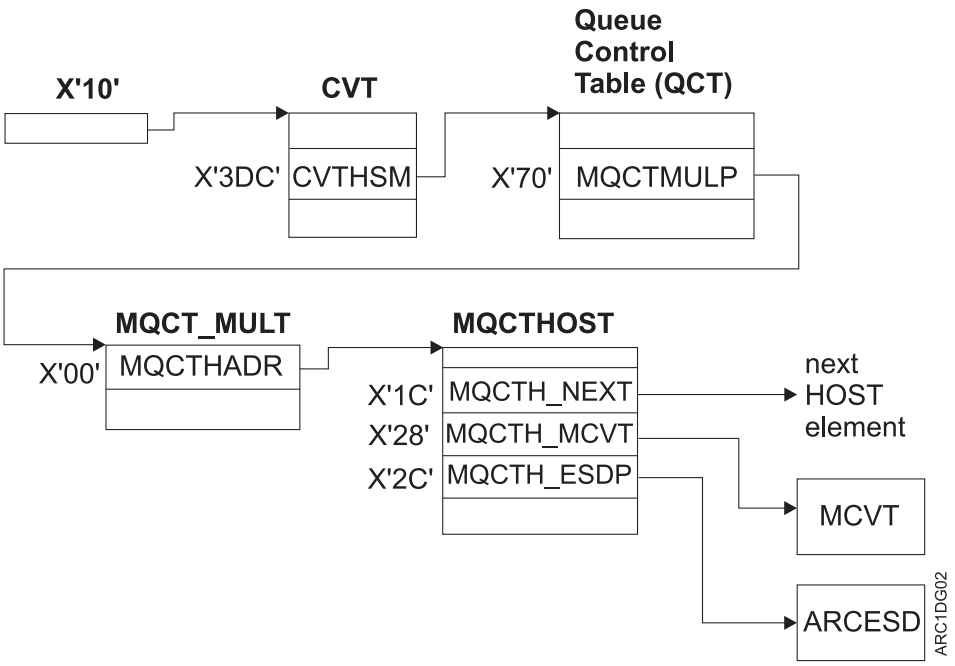


Figure 1. Locating the MCVT and Module ARCESD

When the low storage is not available, it is helpful to know that the MCVT follows the module ARCCBS, and that ARCESD and ARCCBS are ordered respectively as the first two modules of the DFSMShsm load module ARCCTL.

The first occurrence of the word MCVT is in ARCESD (see the following example), and the associated address gives the actual location of the MCVT.

All DFSMShsm modules and major DFSMShsm control blocks, such as the MCVT, the management control record (MCR), and the backup control record (BCR) are each pointed to by the ARCESD module.

Examples: These dump examples were obtained and processed by using IPCS:

- This example, a segment of a dump, shows the beginning of ARCESD. It contains eight-byte module name fields followed by corresponding four-byte address fields for each DFSMShsm module. By using this information, you can efficiently locate these modules within the dump. Note that even though most modules have their names at the beginning of the module, ARCESD does not.

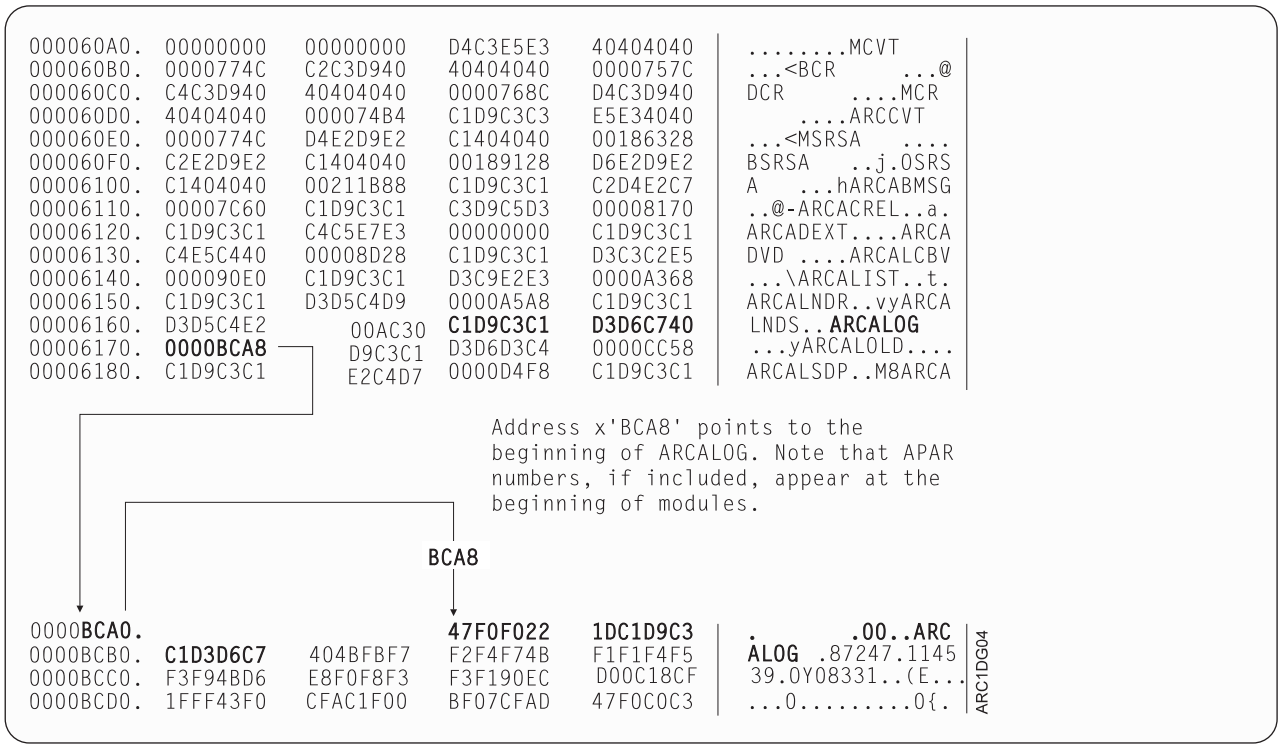
```
 Beginning of ARCESD──────────┐
                              ▼
000060A0.  00000000   00000000   D4C3E5E3   40404040 │ ........MCVT     │
000060B0.  0000774C   C2C3D940   40404040   0000757C │ ...<BCR      ...@ │
000060C0.  C4C3D940   40404040   0000768C   D4C3D940 │ DCR      ....MCR │
000060D0.  40404040   000074B4   C1D9C3C3   E5E34040 │      ....ARCCVT  │
000060E0.  0000774C   D4E2D9E2   C1404040   00186328 │ ...<MSRSA    .... │
000060F0.  C2E2D9E2   C1404040   00189128   D6E2D9E2 │ BSRSA    ..j.OSRS │
00006100.  C1404040   00211B88   C1D9C3C1   C2D4E2C7 │ A    ...hARCABMSG │
00006110.  00007C60   C1D9C3C1   C3D9C5D3   00008170 │ ..@-ARCACREL..a. │
00006120.  C1D9C3C1   C4C5E7E3   00000000   C1D9C3C1 │ ARCADEXT....ARCA │
00006130.  C4E5C440   00008D28   C1D9C3C1   D3C3C2E5 │ DVD ....ARCALCBV │
00006140.  000090E0   C1D9C3C1   D3C9E2E3   0000A368 │ ...\ARCALIST..t. │
00006150.  C1D9C3C1   D3D5C4D9   0000A5A8   C1D9C3C1 │ ARCALNDR..vyARCA │
00006160.  D3D5C4E2   0000AC30   C1D9C3C1   D3D6C740 │ LNDS....ARCALOG  │
00006170.  0000BCA8   C1D9C3C1   D3D6D3C4   0000CC58 │  ...yARCALOLD.... │
00006180.  C1D9C3C1   D3E2C4D7   0000D4F8   C1D9C3C1 │ ARCALSDP..M8ARCA │ ARC1DG03
```

- This example demonstrates how to locate a specific module in the dump (in this case ARCABNMB). The module name and its corresponding address are located in ARCESD, and then the leftmost column is scanned until this address is again located.

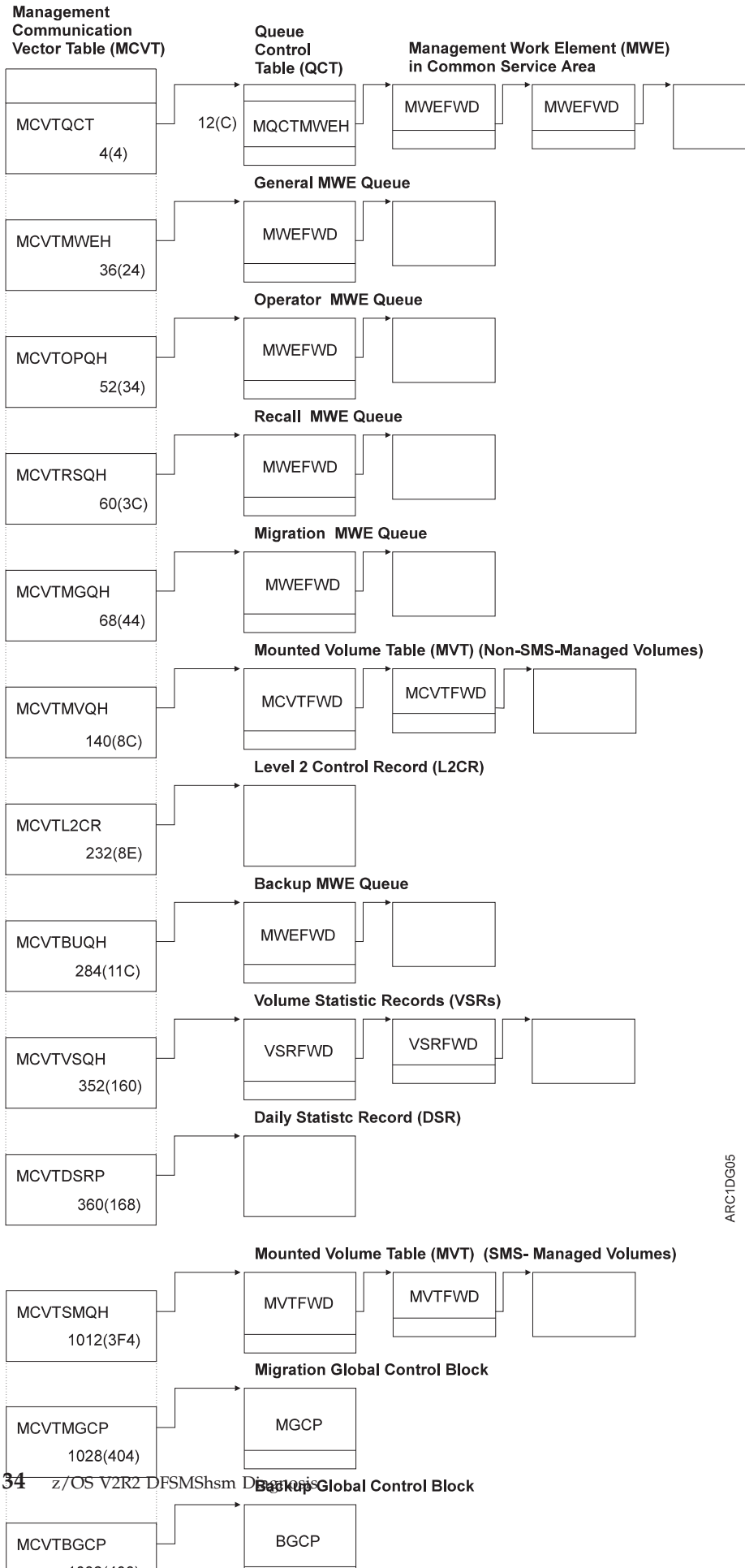
```
08100BC0.  D4C3E5E3   40404040   08103230   C1C2D9C3 8C │  MCVT    ....ABRC │
08100BD0.  C2404040   08106E68   C2C3D940   40404040    │  B   ..>.BCR      │
08100BE0.  081057E8   C3C3D340   40404040   08104774    │  ...YCCL     .... │
08100BF0.  C3D7E3C2   D3404040   081037F8   C4C3D940    │  CPTBL   ...8DCR  │
08100C00.  40404040   08106068   D4C3D940   40404040    │      ..-.MCR      │
08100C10.  08104FE8   C1D9C3C3   E5E34040   08103230    │  ..|YARCCVT  .... │
08100C20.  C1D9C3C1   C2D9C3C2   08106E68   D4E2D9E2    │  ARCABRCB..>.MSRS │
08100C30.  C1404040   00017AD0   C2E2D9E2   C1404040    │  A   ..:.BSRSA    │
08100C40.  00017658   D6E2D9E2   C1404040   000171E0    │  ....OSRSA   .... │
08100C50.  C2C7C3C2   40404040   08103918   C2E3C3D9    │  BGCB    ....BTCR │
08100C60.  E3404040   0810469C   C4C7C3C2   40404040    │  T   ....DGCB     │
08100C70.  08103A10   C5C7C3C2   40404040   08103A30    │  ....EGCB    .... │
08100C80.  D4C7C3C2   40404040   08103B28   E8C7C3C2    │  MGCB    ....YGCB │
08100C90.  40404040   081043C4   E2C7C3C2   40404040    │      ...DSGCB     │
08100CA0.  08104638   D1C7C3C2   40404040   08104B90    │  ....JGCB    .... │
08100CB0.  D8C3E340   40404040   00F92450   C4D4E5E2    │  QCT     .9.&DMVS │
08100CC0.  E3404040   00006048   C7D9E2C3   C2404040    │  T   ..-.GRSCB    │
08100CD0.  00F92E50   C1D9C3C1   C2C4C1E2   08156FB0    │  .9.&ARCABDAS..?. │
08100CE0.  C1D9C3C1   C2D4C3C2   08157BE8   C1D9C3C1    │  ARCABMCB..#YARCA │
08100CF0.  C2D4C9C7   081581C8   C1D9C3C1   C2D4E2C7    │  BMIG..aHARCABMSG │
08100D00.  08158CB8   C1D9C3C1   C2D5D4C2   08159348 ──┐│  ....ARCABNMB..1. │
                                                       ▼
                                  Address x'8159348' points to the
                                  beginning of ARCABNMB. Note that APAR
                                  numbers, if included, appear at the
                                  beginning of modules.

08159340.                         47F0F032   2DC1D9C3 8C │  ..F-.....00..ARC │
08159350.  C1C2D5D4   C24BF0F2   F1F7F04B   F2F0F4F1    │  ABNMB.02170.2041 │
08159360.  F1F84BC8   C4E9F1F1   C8F04B5C   40404040    │  18.HDZ11H0.*     │
08159370.  404040E5   C5D97EF1   C8F090EC   D00C18CF    │     VER=1H0.....  │ ARC1DG07
```

- Here is another example of how to locate a specific module in the dump (in this case ARCALOG).



The MCVT, in addition to storing flags, also contains pointers of work queues and chained lists. Figure 2 on page 34 shows some of the useful work queues and global data areas. Note that the MWEs in CSA (pointed to by MQCTMWEH) represent only those requests from the host started with HOSTMODE=MAIN.


Management Communication Vector Table (MCVT)
Queue Control Table (QCT)
Management Work Element (MWE) in Common Service Area
MCVTQCT
4(4)
12(C)
MQCTMWEH
MWEFWD
MWEFWD
General MWE Queue
MCVTMWEH
36(24)
MWEFWD
Operator MWE Queue
MCVTOPQH
52(34)
MWEFWD
Recall MWE Queue
MCVTRSQH
60(3C)
MWEFWD
Migration MWE Queue
MCVTMGQH
68(44)
MWEFWD
Mounted Volume Table (MVT) (Non-SMS-Managed Volumes)
MCVTMVQH
140(8C)
MCVTFWD
MCVTFWD
Level 2 Control Record (L2CR)
MCVTL2CR
232(8E)
Backup MWE Queue
MCVTBUQH
284(11C)
MWEFWD
Volume Statistic Records (VSRs)
MCVTVSQH
352(160)
VSRFWD
VSRFWD
Daily Statistc Record (DSR)
MCVTDSRP
360(168)
ARC1DG05
Mounted Volume Table (MVT) (SMS- Managed Volumes)
MCVTSMQH
1012(3F4)
MVTFWD
MVTFWD
Migration Global Control Block
MCVTMGCP
1028(404)
MGCP
Backup Global Control Block
MCVTBGCP
BGCP

Chapter 8. Diagnosing from return codes and reason codes

Many DFSMShsm error conditions generate nonzero return codes and reason codes. The return code usually identifies the error; the reason code provides additional detailed information about the problem.

When an error is reported in a DFSMShsm message, the return code is used to select the appropriate message. The return code is often used as part of the message identification number. For example, a return code of 5 can result in message ARC*nn*05I, where *nn* is a variable depending on the function performed. (For recall and recovery, *nn* is 11; for migration, *nn* is 12; for backup, *nn* is 13; for command processing, *nn* is 16; and for ABARS processing, *nn* is 60 or 61.) Each message explanation includes a description of any reason codes that may be associated with the message. For an explanation of each message, see *z/OS MVS System Messages, Vol 2 (ARC-ASA)*.

Return code processing

Selected return codes that result from DFSMShsm processing are given special attention by DFSMShsm. These codes are passed to the ARCERP module from the detecting DFSMShsm module, along with a request for a particular level of automatic error processing. m lists the modules that pass codes to ARCERP, the error codes, the error processing levels, and brief descriptions of the problem. The error processing levels are listed next in order of increasing severity:

Debug
No action is required unless the TRAP command has been entered for the codes.

Log Write a message to the log and the operator, then continue.

Snap Write a message to the log and the operator, generate a snap dump, and continue processing. The snap parameters are as follows:
- ONCE (default)
- ALWAYS
- NEVER

Abend
Write a message to the log and the operator, optionally generate an abnormal end dump, and end the task. The abend parameters are as follows:
- ONCE (default)
- ALWAYS
- NEVER

Fatal Write a message to the log and the operator, generate an abnormal end dump, and shut down DFSMShsm.

Part of the ARCERP error processing includes a test to check whether the return code contains an associated trap. The following traps are set by the TRAP command, which can be:
- Issued by the operator
- Issued by a DFSMShsm-authorized user using HSENDCMD
- Placed in the DFSMShsm startup member of PARMLIB

When a particular return code is passed to ARCERP, DFSMShsm examines both the requested error processing level passed by the calling routine and the level set by a possible TRAP command to determine the actual level of error processing used. In many cases, but not all, the TRAP command can be used to override the requested processing level as shown in Table 6 on page 130.

Note that in a few cases, a code returned from a module may have resulted from several conditions. For example, the ARCCOPEN module passes back OPEN macro return codes and VERIFY macro return codes. The return code itself does not identify which macro has generated it. In such cases, the dump that results from the error processing must be examined to find the offset pointer into the ARCCOPEN module from which the ARCERP call has been made. The ARCCOPEN source code immediately advances this offset and indicates whether an OPEN or a VERIFY macro has been attempted.

Also note that if an abnormal end occurs, the system abnormal end code is reported in decimal format. For example, an abnormal end 0C1 in ARCCATBU is listed as "ARCO205I TRAP" in module ARCCATBU for code 0193 by the QUERY TRAP command. To obtain a snap dump for this example, enter: TRAP ARCCATBU 193 SNAP

Table 3. Entries That Pass Error Codes to ARCERP

Error Code	Process Level	Description of Problem
Note: 1. Messages 200 and 208 generally come in pairs, indicating only one error. When there is a digit from 1 to 9 at the end of the entry name and you cannot find the entry with the last digit included, look at the entry without it. 2. If you receive an error code not in this list, contact your IBM support personnel.		
ARCABMSG		
4*nn*	Debug	Parameter list is not valid.
ARCACASS		
4*nn*	Debug	SASINDEX is not between 0 and 64.
ARCACREL		
1	Log	Mass Storage System acquire fails.
OBTAIN macro return code	Log	Cannot allocate a volume.
OBTAIN macro return code plus 100	Log	Cannot read format 5 DSCB.
ARCADVD		
4*nn*	Debug	Parameter list is not valid.
ARCALMVL		
4*nn*	Debug	Parameter list is not valid.
ARCALNDS		
16	Debug	Unexpected dynamic allocation error.
400	Debug	Parameter list is not valid.
ARCALOG		
4*nn*	Snap	Parameter list is not valid.
ARCALOLD		
8	Debug	Parameter list is not valid.
12	Debug	Dynamic allocation parameter list is not valid.
ARCALSDP		

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
1	Debug	VSA not found.
4	Debug	Allocation indicated SDSP in use.
6	Debug	All entries in processing unit ID array are in use.
24	Debug	Parameter list is not valid.
CDS read/update return code	Debug	CDS read/update return error.
DAIR return code	Snap	DAIR dynamic deallocation failed.
ARCALSNV		
4*nn*	Debug	Parameter list is not valid.
ARCALTDS		
400	Debug	Parameter list is not valid.
ARCALVOL		
4	Debug	Data set in use.
6	Debug	DASD volume in use.
8	Debug	Installation denied SVC 99 request.
12	Debug	Parameter list is not valid.
16	Debug	GETMAIN failed.
20	Debug	Unit name "3590-1" will not be used because its setup failed during DFSMShsm initialization.
27	Debug	Compaction incompatibility.
34	Debug	Received error from ARCALTDS while attempting to get addressability to JFCB.
DYNALLOC macro error reason code	Debug	Dynamic allocation fails.
ARCATIO		
0	Snap	Bad parameter. The reason code contains the number of data blocks remaining in the control unit buffer. This number should be 0; if not, terminate.
ARCATTEC		
4*nn*	Debug	Parameter list is not valid.
ARCAUCPP		
4*nn*	Debug	Parameter list is not valid.
ARCAUCP1		
4*nn*	Debug	Parameter list is not valid.
501	Debug	FRTV record update failure.
ARCAUCP2		
4*nn*	Debug	Parameter list is not valid.
ARCAUDBC		
6	Debug	No buffer provided for read.
ARCAUDBV		
6	Debug	No buffer provided for read.
7	Abend	JFCB read fails.
ARCAUDCP		

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
4*nn*	Debug	Parameter list is not valid.
ARCAUDDS		
nn	Debug	Catalog read failed. Parameter list is not valid.
ARCAUDMC		
6	Debug	No buffer provided for read.
ARCAUDSC		
6	Debug	No buffer provided for read.
ARCAUDTV		
6	Debug	No buffer provided for read.
ARCAUDVL		
7	Abend	JFCB read fails.
ARCAZSER		
8	Snap	Error reserving/releasing CDS.
4*nn*	Debug	Parameter list is not valid.
ARCBAKDS		
4*nn*	Debug	Parameter list is not valid.
ARCBATTC		
4*nn*	Debug	Parameter list is not valid.
ARCBCEBV		
4*nn*	Debug	Parameter list is not valid.
ARCBCLN		
4*nn*	Debug	Parameter list is not valid.
ARCBDNN		
4*nn*	Snap	Parameter list is not valid.
ARCBDSAM		
401	Debug	Parameter list is not valid.
ARCBDSBC		
4*nn*	Debug	Parameter list is not valid.
ARCBDSIP		
401	Debug	Parameter list is not valid.
ARCBDSMF		
401	Debug	Parameter list is not valid.
ARCBDSN		
401	Debug	Post code is not valid.
ARCBDSSP		
401	Debug	Parameter list is not valid.
402	Snap	Post request is not valid.
ARCBDSUX		
400	Debug	Parameter list is not valid.
401	Debug	Parameter list is not valid.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
402	Debug	Parameter list is not valid.
403	Debug	Parameter list is not valid.
404	Debug	Parameter list is not valid.
405	Debug	Parameter list is not valid.
406	Debug	Parameter list is not valid.
ARCBELIG		
4*nn*	Debug	Parameter list is not valid.
ARCBFULL		
400	Debug	MVT pointer is not valid.
ARCBFVOL		
8	Debug	MCT record indicates that a volume is not valid.
4	Debug	Error reading/writing BVR record.
20	Debug	BVR in use by another host.
ARCBGEN		
nn	Snap	Nonzero return code from ARCGMAIN.
ARCBKMSG		
4*nn*	Debug	Parameter list is not valid.
ARCBODS		
16	Debug	I/O error on input data set.
ARCBMBV		
nn	Debug	If *nnn* is greater than or equal to 100 and less than 200, subtract 100 from *nnn* and see message ARC13*nn*I for an explanation of the failed condition. Note that the leading zero in *nnn* is not used in the message identifier. If *nnn* is greater than or equal to 9*nn*, an abnormal end occurs when the return code is set to *nn*. See message ARC0003I for more information about the abnormal end. If the return code is greater than 0, see message ARC13*nn*I for the reason for the abnormal end.
ARCBPPDS		
20	Debug	See ARCBUDS explanation.
24	Debug	Error reading MCDS.
43	Debug	I/O error reading VTOC.
68	Debug	See ARCBUDS explanation.
70	Debug	See ARCBUDS explanation.
ARCBROLV		
4*nn*	Debug	Parameter list is not valid.
ARCBSPIL		
7	Debug	JFCB read fails.
4*nn*	Debug	Parameter list is not valid.
ARCBUDS		

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
nn	Debug	If *nnn* is greater than or equal to 100 and less than 200, subtract 100 from *nnn* and see message ARC13*nn*I for an explanation of the failed condition. Note that the leading zero in *nnn* is not used in the message identifier. If *nnn* is greater than or equal to 9*nn*, an abnormal end occurs when the return code is set to *nn*. See message ARC0003I for more information about the abnormal end. If the return code is greater than 0, see message ARC13*nn*I for the reason for the abnormal end.
ARCBVBEG		
97	Debug	DFSMShsm error.
4*nn*	Debug	Parameter list is not valid.
ARCBVDS		
nn	Debug	If *nnn* is greater than or equal to 100 and less than 200, subtract 100 from *nn* and see message ARC13*nn*I for an explanation of the failed condition. Note that the leading zero in *nnn* is not used in the message identifier. If *nnn* is greater than or equal to 9*nn*, an abnormal end occurs when the return code is set to *nn*. See message ARC0003I for more information about the abnormal end. If the return code is greater than 0, see message ARC13*nn*I for the reason for the abnormal end.
ARCBVDS1		
8	Abend	In preparing for an ENQ operation on a VSAM component, the length of that components name was found to be zero.
ARCBVEND		
4*nn*	Debug	Parameter list is not valid.
ARCBVOL		
7	Debug	Error reading JFCB.
400	Debug	Parameter list is not valid.
ARCBVR		
36	Debug	No volume entry associated with this BVR.
ARCBVR23		
6	Debug	No buffer provided for read.
ARCBZKEY		
48*n*	Snap	Parameter list not valid.
ARCCABRC		
nn	Abend	Nonzero return code from ARCGMAIN.
ARCCATBC		
4*nn*	Debug	Parameter list is not valid.
ARCCATBD		
4*nn*	Debug	Parameter list is not valid.
ARCCATBE		

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
nn	Snap	An abend occurred in the link macro. The reason code is found in register 1; the return code is 14.
4*nn*	Debug	Parameter list is not valid.
ARCCATBS		
12	Snap	Queued journal entries did not finish within allotted time. See ARC0799I explanation.
13	Fatal	The dynamic keyrange values in the MHCR could not be journaled after CDS backup.
405	Snap	CDS serialization error.
4*nn*	Debug	Parameter list is not valid.
ARCCATMS		
4*nn*	Log	Parameter list is not valid.
ESTAE return code	Log	ESTAE not established. See ESTAE macro documents for meanings of the return codes.
ARCCATRN		
4*nn*	Debug	Parameter list is not valid.
ARCCATZ		
4	Debug	An orphaned task entry was found in the CatTable during the ARCCAT release function. The entry is removed, and processing continues.
8	Debug	The ARCCAT release function was attempting to re-enqueue the ARCCAT resource on behalf of a task, but the task was already holding the resource. Processing continues.
12	Snap	An unexpected error occurred while attempting to re-enqueue the ARCCAT resource during the ARCCAT release function. ARCCAT release is disabled on the DFSMShsm host that issued this error until DFSMShsm is restarted. The first such error will result in a snap, and subsequent errors will result in debug-level traps.
16	Snap	The transition lock could not be obtained during the ARCCAT release function. ARCCAT release is disabled on the DFSMShsm host that issued this error until DFSMShsm is restarted.
20	Snap	The “pending CDS updates” counter did not reach zero in a reasonable time during the ARCCAT release function. ARCCAT release is disabled on the DFSMShsm host that issued this error until DFSMShsm is restarted.
24	Snap	A timeout occurred waiting for all tasks to release CDS resources in a MASH or MASH-eligible environment. ARCCAT release is disabled on the DFSMShsm host that issued this error until DFSMShsm is restarted.
ARCCBINT		
4*nn*	Debug	Parameter list is not valid.
ARCCBVR		

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
36	Debug	No volume entry associated with this BVR.
401	Log	Parameter list identifier is not valid.
402	Log	Incorrect BVR type is passed.
403	Log	Incorrect BVR indicator is passed.
404	Log	Incorrect hard log type is passed.
405	Log	Updates are not valid; BVR pointer is passed.
ARCCKBUP		
4*nn*	Debug	Parameter list is not valid.
ARCCKEOS		
36	Debug	Volume for RACF check of ERASE status of original data set not found.
40	Debug	Volume for RACF check of ERASE status of migrated data set not located.
44	Debug	The volume serial number returned from LOCATE was zero or null.
nn	Debug	Any nonzero return code from ARCRACF or ARCVSCHK.
4*nn*	Debug	Parameter list is not valid.
ARCCKRNT		
4*nn*	Debug	Parameter list is not valid.
ARCCKRNS		
4*nn*	Debug	Parameter list is not valid.
ARCCLUSZ		
4*nn*	Debug	Parameter list is not valid.
ARCCMHRD		
6	Debug	No buffer supplied for read.
ARCCOPEN		
98	Fatal	GETMAIN failed.
99	Snap	GENCB failed.
401	Debug	Parameter list is not valid.
OPEN macro return code	Snap	VSAM open fails on the backup control data set.
OPEN macro return code	Fatal	VSAM open fails on the migration control data set.
OPEN macro return code	Snap	VSAM open fails on the offline control data set.
VERIFY macro return code	Fatal	VSAM verify fails on the migration control data set.
ARCCPCTS		
1	Abend	End-of-volume-type EXLST ENTRY not found.
ARCCPDDV		
400	Debug	Volume serial number is not valid.
ARCCPDEF		
16	Snap	Read or write error accessing the level 2 control record or volume record for level 2 volumes.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
ARCCPDIS		
4	Debug	Tag not found.
ARCCPLM3		
4*nn*	Debug	Parameter list is not valid.
ARCCPLMC		
52	Debug	FREEMAIN failure.
ARCCPOP		
999	Abend	DFSMShsm STOP command issued with the DUMP parameter.
ARCCPPRI		
100	Debug	Error establishing ESTAE environment.
ARCCPQST		
nn greater than 4	Snap	Error reading parameter list.
ARCCPRCY		
nn	Debug	Message ARC0445I expecting return code 24, but did not receive it.
ARCCPRTN		
6	Debug	No buffer passed for read request.
ARCCPSEM		
12	Debug	The SETSYS EMERGENCY or SETSYS NOEMERGENCY command was issued on a host and secondary host promotion was not able to notify other hosts in the sysplex of the status change because the associated control blocks were not available. Try to issue the command again.
ARCCPSES		
321	Debug	Unable to obtain memory for 3590-1 UCB.
322	Debug	Other error creating list of 3590-1 UCB.
4*nn*	Debug	Parameter list is not valid.
ARCCPSET		
12	Debug	The SETSYS SSMSTART(*hhmm1 hhmm2*) command was issued on a host, and secondary host promotion was not able to notify other hosts in the sysplex of the status change because the associated control blocks were not available. Try issuing the command again.
ARCCRCB		
16	Abend	Either the size of the RCB changed or the number of data sets specified (NDSE) by the caller is greater than 4. If the length of the RCB or the number of data sets causing this problem are correct then the size of the RCB cells must be increased to accommodate this new size.
ARCCTAMS		
400	Debug	Invalid macro identifier

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
401	Debug	Invalid management class
402	Debug	Invalid data set name
403	Debug	Invalid function
404	Debug	Invalid LSCTD field pointer
ARCCTDSS		
400	Debug	Invalid macro ID for ARCCTDSP
401	Debug	Invalid pointer to SMSWA
402	Debug	Invalid data set name
403	Debug	Invalid pointer to RCB
404	Debug	Invalid copy technique
ARCCTELG		
400	Debug	Incorrect parmlist ID
401	Debug	Invalid pointer to input SMSWA
402	Debug	Invalid pointer to output SMSWA
403	Debug	Invalid current date specified
404	Debug	Invalid management class definition
405	Debug	Invalid F1/F8 DSCB pointer
ARCCTL		
1	Snap	Management work element is not valid. queue element.
4	Debug	Improper authorization to issue a command.
12	Snap	Request is not valid.
16	Snap	Request is not valid.
20	Snap	Request is not valid.
24	Snap	Request is not valid.
28	Snap	Request is not valid.
40	Snap	Request is not valid.
52	Snap	Request is not valid.
nn	Snap	Unknown error.
ARCCTLTK		
401	Debug	Parameter list is not valid.
ARCCTRAR		
400	Debug	Invalid data set name
401	Debug	Invalid pointer to input SMSWA
402	Debug	Invalid pointer to output SMSWA
403	Debug	Invalid pointer to storage group list
ARCCTUIM		
400	Abend	Pointer to the ADREIDB is not specified
401	Abend	ADREIDB control block ID is not valid
402	Abend	Pointer to ARCCTDSP is not specified

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
403	Abend	ARCCTDSP control block ID is not valid
404	Debug	Pointer to the RCB is not specified
405	Debug	Pointer to the FSR is not specified
406	Debug	CTDS_DSN is not valid or passed
407	Debug	CTDS_SYSINNUM is not valid or passed
408	Debug	Any message anchor pointer is not valid or passed
409	Debug	Message buffer blocksize or subpool is not valid or passed
EIOPTION	Debug	Called each time ARCCTUIM is entered when patchable MGCB_CTUIM field has been patched to request debug messages. (Refer to DFSMSdss processing).
EIOPTION	Abend	ARCCTUIM returned an option to DFSMSdss that is not valid.
EIRETCOD	Debug	Called each time ARCCTUIM is exited when patchable MGCB_CTUIM field has been patched to request debug messages. (Refer to DFSMSdss processing).
ARCCUIM		
4*nn*	Abend	Parameter list is not valid.
ARCCUIM1		
EIOPTION	Debug	DSS processing option is not valid.
ARCCUMC1		
6	Debug	No buffer provided for reading MC1 record.
ARCCVSR		
1	Snap	Broken VSR chain encountered.
2	Abend	GETMAIN for VSR failed.
ARCDADCE		
900	Debug	Processing of expired dump copy failed.
ARCDATAM		
40	Debug	Work-to-do code is not valid.
4*nn*	Debug	Parameter errors.
ARCDATTC		
8	Debug	Attach failed.
400	Debug	Parameter list is not valid.
ARCDAUTO		
400	Debug	Parameter list is not valid.
ARCDBAUT		
52	Debug	ARCGMAIN return code.
ARCDCLOS		
4	Debug	FREEPOOL failure.
8	Debug	Nonzero return code is in register 15 after processing the CLOSE macro.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
12	Debug	CLOSE abnormal end X'14' occurs during processing of the CLOSE macro.
16	Debug	Undetermined abnormal end occurs during the processing of the CLOSE macro.
20	Debug	Abnormal end occurs during processing of the CLOSE macro.
42	Debug	Partial release failed.
4*nn*	Debug	Parameter list is not valid.
ARCDCMD		
nn	Snap	Nonzero return code from ARCGMAIN.
400	Debug	The ECB list was not passed.
ARCDCOLL		
4*nn*	Debug	Parameter list is not valid.
ARCDEDVC		
900	Debug	An abend occurred during processing of excess dump VTOC copy data sets.
ARCDGEN		
nn (post code)	Debug	Post code is not valid or dump level ECB is not valid.
ARCDGVSZ		
4*nn*	Debug	Parameter list is not valid.
ARCDIDC		
4*nn*	Debug	Parameter list is not valid.
ARCDLVDV		
4*nn*	Debug	Parameter list is not valid.
ARCDMDS		
nn	Debug	RACF delete failed.
400	Debug	Parameter list is not valid.
ARCDOPEN		
400	Debug	Parameter list is not valid.
ARCDRDSS		
4*nn*	Debug	Parameter list is not valid.
ARCDRSTM		
8	Debug	Failure in the attach macro.
16	Debug	Abnormal end in the ARCDRTMT subtask.
4*nn*	Debug	Parameter list is not valid.
ARCDRSYN		
4*nn*	Debug	Parameter list is not valid.
ARCDRTMT		
400	Debug	Parameter list is not valid.
401	Debug	Zero UCB pointer.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
999	Snap	No match between tape request and the active TCB.
ARCDTTRM		
400	Debug	Input parameter contains a value that is not valid. The parameters are the wrong length.
ARCDTVSV		
400	Debug	OEVSE pointer is zero.
401	Debug	MVT pointer is 0.
ARCDUDVL		
4	Debug	I/O error reading or writing DVL record or error adding the volume to RACF.
4*nn*	Debug	Parameter list is not valid.
ARCDUUIM		
4*nn*	Debug	Parameter list is not valid.
EIOPTION	Debug	Called each time that ARCDUUIM is entered when this module has been patched to request debug messages. (From module ARCDIUM1.)
EIOPTION	Abend	ARCDUUIM returns an option to DFSMSdss that is not valid. (From module ARCDIUM2.)
EIOPTION	Abend	The DDNAME passed to this installation exit does not match any of the DDNAMEs that DFSMShsm has recorded in the output MVT records. (From module ARCDIUM3.)
EIRETCOD	Debug	Called each time that ARCDUUIM is exited when this module has been patched to request debug messages. For an explanation of EIOPTION and EIRETCOD (from module ARCDIUM4), see *z/OS DFSMSdss Storage Administration*.
ARCDVBEG		
4*nn*	Debug	Parameter list is not valid.
ARCDVCLN		
4*nn*	Debug	Parameter list is not valid.
900	Debug	The dump volume processing ends abnormally.
ARCDVOL		
4*nn*	Debug	Parameter list is not valid.
Return code from DVL read	Abend	Error reading DVL record.
ARCESTAE		
nn	Debug	The SDWA abend code is not a X'37' type, and the force re-entry flag is on.
ARCFCRE		
4*nn*	Abend	Parameter list is not valid.
ARCFDEL		
4*nn*	Abend	Parameter list is not valid.
ARCFDIS		
4*nn*	Abend	Parameter list is not valid.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
ARCFDDSS		
4*nn*	Debug	Parameter list is not valid.
ARCFDMCL		
4*nn*	Abend	Parameter list is not valid.
ARCFDUIM		
4*nn*	Debug	Parameter list is not valid.
ARCFEXP		
4*nn*	Abend	Parameter list is not valid.
ARCFNEWK		
4*nn*	Abend	Parameter list is not valid.
ARCFPAT		
4*nn*	Abend	Parameter list is not valid.
ARCFRBM		
4	Debug	FRD record was not found for a Fast Replication backup version for which dump copies were indicated.
8	Snap	Undefined FRB_STATE (for ARC0200I, ARC0205I, ARC0208I, and ARC0900I)
ARCFRCDM		
5	Debug	Error closing a catalog information data set.
6	Debug	Error de-allocating a catalog information data set.
9	Debug	Catalog error encountered.
10	Debug	Error uncataloging a catalog information data set.
11	Debug	Error scratching a catalog information data set.
52	Debug	GETMAIN or FREEMAIN error encountered.
4*nn*	Snap	Parameter list is not valid.
ARCFRCSI		
4*nn*	Debug	Parameter list is not valid.
ARCFRCRW		
8	Debug	Error reading the catalog information data set.
12	Debug	Error writing to the catalog information data set.
16	Debug	Abend X'37' encountered while processing catalog information data set.
52	Debug	GETMAIN or FREEMAIN error encountered.
4*nn*	Snap	Parameter list is not valid.
ARCFRDMP		
4*nn*	Debug	Parameter list is not valid.
ARCFRDSD		
4*nn*	Debug	Parameter list is not valid.
ARCFRDSM		
4*nn*	Debug	Parameter list is not valid.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
ARCFRDSS		
4*nn*	Debug	Parameter list is not valid.
ARCFRDSU		
4*nn*	Debug	Parameter list is not valid.
ARCFRDSV		
4*nn*	Debug	Parameter list is not valid.
ARCFRDUT		
4*nn*	Debug	Parameter list is not valid.
ARCFRDVS		
4*nn*	Debug	Parameter list is not valid.
ARCFREE		
4	Snap	DAIR parameter list is not valid.
12	Snap	DAIR dynamic allocation fails.
ARCFRGDU		
400	Snap	Macro ID does not match.
401	Snap	MWE pointer not passed.
402	Snap	CPREL pointer is null.
403	Snap	FRREL pointer is null.
ARCFRGRU		
400	Snap	Macro ID does not match.
401	Snap	MWE pointer is null.
402	Snap	Record type is not valid.
403	Snap	Record key not passed.
404	Snap	Source volume serial number not passed.
405	Snap	Version not passed.
406	Snap	Caller not passed.
407	Snap	Volume serial number in FRVP not found.
408	Snap	Version in FRB not found.
ARCFRMCH		
4*nn*	Debug	Parameter list is not valid.
ARCFRMSG		
4*nn*	Debug	Parameter list is not valid.
ARCFRPAR		
4*nn*	Debug	Parameter list is not valid.
ARCFRRM		
12	Debug	CDS I/O error.
4*nn*	Debug	Parameter list is not valid.
ARCFRMC2		
400	Debug	Parameter list is not valid.
401	Debug	FRVP pointer is null.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
402	Debug	FRB pointer is null.
403	Debug	Storage group name is null.
404	Debug	RCB pointer is null.
405	Debug	SMS lock is null.
ARCFRSDM		
400	Debug	Requested function is not valid.
401	Debug	Source volume serial number not passed.
402	Debug	Target volume serial number not passed.
403	Debug	Volume list pointer not passed.
404	Debug	Source and target volume serial numbers not passed.
405	Debug	FlashCopy® target list area not passed.
ARCFRTM		
400	Debug	Internal error.
401	Debug	Internal error.
ARCFRUIM		
400	Debug	Pointer to the EIDB not passed.
401	Debug	Addressability could not be established to the ARCFRTCB data area.
EIOPTION	Debug	ARCFRUIM entered when this module has been patched to request debug messages (Refer to DFSMSdss processing).
EIOPTION	Debug	ARCFRUIM returned an option to DFSMSdss that is not valid.
EIRETCOD	Debug	ARCFRUIM exited when this module has been patched to request debug messages (Refer to DFSMSdss processing).
ARCFRUPD		
4*nn*	Debug	Parameter list is not valid.
ARCFRUTL		
100	Debug	Error during enqueue/dequeue.
4*nn*	Debug	Parameter list is not valid.
ARCFRVDV		
400	Debug	Parameter list is not valid.
401	Debug	MWE pointer not found.
402	Debug	RCB pointer not found.
ARCFRVOL		
4*nn*	Debug	Parameter list is not valid.
ARCFVER		
4*nn*	Debug	Parameter list is not valid.
ARCGATTC		
4*nn*	Debug	Parameter list is not valid.
ARCGCLN		

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
400	Debug	Input parameter passed to ARCGCLM is not valid.
ARCGDS		
4	Debug	Target UCB not found.
400	Snap	Parameter address that was passed is not valid or parameter list not set properly.
410	Snap	Recovery task control block (GTCB) that was passed is not valid.
4*nn*	Debug	Parameter list is not valid.
nn	Debug	See the return codes for RECALL and RECOVERY functions for ARC11*nn* error message table in *z/OS MVS System Messages, Vol 2 (ARC-ASA)* for more information.
ARCGDSN		
18	Snap	Error processing DAOPTION. If the reason code is 30, no DVT entry was returned to the original volume type. If the reason code is 34, there is no MCC for the recover operation (as opposed to the restore operation).
4*nn*	Debug	Parameter list is not valid.
ARCGDSRV		
400	Snap	GTCB index is not valid.
ARCGDSSC		
4*nn*	Debug	Parameter list is not valid.
ARCGDS1		
nn	Debug	Nonzero return code from ARCTCLOS when tape data set close fails. See ARCTCLOS explanation.
ARCGDS2		
nn	Debug	Nonzero return code from ARCDCLOS when DASD data set close fails. See ARCDCLOS explanation.
ARCGDS3		
Scratch macro return code	Debug	Nonzero return code from ARCZSCR when scratching replaced data set.
ARCGENLG		
400	Debug	Activity log type is not valid; input parameter is not valid.
ARCGETCH		
4*nn*	Debug	Parameter list is not valid.
ARCGMAIN		
52	Debug	FREEMAIN failed.
52	Snap	GETMAIN failed.
ARCGMCC		
400	Debug	Input MWE pointer isn't valid.
ARCGODS		
12	Snap	I/O error reading migration copy.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
13	Snap	I/O error on restore copy.
52	Debug	Error freeing virtual storage.
ARCGRACS		
4*nn*	Debug	Parameter list is not valid.
ARCGRAIN		
4*nn*	Debug	Parameter list is not valid.
ARCGRCAT		
4*nn*	Debug	Parameter list is not valid.
ARCGRCLN		
400	Debug	Input parameter that was passed is not valid.
ARCGRDCL		
4*nn*	Debug	Parameter list is not valid.
ARCGRDGN		
4*nn*	Debug	Parameter list is not valid.
ARCGRDPR		
400	Debug	Non-valid index to GTCB was passed as a parameter by the caller.
ARCGRDS		
nn	Debug	See the return codes for RECALL and RECOVERY functions for ARC11*nn* error message table in *z/OS MVS System Messages, Vol 2 (ARC-ASA)* for more information.
4*nn*	Debug	Parameter list is not valid.
900	Debug	DFSMSdss abnormal end.
ARCGRDVC		
4	Debug	Error in reading MCV/MCT record.
8	Debug	Error in allocating the dump VTOC copy data set.
14	Debug	Error in reading the dump VTOC copy data set.
16	Debug	Error in opening the dump VTOC copy data set.
24	Debug	An non-valid or unsupported device type is found for the source backup or dump volume in its control data set.
38	Debug	No VTOC copy data set entry is found for the requested data set.
56	Debug	For a VSAM data set restore of a cluster, the data component name is not found in the dump VTOC copy data set.
58	Debug	For a VSAM data set restore of a cluster, the index component name is not found in the dump.
ARCGRDVL		
400	Debug	The macro ID is not valid.
401	Debug	The caller passed an index for the GTCB that is not valid.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
ARCGREN		
4*nn*	Debug	Parameter list is not valid.
ARCGRFRV		
400	Debug	Parameter list is not valid.
ARCGRGDG		
54	Debug	Abend of user exit during recovery of GDG base entry.
68	Debug	Point macro failed during recovery of GDG base entries from a single file format tape.
4*nn*	Debug	Parameter list is not valid.
ARCGRLOC		
52	Debug	GETMAIN failure in ARCZQBLD.
4*nn*	Debug	Parameter list is not valid.
ARCGRMAN		
4*nn*	Debug	Parameter list is not valid.
ARCGRQE		
4*nn*	Debug	Parameter list is not valid.
ARCGRQE1		
400	Debug	Macro ID is not valid.
401	Debug	Caller passed an index for the GTCB that is not valid.
ARCGRQE2		
400	Debug	Macro ID is not valid.
401	Debug	Caller passed an index for the GTCB that is not valid.
ARCGRQE3		
4*nn*	Debug	Parameter list is not valid.
ARCGRTVR		
430	Debug	Input parameter is not valid.
ARCGRVDS		
nn	Debug	See the return codes for RECALL and RECOVERY functions for ARC11*nn* error message table in *z/OS MVS System Messages, Vol 2 (ARC-ASA)* for more information.
4*nn*	Debug	Parameter list is not valid.
ARCGRVOL		
4*nn*	Debug	Parameter list is not valid.
ARCGRVSU		
5	Debug	Failure to allocate the target SMS volume.
19	Debug	Failure to read base data component data set VTOC entry.
46	Debug	Error in reading data set VTOC entry.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
4*nn*	Debug	Parameter list is not valid.
ARCGSCHD		
4*nn*	Debug	Parameter list is not valid.
ARCGSORR		
400	Snap	ID not correct.
401	Snap	Function is out of range.
402	Snap	GTCB index is not within range.
ARCGSTMP		
4*nn*	Debug	Parameter list is not valid.
ARCGTDS		
402	Debug	Macro ID is not valid.
ARCGVCAT		
52	Debug	FREEMAIN failure.
nn	Debug	JFCB read failure.
ARCGVCRT		
4*nn*	Debug	Parameter list is not valid.
ARCGVDS		
4	Debug	Error in finding SMS target volume UCB pointer after import of base cluster.
5	Debug	Error in allocating SMS target volume after import of base cluster.
14	Debug	LOCATE failure finding data object name for updating data set VTOC entry.
19	Debug	ARCZUF1 fails to update the base data component data set VTOC entry.
nn	Debug	See the return codes for RECALL and RECOVERY functions for ARC11*nn* error message table in *z/OS MVS System Messages, Vol 2 (ARC-ASA)* for more information.
100	Debug	Catalog search interface error.
101	Debug	Error retrieving data set information.
4*nn*	Debug	Parameter list is not valid.
ARCGVDSN		
400	Debug	The index to GTCB is not valid.
ARCGVDS1		
nn	Debug	Nonzero return code from ARCTCLOS when tape data set close fails. See ARCTCLOS explanation.
ARCGVDS2		
nn	Debug	Nonzero return code from ARCDCLOS when tape data set close fails. See ARCDCLOS explanation.
ARCGVDS3		
14	Debug	LOCATE error trying to find a data object name for update of VTOC entry.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
LOCATE macro return code	Debug	LOCATE failure on base cluster name before data set VTOC entry update.
ARCGVOL		
4*nn*	Debug	Parameter list is not valid.
ARCGVTOC		
6	Snap	JFCB read fails.
400	Debug	Parameter list is not valid.
ARCICTL1		
nn	Debug	Return code from ARCZWRIT; write error.
ARCICTL2		
nn	Debug	Return code from ARCZWRIT; write error.
ARCICTL3		
nn	Debug	Return code from ARCWRITE; write macro.
ARCILOG		
98	Snap	Incompatible serialization techniques.
99	Fatal	The CDS serialization techniques defined for one z/OS image are incompatible with the CDS serialization techniques defined for another z/OS image in a multiple DFSMShsm-host environment.
900	Snap	Abnormal end occurred.
ARCINIT		
HSM SVC return code	Snap	HSM SVC error initializing DFSMShsm.
ARCISTAT		
READ macro return code	Debug	Failure when reading DSR record to storage.
WRITE macro return code	Debug	Failure when writing DSR record to DASD.
ARCISTA1		
nn greater than 4	Debug / Log	Return code from ARCZREAD; read error.
ARCISTA2		
nn	Debug	Return code from ARCZWRIT; write error.
nn	Log	Return code from ARCZUPDT; update error.
ARCITTOC		
4*nn*	Debug	Parameter list is not valid.
ARCJCFCM		
4*nn*	Debug	Parameter list is not valid.
ARCJCFLK		
nn	Snap	Unexpected return code from coupling facility.
400	Debug	Parameter list is not valid.
401	Debug	Parameter list is not valid.
402	Debug	Parameter list is not valid.
403	Debug	Parameter list is not valid.
404	Debug	Failure obtaining lock.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
ARCJGFCL		
4*xx*	Debug	Parameter list is not valid
ARCJGFCM		
4*xx*	Debug	Parameter list is not valid
001	Debug	Attach of ARCJGRF failed, attach retcode in diagnostic code
002	Debug	ARCJGRF task already attached
003	Debug	Load of an common queue exit failed
004	Debug	Getmain for ?IXCJOIN work area failed
005	Debug	Getmain for ?IXCQUERY work area failed
006	Debug	?IXCJOIN failure
007	Debug	Getmain for gxmcb (member control block) failed
008	Debug	Getmain for ?IXCQUERY needed area length failed
009	Debug	?IXCQUERY failure
010	Debug	?IXCLEAVE failure
011	Debug	?IXCSETUS failure
ARCJGFCX		
4*xx*	Debug	Parameter list is not valid
ARCJGRF		
001	Debug	MWE_Submit error
002	Debug	MWE_Assigned error
003	Debug	MWE_Work_Completed error
004	Debug	MWE_Command_Completed err
005	Debug	MVT_Assigned error
006	Debug	MVT_Work_Completed error
007	Debug	Auto dump sch error
008	Debug	Auto dump start error
009	Debug	Auto dump completed error
010	Debug	Dump task should be available but it is not
ARCJMSND		
4*nn*	Debug	Parameter list is not valid
001	Debug	Master Scheduler not enabled in group
002	Debug	Host not connected to the group
003	Debug	Function held on this host
004	Debug	All hosts have function held
006	Debug	IXCMSGOX failure. RC is IXCMSGOX RC, reason code is the IXCMSGOX RSC.
007	Debug	Getmain failure for message buffer. Getmain RC is the diagnostic code.
009	Debug	Target host not connected in group
ARCJRALP		

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
4	Debug	HSM message error.
8	Debug	Adjunct area is not valid.
ARCJRAUD		
4*nn*	Debug	Parameter list is not valid.
ARCJRCLN		
400	Debug	Parameter list is not valid.
ARCJRELP		
4*nn*	Debug	Parameter list is not valid.
ARCJRENT		
4*nn*	Debug	Parameter list is not valid.
ARCJREP		
4	Debug	DFSMShsm connected to the CRQ structure, but the connection token was not set within the time allowed.
4*nn*	Debug	Parameter list is not valid.
ARCJRQRY		
4*nn*	Debug	Parameter list is not valid.
ARCJRSEL		
4*nn*	Debug	Parameter list is not valid.
ARCJRSM		
400	Debug	Input is not valid.
ARCJRUTL		
4*nn*	Debug	Parameter list is not valid.
ARCLBUC		
4*nn*	Debug	Parameter list is not valid.
ARCLDCLS		
1	Debug	I/O error. Reason code=1 for ARCZPOS; reason code=2 for ARCZNXT; reason code=3 for ARCZREAD.
2	Debug	List held, or DFSMShsm is shut down.
4	Debug	No dump class record found.
4*nn*	Debug	Parameter list is not valid.
ARCLDVOL		
4*nn*	Debug	Parameter list is not valid.
ARCLFR		
8	Debug	BCDS record is not valid.
4*nn*	Debug	Parameter list is not valid.
ARCLFRBG		
4*nn*	Debug	Parameter list is not valid.
ARCLFRCD		
404	Debug	Internal data inconsistency.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
4*nn*	Abend	Parameter list is not valid.
ARCLFRDM		
4*nn*	Debug	Parameter list is not valid.
ARCLSPAC		
1	Debug	LSPACE macro failed.
4	Debug	Volume not found.
8	Debug	Error processing VTOC.
12	Debug	OBTAIN encountered a permanent I/O error.
20	Debug	Volume type not supported.
24	Debug	Obtain macro error.
28	Debug	Allocation error.
32	Debug	Deallocation error.
4*nn*	Debug	Parameter list is not valid.
ARCMBBUF		
123	Debug	End of volume abend because of CAPACITYMODE error.
124	Debug	End of volume CAPACITYMODE error on original.
125	Debug	End of volume CAPACITYMODE error on alternate.
126	Snap	Bad registers were detected.
ARCMBFSR		
400	Debug	Parameter list or FSR pointer are not valid.
ARCMBUIM		
400	Abend	Pointer to the EIDB not specified, or the control block identifier is not valid.
401	Abend	Pointer to the ARCMDSSP could not be determined because EIUSEPTR was 0.
402	Abend	The ARCMDSSP control block identifier is not valid (MDSS-NAME).
4*nn* greater than 402	Debug	Parameter list is not valid.
EIOPTION	Abend	DFSMSdss processing option.
ARCMCCHK		
400	Snap	MTCB index is not valid.
401	Snap	MTCB block ID is not valid.
406	Snap	MTCB address in MTCB is not valid.
407	Snap	CDS entry type is not valid.
ARCMCDST		
4	Debug	Error locking the MTCB for concurrent update. The update to the MTCB is skipped and processing continues.
4*nn*	Debug	Parameter list is not valid.
ARCMCKF1		
4*nn*	Debug	Parameter list is not valid.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
ARCMCLN		
400	Debug	Index to migration task (MTCB) array is not valid.
ARCMCLNI		
4*nn*	Debug	Parameter list is not valid.
ARCMCLUS		
4*nn*	Debug	Parameter list is not valid.
ARCMCPSV		
400	Debug	MWE pointer is not valid.
ARCMCTL		
8	Snap	Getmain error trying to obtain storage for the migration subtasking control block. Migration subtasking is disabled for this DFSMShsm host until it is restarted and storage for this control block can be successfully obtained.
21	Snap	MHCR record not found. MCDS data set may be damaged. Check MCDS and correct any errors.
ARCMCVLT		
4*nn*	Debug	Parameter list is not valid.
ARCMDSMV		
nn	Debug	See the return codes for MIGRATION functions for ACR12*nn* error message table in *z/OS MVS System Messages, Vol 2 (ARC-ASA)* for more information.
ARCMDSN		
409	Log	Incorrect combination of DS1RECAL and DS1IND08 flags in the Format 1 DSCB.
ARCMDSS		
14	Debug	Error deleting SDSP data records.
52	Debug	GETMAIN failed.
60	Debug	Error establishing ESTAE environment.
65	Debug	Error closing output data set.
68	Snap	Error in DFSMSdss processing.
69	Debug	SYNCDEV macro error.
76	Snap	Error in link macro.
77	Debug	Failure to allocate DD dummy.
4*nn*	Debug	Parameter list is not valid.
ARCMDSUV		
nn	Debug	See the return codes for MIGRATION functions for ACR12*nn* error message table in *z/OS MVS System Messages, Vol 2 (ARC-ASA)* for more information.
100	Debug	Failure dequeueing GDS data set
ARCMDSUX		
4*nn*	Debug	Parameter list is not valid.
ARCMELIG		

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
4*nn*	Debug	Parameter list is not valid.
ARCMETRG		
400	Debug	MTCB index is not valid.
401	Debug	The index to MTCB reference is not a valid MTCB.
ARCMEXP		
4*nn*	Debug	Parameter list is not valid.
ARCMFVV		
400	Debug	Parameter list is not valid.
401	Snap	Post code is not valid.
ARCMFVVA		
400	Snap	Passed macro ID is not valid.
401	Snap	No requested function was passed.
402	Snap	Passed MTCB index is not valid.
403	Snap	The MFVV_POSTCODE is nonzero but the MFVV_POST flag is off.
ATTACH return code	Snap	Error returned from ATTACH macro when a subtask attaching has failed.
ARCMGLM		
400	Debug	Index is not valid (must be between 1 and 16).
ARCMGLO		
400	Debug	Index to MTCB that is passed is not valid (must be between 1 and 16).
ARCMG2TP		
400	Debug	Parameter list not set up correctly. This is an indicator that the caller did not set up parameter lists correctly.
nn greater than 900	Abend	Migration to tape error.
ARCMIGDS		
4*nn*	Debug	Parameter list is not valid.
ARCMLCLN		
nn	Debug	Error returned from FREEMAIN macro when you free storage for mounted volume tables for migration level 2 volumes.
400	Debug	Wrong TCB passed.
401	Debug	No RCB address passed.
ARCMMCHK		
400	Snap	Single task SSM mode that was passed is not valid.
401	Snap	MTCB index isout of range.
402	Snap	No pointer to the recovery control block.
403	Snap	Subtask name is not valid.
404	Snap	MCD address in MTCB in not valid.
405	Snap	Migration CDS entry type is not valid.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
ARCMMCLD		
400	Snap	MTCB block index is not valid.
401	Snap	MTCB block ID is not valid.
404	Snap	MSMCV block pointer is not valid.
405	Snap	MSMCV block ID is not valid.
406	Snap	MCD address in MTCB is not valid.
407	Snap	CDS entry type is not valid.
ARCMMCLI		
400	Snap	MTCB block index is not valid.
401	Snap	MTCB block ID is not valid.
404	Snap	MSMCV block pointer is not valid.
405	Snap	MSMCV block ID is not valid.
406	Snap	MCD address in MTCB is not valid.
407	Snap	CDS entry type is not valid.
408	Snap	SMQE address that was passed to this module is not valid.
409	Snap	SMQE block ID is not valid.
ARCMMCLM		
400	Snap	MTCB block index is not valid.
401	Snap	MTCB block ID is not valid.
402	Snap	No pointer to the recovery control block.
404	Snap	MSMCV block pointer is not valid.
405	Snap	MSMCV block ID is not valid.
406	Snap	MCD address in MTCB is not valid.
407	Snap	CDS entry type is not valid.
ARCMMCLN		
400	Snap	MTCB block index is not valid.
401	Snap	MTCB block ID is not valid.
402	Snap	No pointer to the recovery control block.
403	Snap	RCBID information is not valid.
404	Snap	MSMCV block pointer is not valid.
405	Snap	MSMCV block ID is not valid.
ARCMMCLR		
400	Snap	MTCB block index is not valid.
401	Snap	MTCB block ID is not valid.
402	Snap	No pointer to the recovery control block.
404	Snap	MSMCV block pointer is not valid.
405	Snap	MSMCV block ID is not valid.
406	Snap	MCD address in MTCB is not valid.
407	Snap	CDS entry type is not valid.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
ARCMMCLS		
400	Snap	MTCB block index is not valid.
401	Snap	MTCB block ID is not valid.
402	Snap	No pointer to the recovery control block
404	Snap	MSMCV block pointer is not valid.
405	Snap	MSMCV block ID is not valid.
406	Snap	MCD address in MTCB is not valid.
407	Snap	CDS entry type is not valid.
ARCMMCLT		
400	Snap	MTCB block index is not valid.
401	Snap	MTCB block ID is not valid.
402	Snap	No pointer to the recovery control block.
404	Snap	MSMCV block pointer is not valid.
405	Snap	MSMCV block ID is not valid.
FREEMAIN return code	Debug	Error returned from FREEMAIN macro when you free storage for mounted volume tables chain.
ARCMMCTL		
400	Snap	CKST_RC parameter is not valid.
401	Snap	CYCL_DATE is not passed.
ARCMMCTP		
400	Snap	Macro ID in ARCMMCP is not valid.
401	Snap	None of the requested functions were set in ARCMMCP.
ARCMMDS		
16	Debug	I/O error on input data set. Reason codes are: **00** Actual I/O error on input **10** Data set not found in SDSP **12** GENCB error **22** CRDLEN is not valid
123	Debug	End of volume abend because of CAPACITYMODE error.
124	Debug	End of volume CAPACITYMODE error on original.
125	Debug	End of volume CAPACITYMODE error on alternate.
ARCMMGDS		
4*nn*	Debug	Parameter list is not valid.
ARCMMLV1		
400	Debug	Single task SSM mode passed is not valid.
401	Debug	MTCB index is out of range.
402	Debug	No pointer to the recovery control block.
403	Debug	Subtask name is not valid.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
403	Debug	MSMCV block pointer is not valid.
405	Debug	MSMCV block ID is not valid.
FREEMAIN return code	Debug	Error returned from FREEMAIN macro when you free storage for mounted volume tables chain.
ARCMMQ		
400	Snap	MTCB index is not valid.
401	Snap	Post code is not valid.
ARCMMQA		
400	Snap	Macro ID in ARCMMQP is not valid.
401	Snap	None of the request functions were set in ARCMMQP.
402	Snap	The MTCB index passed is not valid.
403	Snap	The MMQ_POSTCODE is nonzero, but the MMQ_POST flag is off.
ATTACH return code	Snap	Error returned from ATTACH macro when a subtask attaching has failed.
ARCMMVF1		
405	Debug	Pointer to MDQE area is not valid.
4*nn*	Debug	Parameter list is not valid.
ARCMPDS		
15	Debug	More than one note list in member.
35	Debug	Error opening input data set.
58	Debug	Bad directory block.
17	Debug	I/O error reading input directory.
16	Debug	I/O error reading input data set.
123	Debug	End of volume abend because of CAPACITYMODE error.
124	Debug	End of volume CAPACITYMODE error on original.
125	Debug	End of volume CAPACITYMODE error on alternate.
ARCMPVF1		
409	Log	Incorrect combination of DS1RECAL and DS1IND08 flags in the Format 1 DSCB.
4*nn*	Log	Parameter list is not valid.
ARCMRCNB		
4	Snap	Unable to back out all reconnection control data set (CDS) changes.
8	Snap	Unable to back out TTOC updates for a reconnection failure.
4*nn*	Snap	Parameter list is not valid.
ARCMRECN		
nn	Snap	Error building the MVT for the MC2 volume. *nn* is the return code from ARCZMVT.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
nn	Snap	Error during deallocation. *nn* is the return code from ARCFREE.
400	Abend	Parameter list is not valid.
4*nn*	Snap	Parameter list is not valid.
ARCMRSLT		
4*nn*	Debug	Parameter list is not valid.
ARCMSCDS		
4*nn*	Debug	Parameter list is not valid.
ARCMSCLN		
400	Snap	MTCB block index is not valid.
401	Snap	MTCB block ID is not valid.
402	Snap	No pointer to the recovery control block.
403	Snap	RCBID information is not valid.
ARCMSDP		
1	Debug	VSAM request parameter list active on another processing unit.
8	Debug	Fails to build a request parameter list for reading SDSP.
12	Debug	VSAM physical error on reading SDSP.
400	Debug	ARCMVL1P macro ID is not valid.
401	Debug	MDQE area pointer is not valid.
402	Debug	MVOL TASKID is not valid.
4*nn*	Debug	Parameter list is not valid.
ARCMSDS		
16	Debug	A record with an non-valid length was read from the input (user's) data set.
ARCMSLV2		
400	Debug	Parameter list is not valid.
ARCMSDSP		
4*nn*	Debug	Parameter list is not valid.
ARCMSMCL		
4*nn*	Debug	Parameter list is not valid.
ARCMSMV		
4*nn*	Debug	Parameter list is not valid.
ARCMSPV		
12	Debug	Unexpected error.
ARCMSS		
400	Snap	MTCB block index is not valid.
401	Snap	MTCB block ID is not valid.
402	Snap	No pointer to the recovery control block.
410	Snap	Completion code in work-to-do ECB is not valid.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
411	Snap	MTCBFACT flag value in the MTCB block is not valid.
ARCMSSMP		
400	Snap	Macro ID in ARCMSSP is not valid.
401	Snap	None of the requested functions were set in ARCMSSP.
402	Snap	MTCB block index is not valid.
403	Snap	Both the wait for one specific MSS subtask and the wait for all MSS subtasks are specified.
404	Snap	The MSS_POSTCODE is nonzero, but the MSS_POST flag is off.
ATTACH return code	Debug	Error returned from ATTACH macro when a subtask attaching has failed.
ARCMS1EL		
400	Snap	Parameter list is not valid.
401	Snap	UCB pointer not provided.
402	Snap	Extract list entry error.
403	Snap	Format 1 DSCB area error.
404	Snap	Entry pointer is not valid.
405	Snap	Buffer pointer is not valid.
ARCMVCDS		
20	Debug	Error in closing or deallocating the new VTOC copy data set after another error encountered.
22	Debug	Error in closing or deallocating the old VTOC copy data set after another error was encountered.
4*nn*	Debug	Parameter list is not valid.
ARCMVDS		
nn	Debug	If *nnn* is greater than or equal to 100 and less than 200, subtract 100 from *nnn* and see message ARC12*nn*I for an explanation of the failing condition. Note that the leading zero in *nnn* is not used in the message identifier. If *nnn* is greater than or equal to 9*nn*, an abnormal end occurred when the return code was set to *nn*. See message ARC0003I for more information about the abnormal end. If the return code is greater than 0, see message ARC12*nn*I for the reason for the abnormal end.
4*nn*	Debug	Parameter list is not valid.
ARCMVBEG		
400	Debug	Parameter list ID that was passed is not valid.
401	Debug	Task ID that was passed is not valid.
ARCMVCLN		
400	Debug	Task ID is not valid.
401	Debug	Parameter list ID that was passed is not valid.
ARCMVOL		

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
nn	Debug	See the return codes for migration functions for ACR12*nn* error message table in *z/OS MVS System Messages, Vol 2 (ARC-ASA)*.
404	Debug	Bad ECB completion code.
411	Snap	A migration MVOL task is active even though its 'active' bit is off.
4*nn*	Debug	Parameter list is not valid.
ARCMVOLP		
4*nn*	Debug	Parameter list is not valid.
ARCMVTOC		
400	Debug	ARCMVL1P macro ID is not valid.
401	Debug	MDQE area pointer is not valid.
402	Debug	MVOL TASKID is not valid.
4*nn*	Debug	Parameter list is not valid.
ARCMXMGR		
1	Snap	Error in establishing an ESTAE routine.
2	Snap	Migration subtask could not be found.
3	Debug	Abend in ARCMXMGR.
4	Snap	Migration subtask could not be attached.
4*nn*	Debug	Parameter list is not valid.
ARCMXSB		
2	Debug	MXSB TCB post code invalid.
nn	Snap	Non-zero return code from storage release for the SGL, MIC, or SDATA control blocks.
ARCMXSS		
1	Snap	Error in establishing an ESTAE routine.
2	Snap	Error in establishing a timer routine.
3	Debug	Abend in ARCMXSS.
4	Debug	MXSS TCB post code invalid.
5	Debug	MXSB TCB post code invalid.
6	Debug	No ECB found posted.
7	Snap	Fields on exit found not zero.
ARCMXSSA		
1	Snap	Attach of MXSS TCB failed.
2	Snap	Error in establishing an ESTAE routine.
3	Debug	Abend in ARCMXSSA.
4*nn*	Debug	Parameter list is not valid.
ARCOSDP		
1	Abend	Bad request parameter list.
4	Debug	GENCB fails while attempting to create an ACB for open.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
8	Debug	OPEN macro fails for an SDSP data set.
12	Debug	GENCB fails while attempting to create a request parameter list to verify an SDSP data set.
16	Debug	Verify fails.
nn	Abend	Error when reading, writing, or erasing a small data set packing data set record. See *z/OS DFSMS Macro Instructions for Data Sets* for specific return code values.
ARCPBQ		
4	Snap	The ARCRPEXT installation-wide exit routine improperly modified the MWE control block, which was passed as input to the exit.
400	Debug	Input MWE pointer is not valid.
ARCPDO		
12	Snap	A record that is not valid was found. The block of data containing the record was not written to the ARCPDOX data set.
ARCPMDQE		
4*nn*	Debug	Parameter list is not valid.
ARCPMWE		
10	Snap	FREEMAIN failure.
nn	Debug	Error during FREEMAIN.
400	Debug	Parameter list is not valid.
ARCPROPN		
3	Debug	Data set open fails.
7	Snap	The DCB is closed and is not the DCB for either the migration, backup, or dump activity logs.
8	Debug	Error allocating non-sysout data set.
12	Debug	Unexpected error.
ARCZEST return code	Debug	Either ESTAI or ESTAE is not established.
ARCPRQ		
4	Snap	The ARCRPEXT installation-wide exit routine improperly modified the MWE control block, which was passed as input to the exit.
ARCQSORT		
4*nn*	Debug	Parameter list is not valid.
ARCRACF		
4*nn*	Debug	Parameter list is not valid.
ARCRALOT		
6	Debug	Error allocating the data set.
20	Debug	SAMETRK specified but target and original track sizes are different.
22	Debug	RELTRK specified but target track size is smaller than the original track size.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
26	Debug	No DAOPTION specified but the target track size is smaller than the original track size.
28	Debug	Unknown DAOPTION and track length combination.
420	Debug	Parameter address is null or parameter list ID is not set properly.
421	Debug	The pointer for one of the following is null: • The input data set name • The MVT • The RCB • The CDD
ARCRCTL		
nn	Fatal	ARCGMAIN called to get storage for the recall task control blocks. ARCGMAIN returns a nonzero return code (*nn*).
ARCRCYDS		
n	Debug	ERRORALTERNATE(MARKFULL) option is in effect and an error occured. *n* represents the kind of data movement error.
nn	Snap	ERRORALTERNATE(MARKFULL) option is in effect and a SYNCDEV error occured. *nn* represents the ECB completion code.
99	Snap	MVTFATDS is on.
4*nn*	Snap	Parameter error.
ARCRCYPQ		
5	Abend	New FBID is less than current (backup).
7	Abend	New FBID is less than current (migration).
400	Abend	YTCB pointer is not valid.
401	Abend	YDSP pointer is not valid.
ARCRCYV		
7	Snap	JFCB read fails.
8	Snap	JFCB and DCB errors.
19	Snap	Could not find TIOT for this ddname.
ARCRDIN		
1	Abend	IKJSCAN error.
nn	Debug	No buffer for messages to be returned by CPS return code, which is returned by ARCGMAIN.
ARCRDS		
4	Debug	A UCB for the target SMS volume not found.
ARCRDSS		
69	Debug	Abnormal end in DFSMSdss.
76	Snap	Failure attempting to link the DFSMSdss load module.
4*nn*	Debug	Parameter list is not valid.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
ARCRELMV		
4*nn*	Debug	Parameter list is not valid.
ARCREUIM		
4	Abend	A UCB pointer could not be found for the volume serial number DFSMSdss passed to the new-volume notification exit.
4*nn*	Debug	Parameter error.
EIOPTION	Debug	(From module ARCRUIM1) Called each time that ARCREUIM is entered when this module has been patched to request debug messages.
EIOPTION	Abend	(From module ARCRUIM2) ARCREUIM returns a option to DFSMSdss that is not valid.
EIRETCOD	Debug	(From module ARCRUIM4) Called each time that ARCREUIM is exited when this module has been patched to request debug messages.
ARCRGUIM		
400	Abend	Pointer to the EIDB not specified or the control block identifier is not valid (ADREIBID).
401	Abend	Pointer to the ARCRDSSP could not be determined because EIUSEPTR was 0.
402	Abend	The ARCRDSSP control block identifier is not valid (RDSS_NAME).
4*nn* greater than 402	Debug	Parameter list is not valid.
EIOPTION	Abend	DFSMSdss did not like how the option was processed. The entry point is used to indicate this error.
ARCRMGDS		
400	Debug	MDQE pointer is not valid.
401	Debug	MVT-entry pointer is not valid.
ARCRNVDS		
12	Debug	Failed to locate the existing data set.
16	Debug	Failed to scratch the existing data set.
18	Debug	Error processing DAOPTION.
41	Debug	Failure to recreate the MIGRAT entry when DFDSS restore fails.
70	Debug	Failure performing SMS related function.
4*nn*	Debug	Parameter list is not valid.
ARCRNVRC		
4*nn*	Debug	Parameter list is not valid.
ARCROPIN		
nn	Debug	See the return codes for RECALL and RECOVERY functions for ARC11*nn* error message table in *z/OS MVS System Messages, Vol 2 (ARC-ASA)*.
4*nn*	Debug	Parameter is not valid.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
ARCRPDS		
11	Snap	Unexpected end-of-file reached on migration copy.
12	Snap	I/O error in reading migration copy, or CDD record length is not valid.
13	Snap	I/O error in writing restored copy.
52	Snap	FREEMAIN failure.
123	Debug	End of volume CAPACITYMODE error.
ARCRPF1		
401	Debug	Target MVT address or target MVT is not valid.
402	Debug	The address of the RTCB, the MWE or the MCD is missing from the RDMP.
ARCRRBLK		
4*nn*	Debug	Parameter list is not valid.
ARCRSDS		
12	Debug	I/O error reading migration copy.
13	Debug	I/O error writing restored copy.
99	Debug	Error during recall or recovery.
ARCRSTAT		
400	Debug	Entry point called with a volume type that is not valid.
ARCRSTR		
nn	Debug	If *nnn* is greater than or equal to 9*nn*, an abnormal end occurred when the return code was set to *nn*. See message ARC0003I for more information about the abnormal end. If the return code is greater than 0, see message ARC11*nn*I for the reason for the error.
nn	Fatal	GETMAIN failed.
ARCRSTR1		
nn	Snap	Errors while reading JFCB. For more detail about this RDJFCB return code see *z/OS DFSMS Using Data Sets*.
ARCRVDS		
4	Debug	Cannot find UCB.
12	Debug	Failed to delete the existing data set.
16	Debug	DFSMSdss restored data set but allocation of the data set fails.
20	Debug	Failure to locate existing data set.
400	Debug	Parameter list is not valid.
ARCSACLN		
400	Debug	Parameter list ID is not valid.
ARCSADVL		
4*nn*	Debug	Parameter list is not valid.
ARCSAINT		

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
400	Debug	Parameter list is not valid.
ARCSALSZ		
4*nn*	Debug	Parameter list is not valid.
ARCSBELG		
4*nn*	Debug	Parameter list is not valid.
ARCSBVOL		
400	Debug	BTCB index is not valid.
ARCSELAV		
16	Debug	STIMER macro failure.
4*nn*	Debug	Parameter list is not valid.
ARCSELDV		
4*nn*	Debug	Parameter list is not valid.
ARCSELMV		
4*nn*	Debug	Parameter list is not valid.
ARCSELTV		
1	Debug	This problem occurs during initial-volume processing if the volume serial number is not marked with blanks, nulls, or SCRATCH. It also occurs during end-of-volume processing if the volume serial number on the tape does not match the entry in the MVTVSN field. The tape volume is marked as full to prevent DFSMShsm from using it.
2	Debug	Internal error, no TTX found.
ARCSLVOL		
18	Snap	Error processing DAOPTION; device type for last L0 volume is unknown. Reason code=30.
ARCSMBMQ		
400	Debug	MTCB index is not valid.
401	Debug	MTCB index does not refer to a valid MTCB.
ARCSMELG		
4*nn*	Debug	Parameter list is not valid.
ARCSMEV		
409	Log	Incorrect combination of DS1RECAL and DS1IND08 flags in the Format 1 DSCB.
4*nn*	Debug	Parameter list is not valid.
ARCSMINT		
400	Debug	Parameter list is not valid.
ARCSMPED		
4*nn*	Debug	Parameter list is not valid.
ARCSMPMQ		
400	Debug	MTCB index is not valid (too high or too low).

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
401	Debug	MTCB index does not refer to a valid MTCB.
ARCSMSAD		
4*nn*	Debug	Parameter list is not valid.
ARCSMSVM		
4*nn*	Debug	Parameter list is not valid.
ARCSMVF1		
4*nn*	Debug	Parameter list is not valid.
ARCSMVOL		
400	Debug	MTCB index is not valid.
401	Debug	MTCB index does not refer to a valid MTCB.
ARCSM570		
4*nn*	Debug	Parameter list is not valid.
ARCSSDP		
400	Snap	MVT pointer is not valid.
ARCSVCUT		
4*nn*	Debug	Parameter list is not valid.
ARCTACS		
400	Debug	Parameter list is not valid.
401	Debug	No data set name.
ARCTCLOS		
4	Debug	Nonzero return code in register 15 after processing of the FREEPOOL macro.
8	Debug	Nonzero return code in register 15 after processing of the CLOSE macro.
12	Debug	Close abnormal end (X'14') in processing the CLOSE macro.
16	Debug	Indeterminate abnormal end in processing the CLOSE macro.
24	Debug	Error in reading or updating an MCV record for a migration volume or a backup volume.
67	Debug	Nonzero return code is returned from ARCZNOTE to obtain a block identifier of the next block to be written.
69	Debug	Nonzero return code is returned from ARCZSYNC to flush the tape buffer and write data to tape.
80	Debug	Nonzero return code was returned from a call to ARCZMESN.
4*nn*	Debug	Parameter list is not valid.
900	Debug	Error from ARCZRBL.
ARCTCOM		
100	Abend	Abnormal error during parse check returned from ARCPARSE.
ARCTECDM		

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
400	Debug	Parameter list is not valid.
ARCTEOV		
4	Debug	Error in creating MCDS MCV record.
8	Debug	Error in deleting MCDS MCV record.
12	Debug	Error in reading MCDS MCV record.
16	Debug	Error in writing MCDS MCV record.
20	Debug	Error in creating BCDS MCT record.
24	Debug	Error in deleting BCDS MCT record.
28	Debug	Error in reading BCDS MCT record.
32	Debug	Error in writing BCDS MCT record.
36	Debug	Error in creating OCDS TTOC record.
40	Debug	Error in reading OCDS TTOC record.
44	Debug	Error in writing OCDS TTOC record.
48	Debug	Error in changing key of OCDS TTOC record.
52	Debug	Error in deleting BCDS BVR volume entry.
56	Debug	Error in creating BCDS BVR volume entry.
60	Debug	Error in updating BCDS BVR volume entry.
64	Debug	The tape volume contains valid DFSMShsm data, but the file sequence number in the MCV or MVT record is nonzero.
68	Debug	The tape volume is RACF-protected, but does not appear in DFSMShsm's RACF tape volume set. Also, the tape volume is not in DFSMShsm's inventory of backup or migration volumes.
72	Debug	The tape volume is RACF-protected, but does not appear in DFSMShsm's RACF tape volume set. Also, the tape volume is in DFSMShsm's inventory of backup or migration volumes. The MCT or MCV records indicate that the tape volume is empty (the file sequence number is zero). The tape volume is being removed from DFSMShsm's inventory of backup or migration volumes.
80	Debug	Error in attempting to add the tape volume to DFSMShsm's RACF tape volume set.
84	Debug	Error in attempting to delete tape volume from DFSMShsm's RACF tape volume set.
nn	Debug	Internal ADDVOL process of the tape volume was not successful.
ARCTFULL		
401	Log	Incorrect parameter list identifier.
402	Log	Incorrect volume type is passed.
403	Log	Incorrect volume list pointer is passed.
404	Log	Incorrect HARD LOG type is passed.
ARCTLCC		

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
4*nn*	Debug	Parameter list is not valid.
ARCTLEV1		
400	Debug	Input parameter list passed to this module is not valid.
ARCTLR		
4*nn*	Debug	Parameter list is not valid.
ARCTLU		
400	Debug	Parameter list is not valid.
401	Debug	Volume is not valid.
ARCTMIG		
100	Abend	Abnormal parse error.
ARCTMT		
401	Debug	Zero UCB pointer.
ARCTOPEN		
1	Debug	Length mismatch for duplex tape.
2	Debug	Compaction mismatch for duplex tape.
35	Debug	Error in opening input data set.
36	Debug	Error in opening output data set.
54	Debug	Installation exit abend.
60	Debug	Failure in establishing ESTAE environment.
61	Debug	Failure in internal ADDVOL.
67	Debug	Error in NOTE macro.
68	Debug	Error in POINT macro.
111	Debug	Error in writing MCT or MCV record.
123	Debug	OPEN abend because of CAPACITYMODE error.
124	Debug	OPEN CAPACITYMODE error on input.
125	Debug	OPEN CAPACITYMODE error on output.
ARCTPMSG		
4*nn*	Debug	Parameter list is not valid.
ARCTRES		
220	Abend	Abnormal parse error.
ARCTSPAN		
123	Debug	End of volume abend because of CAPACITYMODE error.
124	Debug	End of volume CAPACITYMODE error on original.
125	Debug	End of volume CAPACITYMODE error on alternate.
201	Debug	Failure in reading TTOC record.
4*nn*	Debug	Parameter list is not valid.
ARCTTCDS		
4*nn*	Debug	Parameter list is not valid.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
ARCTTCIM		
4*nn*	Debug	Parameter list is not valid.
ARCTTCIU		
4*nn*	Debug	Parameter list is not valid.
ARCTTOC		
4*nn*	Debug	Parameter list is not valid.
ARCTTOCA		
4*nn*	Debug	Parameter list is not valid.
ARCTTOCM		
4*nn*	Debug	Parameter list is not valid.
ARCTTOCP		
4*nn*	Debug	Parameter list is not valid.
ARCTVINT		
4*nn*	Debug	Parameter list is not valid.
ARCTVSC		
24	Debug	The selected tape volume is unacceptable because it is RACF-protected but does not appear in the RACF tape volume set. The DFSMShsm control data set volume record (MCV, MCT, or DVL) indicates the tape volume contains valid DFSMShsm data. The tape volume is being marked full to prevent DFSMShsm from using it.
48	Debug	The selected tape volume is unacceptable because it is RACF-protected but does not appear in the RACF tape volume set. The DFSMShsm control data set volume record (MCV, MCT, or DVL) indicates the tape volume is empty. The tape volume is being deleted to prevent DFSMShsm from using it.
ARCTXLCS		
4*nn*	Debug	Parameter list is not valid.
ARCUBVR		
400	Debug	Contradictory CAPACITYMODE update request.
ARCUCTL1		
nn	Log	Return code from ARCZREAD; read error.
ARCUCTL2		
nn	Log	Return code from ARCZUPDT; update error.
ARCUCTL3		
nn	Log	Return code from ARCZREAD; read error.
ARCUCTL4		
nn	Debug	Return code from ARCZUPDT; update error.
ARCUCTL5		
400	Debug	Entry point was called with a volume type that is not valid.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
ARCULOCK		
16	Snap	The transition lock could not be obtained during the ARCCAT release function. ARCCAT release is disabled on the DFSMShsm host that issued this error until DFSMShsm is restarted.
999	Debug	Task reserving control data sets is not performing reserves in the correct order. DFSMShsm must serialize control data sets in the following order: MCDS, BCDS, OCDS.
OBTAIN macro return code	Log	Volume VTOC entry obtain fails.
VERIFY macro return code	Fatal	VSAM verify fails on the migration control data set.
VERIFY macro return code	Snap	VSAM verify fails on the backup control data set.
VERIFY macro return code	Snap	VSAM verify fails on the off-line control data set.
ARCULVOL		
400	Debug	DDNAME not passed.
ARCUPCCL		
4	Debug	CHPID value went negative.
4*nn*	Debug	Parameter list is not valid.
ARCUSDS		
12	Snap	I/O error in reading migrated data set.
13	Snap	I/O error in writing restored copy.
ARCUSTA2		
nn	Log	Return code from ARCZUPDT; update error.
ARCUSTA3		
nn greater than 4	Log	Return code from ARCZREAD; read error.
ARCUSTA4		
nn	Log	Return code from ARCZUPD; update error.
ARCUSTA5		
nn greater than 8	Log	Return code from ARCZREAD; read error.
ARCUSTA6		
nn	Log	Return code from ARCZUPDT; update error.
ARCUSTA7		
nn greater than 4	Log	Return code from ARCZWRIT; write error.
ARCUTTOC		
4*nn*	Debug	Parameter list is not valid.
ARCVCUPD		
6	Debug	No work area supplied for read request.
ARCVSCHK		
4*nn*	Debug	Parameter is not valid.
ARCVSTA4		
nn	Log	Return code from ARCZUPD; update error.
ARCVVSC		

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
405	Debug	Size fields calculated are inconsistent.
4*nn*	Debug	Parameter list is not valid.
ARCWTO		
15	Snap	Length of log record is not valid.
16	Snap	TPUT parameter list error.
184	Debug	I/O error.
ARCYLIST		
400	Abend	YQE element is not valid.
ARCYSCAN		
nn	Debug	Connected set is not supported for RECYCLE.
ARCYTDS		
123	Debug	End of volume CAPACITYMODE error on input.
124	Debug	End of volume CAPACITYMODE error on output for original.
125	Debug	End of volume CAPACITYMODE error on output for alternate.
400	Snap	Input data set is single file format and a 0 block count is passed from caller.
ARCYVOLA		
99	Snap	MVTFATDS is on; the device control block has been allocated.
ARCZACER		
4*nn*	Debug	Parameter list is not valid.
ARCZALAT		
8	Debug	Alternate tape not allocated.
16	Debug	GETMAIN failed for ALT MVT.
20	Debug	Failure checking STGCLASS.
4*nn*	Debug	Parameter list is not valid.
ARCZAL2V		
4*nn*	Debug	Parameter list is not valid.
ARCZAPF		
4*nn*	Debug	Parameter list is not valid.
ARCZBDST		
8	Debug	Greater than 5 I/O errors.
10	Debug	DVL record positioning fails.
52	Debug	Failed to obtain storage for dump volume table.
ARCZCAT		
400	Debug	Parameter list is not valid.
ARCZCDQR		
4*nn*	Debug	Parameter list is not valid.
ARCZCELL		

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
8	Snap	A double FREECELL was attempted.
ARCZCLNI		
400	Debug	Parameter list is not valid.
ARCZCKES		
4*nn*	Debug	Parameter list is not valid.
ARCZCKST		
4*nn*	Debug	Parameter list is not valid.
ARCZCOMP		
400	Debug	Error in parameter list. Macro ID or DSNAME pointer is not valid.
ARCZCP31		
4*nn*	Debug	Parameter list is not valid.
ARCZCSDP		
4*nn*	Debug	Parameter list is not valid.
ARCZCSFT		
4*nn*	Debug	Parameter list is not valid.
ARCZCSI		
4*nn*	Debug	Parameter list is not valid.
ARCZCYCL		
4*nn*	Debug	Macro ID that was passed is not valid.
ARCZDAVL		
4*nn*	Debug	Parameter list is not valid.
ARCZDDSC		
4*nn*	Debug	Parameter list is not valid.
ARCZDEL		
1	Abend	VSAM return code indicates bad request parameter list.
4	Debug	Return code indicates record not found.
nn	Debug	Value of error code in request parameter list returned by VSAM, if DELRC indicated either physical (return code is 16) or logical (return code is 20) error.
300	Snap	A timeout occurred waiting for the transition lock. ARCCAT release is disabled on the DFSMShsm host that issued this error until DFSMShsm is restarted.
4*nn*	Debug	Parameter list not initialized properly.
ARCZDPUT		
8	Snap	Save area is corrupted.
ARCZDSF		
60	Debug	ARCZEST ESTAE failure.
4*nn*	Debug	Parameter list not valid.
ARCZDSNQ		
4*nn*	Debug	Parameter list is not valid.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
ARCZDUCUC		
4*nn*	Debug	Parameter list is not valid.
ARCZEND		
400	Debug	Function specified is not valid.
401	Debug	EOD, EOV, or both not specified.
ARCZESDP		
4*nn*	Debug	Parameter list is not valid.
ARCZEST		
60	Debug	ESTAE setup or cancel failed.
400	Debug	Parameter list is not valid.
ARCZFENT		
4*nn*	Debug	Parameter list is not valid.
ARCZGCAT		
400	Debug	Macro ID is not valid.
408	Debug	Pointer to activity log was not passed.
ARCZGENM		
400	Debug	Invocation is not valid.
ARCZJFCB		
4*nn*	Debug	Parameter list is not valid.
ARCZLOC		
7	Snap	An error return code given for ARCZVRES; for example, conversion failed.
400	Debug	Parameter list is not valid.
ARCZLSRT		
66	Snap	The sort was not started. The linked list appears to be broken. A zero forward pointer was encountered for an element that is not the tail element.
4*nn*	Debug	Parameter list is not valid.
ARCZMASK		
400	Abend	MVT pointer not passed.
ARCZMDST		
4*nn*	Debug	Parameter list is not valid.
ARCZMDXA		
4*nn*	Debug	Parameter list is not valid.
ARCZMESN		
400	Debug	Macro ID is not valid.
401	Debug	DCB pointer is not valid.
ARCZMGCR		
4*nn*	Snap	Parameter list not valid.
ARCZMSGS		
52	Debug	No storage available for chain or element.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
nn	Debug	Error in FREEMAIN (return code from ARCFMAIN).
4*nn*	Debug	Parameter list is not valid.
ARCZMVT		
4*nn*	Debug	Parameter list is not valid.
ARCZNOTE		
4*nn*	Debug	Parameter list is not valid.
ARCZOBT		
400	Debug	Parameter list is not valid.
ARCZOSDP		
4*nn*	Debug	Parameter list is not valid.
ARCZOUCB		
4*nn*	Debug	Parameter list is not valid.
ARCZPART		
4*nn*	Debug	Parameter list is not valid.
ARCZPONT		
4*nn*	Debug	Parameter list is not valid.
ARCZPORD		
4*nn*	Debug	Parameter list is not valid.
ARCZPOS		
1	Abend	Bad request parameter list.
20	Debug	Undetermined logic error.
22	Debug	No OCDS is defined.
24	Debug	No BCDS is defined.
26	Debug	SMS VSAM server error.
109	Debug	VSAM error code indicating the first attempt to access the data set has failed due to a broken index (non-RLS environment).
110	Debug	Error code passed back from VSAM on subsequent attempts to access data set with a corrupt index (non-RLS environment).
157	Debug	VSAM error code indicating the first attempt to access the data set has failed due to a broken index (RLS environment).
158	Debug	Error code passed back from VSAM on subsequent attempts to access data set with a corrupt index (RLS environment).
400	Debug	Parameter list is not valid.
410	Debug	Record type is not valid.
ARCZQBLD		
4*nn*	Debug	Parameter list is not valid.
ARCZQMSG		
8	Debug	Error in building message chain, ending process. No error message necessary.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
4*nn*	Debug	Parameter list is not valid.
ARCZQUE		
66	Snap	A possible broken queue is encountered. See associated reason codes.
4*nn*	Debug	Parameter list is not valid.
ARCZQVOL		
400	Debug	Data set name is not valid.
401	Debug	Volume serial number is not valid.
ARCZRACF		
4*nn*	Snap	Parameter list is not valid.
ARCZRBL		
4*nn*	Debug	Parameter list is not valid.
ARCZREAD		
1	Abend	VSAM request parameter list is not valid.
6	Debug	No buffer provided for read.
22	Debug	OCDS referenced but not defined.
24	Debug	BCDS referenced but not defined.
26	Debug	RLS server error.
30	Debug	MCBR record error encountered.
44	Debug	Work area not large enough.
109	Debug	VSAM error code indicating the first attempt to access the data set has failed due to a broken index (non-RLS environment).
110	Debug	Error code passed back from VSAM on subsequent attempts to access data set with a corrupt index (non-RLS environment).
157	Debug	VSAM error code indicating the first attempt to access the data set has failed due to a broken index (RLS environment).
158	Debug	Error code passed back from VSAM on subsequent attempts to access data set with a corrupt index (RLS environment).
400	Debug	Parameter error is not valid.
410	Debug	Key type is not valid.
ARCZRLSE		
nn	Debug	VSAM ENDREQ error; *nn* is the request parameter list feedback byte (RPLERRCD).
24	Snap	Non-CDS release error.
400	Debug	Parameter list is not valid.
ARCZRNXT		
1	Abend	VSAM request parameter list is not valid.
20	Debug	No VSAM request parameter list passed.
22	Debug	No OCDS defined.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
24	Debug	No BCDS defined.
26	Debug	SMSVSAM server error.
30	Debug	MCBR record error encountered.
44	Debug	Work area not large enough from VSAM.
109	Debug	VSAM error code indicating the first attempt to access the data set has failed due to a broken index (non-RLS environment).
110	Debug	Error code passed back from VSAM on subsequent attempts to access data set with a corrupt index (non-RLS environment).
157	Debug	VSAM error code indicating the first attempt to access the data set has failed due to a broken index (RLS environment).
158	Debug	Error code passed back from VSAM on subsequent attempts to access data set with a corrupt index (RLS environment).
400	Debug	Parameter list is not valid.
ARCZRPLS		
4*nn*	Debug	Parameter list is not valid.
ARCZSACS		
4*nn*	Debug	Parameter list is not valid.
ARCZSALT		
4*nn*	Debug	Parameter list is not valid.
ARCZSAQM		
4	Debug	Remove requested, but a corresponding entry not found.
8	Debug	The SAQ is full, and this task does not yet have an entry in the SAQ.
4*nn*	Debug	Parameter list is not valid.
ARCZSCAN		
4*nn*	Debug	Parameter list is not valid.
ARCZSCLN		
400	Debug	Invocation is not valid.
ARCZSCMC		
400	Debug	Parameter list ID is not valid.
401	Debug	Return DEFN area pointer is not valid.
ARCZSCON		
4*nn*	Debug	Parameter list is not valid.
ARCZSCPD		
4*nn*	Debug	Parameter list is not valid.
ARCZSCR		
4*nn*	Debug	Parameter list is not valid.
ARCZSDEF		

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
4*nn*	Debug	Parameter list is not valid.
ARCZSDEL		
4*nn*	Debug	Parameter list is not valid.
ARCZSDMC		
400	Debug	Parameter list ID is not valid.
401	Debug	Pointer to default MC is not valid.
ARCZSEG		
400	Abend	UCB pointer not passed.
ARCZSERL		
300	Debug	TCB already exists in CATTABLE.
301	Debug	TCB was not found in CATTABLE.
400	Debug	Identifier level is not valid.
401	Debug	SCOPE for ENQ is not valid.
402	Debug	SCOPE for DEQ is not valid.
403	Debug	RET for ENQ is not valid.
404	Debug	RET for DEQ is not valid.
405	Debug	Request is not valid.
406	Debug	Type is not valid (SHR/EXCL).
407	Debug	Directed ENQ is not valid.
408	Debug	RET for reserve is not valid.
409	Debug	SCOPE for reserve is not valid.
410	Debug	Directed DEQ is not valid.
411	Debug	ARCBTAPE requested.
ARCZSFVV		
400	Debug	Parameter list ID is not valid.
ARCZSLKT		
4*nn*	Debug	Parameter list is not valid.
ARCZSLOC		
4*nn*	Debug	Parameter list is not valid.
ARCZSMCD		
4*nn*	Debug	Parameter list is not valid.
ARCZSMPM		
4*nn*	Snap	Parameter list is not valid.
ARCZSMSG		
400	Debug	SSOB extension pointer is not valid.
ARCZSMVT		
400	Debug	Macro ID in ARCMVTP is not valid.
401	Debug	Volume type specified is not valid.
ARCZSSCD		
4*nn*	Debug	Parameter list is not valid.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
ARCZSSGD		
400	Debug	Parameter list ID is not valid.
401	Debug	Storage group name pointer is not valid.
ARCZSSI		
8	Debug	Subfunction unable to complete request.
12	Debug	SSOB extension format is not valid.
16	Debug	Indeterminate error.
4*nn*	Debug	Parameter list is not valid.
ARCZSTC		
4*nn*	Debug	Parameter list is not valid.
ARCZSVCC		
4*nn*	Debug	Incorrect parameter list.
ARCZSVLD		
8	Debug	Volume is not SMS managed.
4*nn*	Debug	Parameter list is not valide.
ARCZSYNC		
4*nn*	Debug	Parameter list is not valid.
ARCZS1EL		
4*nn*	Debug	Parameter list is not valid.
ARCZTFVV		
1	Debug	ARCZMVT encountered an error trying to find or build an MVT for the source volume.
2	Debug	ARCALVOL encountered an error trying to allocate the source volume.
3	Debug	ARCZVTOC encountered an error trying to access the VTOC.
8	Debug	Error freeing storage.
10	Debug	ARCVVSC encountered an error trying to get storage for an SDATA control block.
20	Debug	ARCZTFVV encountered an error trying to get storage for an extract list for a VTOC entry list.
4*nn*	Debug	Parameter list is not valid.
ARCZTMVT		
4*nn*	Debug	Parameter list is not valid.
ARCZTODC		
400	Debug	Parameter list ID is not valid.
401	Debug	Time of day clock value is not valid.
ARCZUDCR		
2	Debug	DCR read error.
4	Debug	DCR write error.
400	Debug	Parameter list is not valid.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
ARCZUF1		
400	Snap	Parameter list not valid.
ARCZULST		
4*nn*	Debug	Parameter list is not valid.
ARCZUPDT		
1	Abend	VSAM request parameter list is not valid.
20	Debug	Undetermined logic error to caller.
26	Fatal	VSAM request parameter list is not valid.
300	Snap	A timeout occurred waiting for the transition lock. ARCCAT release is disabled on the DFSMShsm host that issued this error until DFSMShsm is restarted.
400	Debug	Parameter list is not valid.
410	Debug	Key type is not valid.
VSAM return code	Debug	VSAM update failure.
ARCZUPIN		
4*nn*	Debug	Parameter list is not valid
ARCZVALB		
16	Fatal	I/O error in CDS.
nn	Fatal	Return code from ARCZVSAM.
400	Debug	Parameter list errors.
ARCZVCNM		
4*nn*	Debug	Parameter list is not valid.
ARCZVCRW		
4*nn*	Debug	Parameter list is not valid.
ARCZVCUT		
8	Debug	Module processing may have completed successfully. See reason codes (reason code 4 is a close error, reason code 6 is a deallocation error).
12	Debug	Module processing does not complete. See reason codes.
4*nn*	Debug	Parameter list is not valid.
ARCZVDSC		
4	Debug	SMS not active.
4*nn*	Debug	Parameter list is not valid.
ARCZVDSD		
400	Debug	Pointer to DADSM parameter list is not valid.
ARCZVLAT		
400	Debug	Macro ID is not valid, MVT pointer or MCV pointer not passed, or pointer to MCV record read area not passed, or flag settings are not valid (any parameter list error).
ARCZVSAM		

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
26	Fatal	RLS server error; access to CDSs is lost.
ARCZVSM1		
401	Fatal	Request parameter list pointer is not valid.
402	Fatal	ACB pointer is not valid.
403	Fatal	Function is not valid.
ARCZVSM2		
nn	Fatal	No pointer to CDSB found.
ARCZVSM3		
nn	Fatal	Verify error.
ARCZVSM4		
nn	Fatal	VSAM end request error.
ARCZVSM5		
nn	Fatal	VSAM end request error.
ARCZVTOC		
60	Debug	Unable to establish ESTAE environment.
4*nn*	Debug	Parameter is not valid.
501	Debug	I/O error in reading VTOC.
554	Debug	Open of VTOC fails.
555	Debug	Close of VTOC fails.
557	Debug	Unable to read JFCB.
807	Debug	GETBUF or TRKCALC macro fails.
999	Debug	Error reported to ARCGMAIN or other service module.
ARCZWRIT		
1	Abend	VSAM request parameter list is not valid.
109	Debug	VSAM error code indicating the first attempt to access the data set has failed due to a broken index (non-RLS environment).
110	Debug	Error code passed back from VSAM on subsequent attempts to access data set with a corrupt index (non-RLS environment).
157	Debug	VSAM error code indicating the first attempt to access the data set has failed due to a broken index (RLS environment).
158	Debug	Error code passed back from VSAM on subsequent attempts to access data set with a corrupt index (RLS environment).
300	Snap	A timeout occurred waiting for the transition lock. ARCCAT release is disabled on the DFSMShsm host that issued this error until DFSMShsm is restarted.
400	Debug	Parameter list is not valid.
410	Debug	Key or type is not valid.
VSAM return code	Debug	VSAM write fails.

Table 3. Entries That Pass Error Codes to ARCERP (continued)

Error Code	Process Level	Description of Problem
ARCZW2NW		
400	Debug	Parameter list is not valid.

Abnormal end codes

DFSMShsm generates user abnormal ends in two cases:

- 999 (X'3E7'): Caused when the DFSMShsm STOP command is issued with the DUMP parameter.
- 1000 (X'3E8') + return code: Caused when the DFSMShsm internal return code is flagged for special processing that includes generating an abnormal end dump. See “Return code processing” on page 37 for further explanation.

System abnormal ends can also occur, but all modules in the DFSMShsm address space are ESTAI-protected. Control can be given to the task-related ESTAI module from any DFSMShsm module in the event of a system abnormal end. The task-related ESTAI module performs cleanup functions and then allows a subtask abnormal end process to continue. The main DFSMShsm task establishes subtasks that abnormally ended again.

Chapter 9. Using patches for problem determination

This topic provides information about using DFSMShsm-supported patches for problem determination.

For instances where it is necessary to determine problem causes, such as when an installation exit abnormally ends, the problem determination patches might offer the needed assistance. These DFSMShsm supported patches remain supported from release to release without modification.

Before applying these patches to your system, understand the following:

- Some of the PATCH commands listed in this topic include the VERIFY parameter or comments about the patch. The VERIFY parameter and comments are optional. However, when you patch full bytes of data from a terminal, it is recommended that you use the VERIFY parameter to help catch any errors in the command entry. To see the current value of a byte before changing it, use the DISPLAY command.
- If you are using the PATCH command to change only part of a byte, you should use the BITS parameter.
- If you need to see the output data from a PATCH command online, before DFSMShsm is shut down, you can specify the OUTDATASET parameter of the PATCH command.

For more information on using the PATCH and DISPLAY commands, see Chapter 10, "Using DFSMShsm maintenance commands," on page 95.

Problem determination patches

The problem determination patches are designed to aid users in diagnosing problems. Some of the patches can appropriately be added as startup members, while others are appropriate only for temporary usage.

The following table shows brief descriptions and the associated procedures for all the DFSMShsm-supported problem determination patches.

Problem determination patch descriptions	Associated procedure (See . . .)
Causing a dump to be generated if an installation exit abnormally ends	"Causing a dump to be generated if an installation exit abends" on page 92
Tracing the OPEN/CLOSE/END OF VOLUME for DFSMShsm tape and DASD	"Steps for tracing the OPEN/CLOSE/END OF VOLUME for DFSMShsm tape and DASD" on page 92
Causing DFSMShsm and DFSMSdss dumps when DFSMSdss is the data mover and a selected DFSMSdss error occurs	"Causing DFSMShsm and DFSMSdss dumps when DFSMSdss is the data mover and a selected DFSMSdss error occurs" on page 92
Determining why SMS-managed data sets are not processed.	"Determining why SMS-managed data sets are not processed" on page 93
Increasing the amount of PDA tracing that is performed	"Increasing the amount of PDA tracing performed" on page 93
Analyzing the CELL POOL free chain	"Analyzing the CELL POOL free chain" on page 94

Causing a dump to be generated if an installation exit abends

If an installation exit abnormally ends, you can force DFSMShsm to generate a dump. Setting the MCVTFDMP flag to on prevents DFSMShsm from continuing its normal ESTAE recovery process and instead causes a dump. To set the MCVTFDMP flag to on, enter the PATCH command as follows:

```
PATCH .MCVT.+2D BITS(.......1) /* dump after installation exit */
                               /* abnormally ends                */
```

Note: When a dump is taken due to an installation exit abnormal end, this flag is set to off.

Steps for tracing the OPEN/CLOSE/END OF VOLUME for DFSMShsm tape and DASD

Generalized trace facility (GTF) trace data for open/close/end-of-volume (O/C/EOV) can be collected during DFSMShsm processing for DASD and tape media.

Perform the following steps to collect GTF data during O/C/EOV processing for DFSMShsm.

1. Start the GTF with MODE=EXTERNAL, TRACE=USR.
2. To collect DFSMShsm tape-oriented O/C/EOV records, perform the following DFSMShsm PATCH command:

   ```
   PATCH .MCVT.+F2 X'00' /* allow tracing of tape O/C/EOV */
   ```

3. To collect DFSMShsm DASD-oriented O/C/EOV records, perform the following DFSMShsm PATCH command:

   ```
   PATCH .MCVT.+F3 X'00' /* allow tracing of DASD O/C/EOV */
   ```

4. Initiate the DFSMShsm function that you want to trace.
5. When the DFSMShsm tracing is complete, apply the following DFSMShsm PATCH command:

   ```
   PATCH .MCVT.+F2 X'FFFF' /* ensure the traces are turned off */
   ```

6. Stop the GTF and review the trace output.

Causing DFSMShsm and DFSMSdss dumps when DFSMSdss is the data mover and a selected DFSMSdss error occurs

This PATCH command allows you to create a dump of both DFSMShsm and DFSMSdss when DFSMShsm is using DFSMSdss as the data mover. This capability helps you isolate the source of a problem when multiple products are involved.

The occurrence of a selected DFSMSdss error message triggers dumps of both DFSMShsm and DFSMSdss. The SYS1DUMP/NOSYS1DUMP parameters of the SETSYS command determine whether a SYS1DUMP type of dump is taken.

The DFSMSdss error message that triggers the dumps is controlled by the 3-byte MCVTDSSM field in the MCVT. A user wanting to trigger a dump enters the three digits of a DFSMSdss message in the 3-byte MCVTDSSM field. For example, the

following PATCH command causes a dump when DFSMShsm uses DFSMSdss as a data mover and DFSMSdss issues error message ADR305E:

```
PATCH .MCVT.+454 '305' VERIFY(.MCVT.+454 X'404040')
```

This function is also available when DFSMSdss is invoked during ABARS processing:

```
PATCH .ABRCB.+1C '308' VERIFY(.ABRCB.+1C X'404040')
```

When you have the produced the dumps you want, remove the patch to prevent future dumps from occurring.

Determining why SMS-managed data sets are not processed

If the following patches have been applied, then during volume migration or volume backup, DFSMShsm issues an ARC0734I message for each supported SMS-managed data set even though it does not meet the selection criteria. The reason code issued with the message explains why the SMS-managed data set does not qualify for selection. For interval migration, return code 45 with reason codes less than 90 are not produced regardless of the patch.

Entering the following PATCH command causes the extra messages to be issued for SMS-managed data sets that were not selected for volume migration:

```
PATCH .MGCB.+26 X'FF'
```

Entering the following PATCH command causes the extra messages to be issued for SMS-managed data sets that were not selected for volume backup:

```
PATCH .BGCB.+24 X'FF'
```

Increasing the amount of PDA tracing performed

Some PDA trace points are conditional, meaning that you can specify to DFSMShsm whether or not to write the trace point to the PDA output data set. Keep in mind that once you turn off the tracing, you may need to again enable tracing to recreate problems.

Table 4 lists the current conditional PDA trace points:

Table 4. Conditional PDA Trace Points

Trace Point	Offset	Initial Setting
Expanded data set backup tracing	.MCVT.+D9	BITS (1... ....) active MCVTF_DSBACKUP _TRACE
Cell pool usage	.MCVT.+558	BITS (1... ....) active MCVTFPDA_CELLS
Volume selection - volume rejected	.MCVT.+558	BITS (..1. ....) active MCVTFPDA_REJR

Analyzing the CELL POOL free chain

Many of the DFSMShsm internal control blocks use CELL POOL services to obtain and free storage. Unlike a FREEMAIN, CELL POOL services will not notify DFSMShsm if it erroneously returns a cell to the free cell chain. Instead, the return of the cell to the free cell chain is performed, allowing that storage to be erroneously used later by two processes.

The following patch enables diagnosis code execution that detects the return of a cell to the free cell chain. If DFSMShsm detects a second attempt to return a cell to the free cell chain, it will not perform the second attempt, and it issues an ARCERP code=8 with a snap dump. The field name is MCVTF_ZCELL_SNIF, the default is OFF and it slightly increases CPU utilization.

```
PATCH .MCVT.+558 BITS (...1 ....)
```

Chapter 10. Using DFSMShsm maintenance commands

There are four DFSMShsm maintenance commands presented in this section for the purpose of maintenance and diagnosis only:

- DISPLAY
- FIXCDS
- PATCH
- TRAP

DFSMShsm maintenance commands are submitted primarily by a DFSMShsm-authorized system programmer who uses the HSENDCMD command to issue these commands from a TSO session. They can also be submitted by a system operator from the system console.

For descriptions of other DFSMShsm commands, see *z/OS DFSMShsm Storage Administration*.

For descriptions of the DFSMShsm control data set records, see *z/OS DFSMShsm Data Areas*, which is available online at the z/OS Internet Library at http://www.ibm.com/systems/z/os/zos/bkserv/.

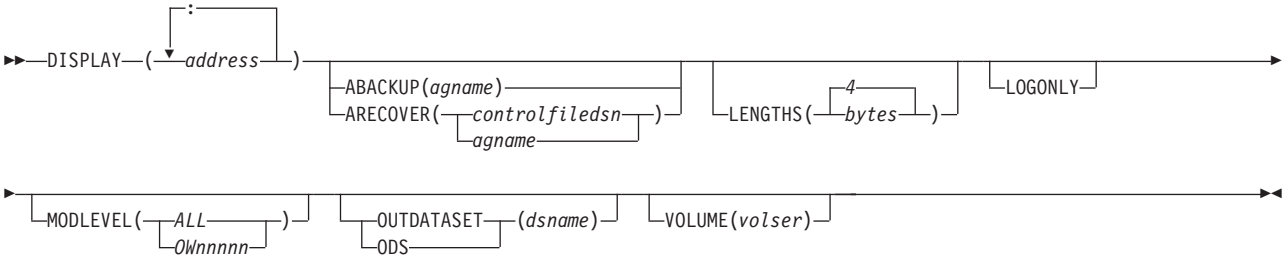
DISPLAY: Displaying DFSMShsm storage locations

The DISPLAY command displays locations within the DFSMShsm address spaces, including the ABARS secondary address spaces.

When you issue the DISPLAY command, you can display a storage location by specifying an absolute address or a qualified address. You can also specify the number of bytes you want displayed.

DFSMShsm lists all DISPLAY commands and their output at the terminal where you issue the command. You can optionally send the results to an output data set.

DISPLAY command syntax



Note: If you specify a list of addresses, you must enclose it in parentheses.

Required parameters of the DISPLAY command

The following sections provide information about the required parameters of the DISPLAY command.

Address: Specifying the location to display

Explanation: *address* [*:address*] is a required positional parameter specifying which locations in the DFSMShsm address space to display. You can specify one address, an address range, a list of separate addresses, or a list of address ranges.

For *address*, substitute the absolute address or the qualified address, of the location you want to display. You specify these addresses in the following manner:

- An absolute address is one to eight hexadecimal digits followed by a period (*hhhhhhhh.*).
- A qualified address is a DFSMShsm load module name, followed sequentially by a period, a CSECT name, a period, a plus sign and one to six hexadecimal digits (*loadmodulename.csectname.+hhhhhh*). Because ARCCTL is the only load module that can be displayed, the load name of ARCCTL is always the load module name if you supply only the CSECT name and offset (*.csectname.+hhhhhh*).

  Note: The leading period is still required.

The DISPLAY command considers the .MVT. control block identifier to be a valid CSECT name.

You can specify a range of addresses by specifying the optional second address [*:address*]. You must specify the colon (:) preceding the second *address* to show that you want to display a range of addresses. If you want to specify a list of separate addresses, do not use colons.

Abbreviations: None.

Defaults: None.

Note:

1. Because *address* is a required positional parameter, you must specify it immediately after the command name.
2. You cannot substitute indirect addressing, registers, expressions, and variable names for *address*.

Optional parameters of the DISPLAY command

The following sections provide information about the optional parameters of the DISPLAY command.

ABACKUP: Displaying storage locations within the DFSMShsm ABARS secondary address space

Explanation: ABACKUP(*agname*) is an optional parameter used to display storage locations within the DFSMShsm secondary address space. The ABACKUP parameter of the DISPLAY command is only valid when the DFSMShsm secondary address space is active.

When a DISPLAY ABACKUP command is issued, DFSMShsm verifies that the selected DFSMShsm secondary address space is active. If the selected DFSMShsm secondary address space is active, it is displayed. If the selected DFSMShsm secondary address space is not active, a message is issued and the processing of the DISPLAY command ends.

Specify *agname* (aggregate name) when the ABACKUP command is issued with the AGGREGATE parameter.

Abbreviations: None.

Defaults: None.

ARECOVER: Displaying storage locations within the DFSMShsm ABARS secondary address space

Explanation: ARECOVER(*controlfiledsn* | *agname*) is an optional parameter used to display storage locations within the DFSMShsm secondary address space. The ARECOVER parameter of the DISPLAY command is only valid when the DFSMShsm secondary address space is active.

When a DISPLAY ARECOVER command is issued, DFSMShsm verifies that the selected DFSMShsm secondary address space is active. If the selected DFSMShsm secondary address space is active, it is displayed. If the selected DFSMShsm secondary address space is not active, a message is issued and the processing of the DISPLAY command ends.

Use *controlfiledsn* (control file data set name) when the ARECOVER command is issued with the DATASETNAME parameter. Specify *agname* (aggregate name) when the ARECOVER command is issued with the AGGREGATE parameter.

Abbreviations: None.

Defaults: None.

LENGTHS: Specifying how much data to display

Explanation: LENGTHS(*bytes ...*) is an optional parameter specifying the number of bytes to display. For *bytes*, substitute a decimal number from 1 to 999999 for the length of the data you want to display.

You can specify a list of bytes that match the specified addresses. For example, the first length you specify applies to the first address, the second length applies to the second address, and so forth. If you specify fewer lengths than addresses, the last length you specify applies to the remaining addresses. The **LENGTHS** parameter does not specify how much data to display for an address range. If you specify a length with an address range, DFSMShsm pairs the length and address range, but it only uses the address range to determine how much to display.

For example, if you specify the addresses as:

```
8EC.,.ARCZWRIT.+4,.ARCBACK.+0
```

where:

- 8EC. is a separate absolute address
- .ARCZWRIT.+4 is a separate qualified address
- .ARCBACK.+0 is another separate qualified address

and you specify the lengths as:

```
LEN(2,4,3)
```

then:

- 2 bytes of data, starting at absolute address 8EC, are displayed
- 4 bytes of data, starting at qualified address .ARCZWRIT.+4, are displayed

- 3 bytes of data, starting at qualified address .ARCBACK.+0, are displayed

Abbreviations: None.

Defaults: If you do not specify LENGTHS, the default is four bytes if the address is *not* the mounted volume table (MVT) control block. If you do not specify LENGTHS when specifying .MVT., the remainder of the MVT control block is displayed.

LOGONLY: Specifying that data not go to the terminal

Explanation: LOGONLY is an optional parameter specifying that the output from the DISPLAY command not go to the terminal of the issuer of the DISPLAY command. All output will be sent to the data set specified with the OUTDATASET command parameter.

Abbreviations: None.

Defaults: If you do not specify LOGONLY, the output from the DISPLAY command goes to the terminal from where you issued the command.

MODLEVEL: Displaying maintenance levels of modules

Explanation: MODLEVEL is an optional parameter that is used to format the maintenance level and compile date/time stamps for either one module or all modules in DFSMShsm. This parameter is only valid if the address is specified as '.ARCxxxxx.+0'. The LENGTH keyword is ignored when you specify the MODLEVEL keyword.

If **MODLEVEL(ALL)** is specified, the maintenance levels of all modules in the DFSMShsm primary address space, and separate load modules such as ARCUTIL and ARCGIVER, are displayed. Modules in the ARCWCTL load module are not included. The **(ALL)** option overrides the address parameter, but you must still include the address for the command to be valid. The output of this display command can be saved by including the OUTDATASET keyword.

Do not use MODLEVEL(ALL) with the DISPLAY ABACKUP or DISPLAY ARECOVER commands.

MODLEVEL(OW*nnnnn*) is an optional parameter used to display all modules, except ABARS modules, that have a maintenance level matching the specified APAR number. The MODLEVEL(OW*nnnnn*) parameter does not apply to ABARS modules.

Abbreviations: None.

Defaults: None.

OUTDATASET: Specifying the output location for the DISPLAY results

Explanation: OUTDATASET(*dsname*) is an optional parameter specifying the name of the data set that you want DFSMShsm to write the output data to.

For *dsname*, substitute the fully qualified name of the data set that is to receive the DISPLAY results. If the data set does not exist, DFSMShsm dynamically allocates and catalogs an output data set with the following characteristics:

- Data set name as specified (dsname)
- Record format of FBA

- Logical record length of 121
- Primary allocation of 20 tracks (see note)
- Automatic secondary allocation of 50 tracks (see note)
- Unit type of SYSALLDA (see note)
- System reblockable

If the data set already exists:

- It must be cataloged and on DASD.
- The record format must be FBA.
- The logical record length must be 121.
- The data set must be system reblockable.
- The user can choose the automatic primary space allocation.
- If DFSMShsm needs additional extents after the automatic primary space allocation, DFSMShsm uses an automatic secondary space allocation of 50 tracks (see note).
- If the data set does not contain data, DFSMShsm starts writing output data at the beginning of the data set.
- If the data set contains data, DFSMShsm writes the output data after the existing data.

Note: You can use the PATCH command to change the unit name, primary allocation, and secondary allocation. For more information, see the topic on tuning DFSMShsm in *z/OS DFSMShsm Implementation and Customization Guide*.

Abbreviations: You can use the abbreviation ODS for OUTDATASET.

Defaults: None.

VOLUME: Displaying a mounted volume table with a specific volume serial number

Explanation: VOLUME(*volser*) is an optional parameter that is used to limit the display to a specific MVT that contains a matching volume serial number for a DFSMShsm-managed level 0 or migration level 1 volume. This parameter is only valid if the address is specified as .MVT.

Abbreviations: None.

Defaults: If the *volser* parameter is not specified when the address is .MVT., all MVT entries for SMS and non-SMS-managed volumes (including migration level 1) will be displayed.

Examples of how to code the DISPLAY command

The following table lists examples of different ways to code the DISPLAY command.

Note: Any values specified here are examples only and should not be interpreted as the specific values to be used for your system.

Displaying the contents of a range of absolute addresses

Example: This example displays the contents of a range of DFSMShsm absolute addresses:

```
DISPLAY 03606258.:03606278.
```

Displaying the contents at a qualified address

Example: This example displays 120 bytes at each of two places in the DFSMShsm address space. The output goes to the terminal that issued the command:

```
DISPLAY (.ARCALVOL.+3D2,.ARCALVOL.+7A) LENGTHS(120)
```

See Figure 3 for an example of the output as it would appear on your terminal screen.

```
ARC0300I **OPER-  ISSUED===>DISPLAY (.ARCALVOL.+3D2,
 .ARCALVOL.+7A) LENGTHS(120)
ARC0206I 0399573A  40388010 47F0C43C 58809048 50809114
ARC0206I (CONT.) 5880904C 50809118 41809214 5080911C
ARC0206I 0399575A  58F0A438 41109114 05EF5880 905041E0
ARC0206I (CONT.) 000850E0 800047F0 C43C5880 904C5880
ARC0206I 0399577A  800058E0 802812EE 4780C3EA D505E01C
ARC0206I (CONT.) 801A4780 C42C5880 904C5880 80004180
ARC0206I 0399579A  801A5080 91144180 91FC5080 911858F0
ARC0206I (CONT.) A4244110 911405EF
ARC0206I 039953E2  D20B9048 10005880 A4441288 4780C060
ARC0206I (CONT.) 58808000 12884780 C06058F0 A4444110
ARC0206I 03995402  A0F005EF D20792FB A49A4180 925C5080
ARC0206I (CONT.) 91E89680 91E8D713 80008000 92148000
ARC0206I 03995422  92018001 58E0A41C 9180E000 4770C094
ARC0206I (CONT.) 96208010 96208011 4180925C 41E09568
ARC0206I 03995442  50E08008 41809290 5080E000 D22F9290
ARC0206I (CONT.) A5344180 00014080
ARC0190I DISPLAY COMPLETE
```

Figure 3. Example of Displaying the Contents at a Qualified Address

Displaying the last date that automatic primary space management ran

Example: This example displays the last date automatic primary space management ran to completion:

```
DISPLAY .MCR.+98 LENGTHS(4)
```

See Figure 4 for an example of the last Julian date that automatic primary space management ran as it would appear on your terminal screen.

```
ARC0300I **OPER-  ISSUED===>DISPLAY .MCR.+98 LENGTHS(4)
ARC0206I 03803B8C  0094028F
ARC0190I DISPLAY COMPLETE
```

Figure 4. Example of Displaying the Last Date Automatic Primary Space Management Ran

Displaying the last date migration cleanup ran

Example: This example displays the last date migration cleanup ran:

```
DISPLAY .MCR.+48 LENGTHS(4)
```

See Figure 5 on page 101 for an example of the last Julian date that migration cleanup ran as it would appear on your terminal screen.

```
ARC0300I **OPER-  ISSUED===>DISPLAY .MCR.+48 LENGTHS(4)
ARC0206I 03803B3C  0094028F
ARC0190I DISPLAY COMPLETE
```

Figure 5. Example of Displaying the Last Date Migration Cleanup Ran

Displaying the last date automatic backup ran

Example: This example displays the last date automatic backup ran:

```
DISPLAY .BCR.+50 LENGTHS(4)
```

See Figure 6 for an example of the last Julian date that automatic backup ran as it would appear on your terminal screen.

```
ARC0300I **OPER-  ISSUED===>DISPLAY .BCR.+50 LENGTHS(4)
ARC0206I 03804344  0094028F
ARC0190I DISPLAY COMPLETE
```

Figure 6. Example of Displaying the Last Date Automatic Backup Ran

Displaying the last date level 1 functions ran

Example: This example displays the last date the level 1 functions of moving backup versions and backing up migrated data sets ran:

```
DISPLAY .BCR.+5C LENGTHS(4)
```

See Figure 7 for example of the last Julian date that the level 1 functions ran as it would appear on your terminal screen.

```
ARC0300I **OPER-  ISSUED===>DISPLAY .BCR.+5C LENGTHS(4)
ARC0206I 03804350  0094028F
ARC0190I DISPLAY COMPLETE
```

Figure 7. Example of Displaying the Last Date Level 1 Functions Ran

Displaying the last date that automatic dump functions ran

Example: This example displays the last date the automatic dump functions ran:

```
DISPLAY .DCR.+5A LENGTHS(4)
```

See Figure 8 for an example of the last Julian date that automatic dump functions ran as it would appear on your terminal screen.

```
ARC0300I **OPER-  ISSUED===>DISPLAY .DCR.+5A LENGTHS(4)
ARC0206I 03804BCE  0094028F
ARC0190I DISPLAY COMPLETE
```

Figure 8. Example of Displaying the Last Date Automatic Dump Functions Ran

Displaying the contents of the MCVT

Example: This example displays the first 60 bytes of the management communication vector table:

```
DISPLAY .MCVT.+0 LENGTHS(60)
```

See Figure 9 for an example of displaying the MCVT as it would appear on your terminal screen.

```
ARC0300I **OPER-  ISSUED===>DISPLAY .MCVT.+0
 LENGTHS(60)
ARC0206I 038027D0  1E010610 00FA2068 00EFC028 00FA7DA0
ARC0206I (CONT.) 00FA7D98 0000C3D8 0000B330 00963228
ARC0206I 038027F0  40000000 00000000 00000000 00000000
ARC0206I (CONT.) 03805374 0001D140 0001D140
ARC0190I DISPLAY COMPLETE
```

Figure 9. Example of Displaying the Contents of the MCVT

Displaying a specific MVT element

Example: This example displays the entire MVT entry for a DFSMShsm-managed volume with a volser of SMS001:

```
DISPLAY .MVT.+0 VOLUME(SMS001)
```

Displaying data and sending the results to an output data set

Example: This example displays the module ARCCTL at hex offset 0. The displayed data is added to the data set **USERID.DISPLAY.DATA** and is also shown at the terminal of the issuer of the DISPLAY command:

```
DISPLAY .ARCCTL.+0 LEN(120) ODS(USERID.DISPLAY.DATA)
```

See Figure 10 for the example as shown on your terminal screen.

```
ARC0300I **OPER-  ISSUED===>DISPLAY .ARCCTL.+0
LEN(120) ODS(USERID.DISPLAY.DATA)
ARC0206I 03832C10  47F0F02A 25C1D9C3 C3E3D340 404BF9F4
ARC0206I (CONT.) F0F0F74B F1F5F1F8 F4F04BD6 E6F0F1F3
ARC0206I 03832C30  F9F540E5 C5D97EF1 F2F090EC D00C18CF
ARC0206I (CONT.) 41B0CFFF 41A0BFFF 4190AA9E 41709FFF
ARC0206I 03832C50  41607FFF 50D09200 41E091FC 50E0D008
ARC0206I (CONT.) 18DE5880 94B44100 9C4C5000 80D05010
ARC0206I 03832C70  96005830 96904120 000A0700 47F0C074
ARC0206I (CONT.) 00000200 180318F2
ARC0190I DISPLAY COMPLETE
```

Figure 10. Example of Displaying the Module ARCCTL

Example: This example displays the module ARCCTL at hex offset 0. The displayed data is added to the data set **USERID.DISPLAY.DATA** but is *not* shown at the terminal of the issuer of the DISPLAY command:

```
DISPLAY .ARCCTL.+0 ODS(USERID.DISPLAY.DATA) LOGONLY
```

Displaying the maintenance level

Example: This example displays the maintenance level of module ARCBGEN. The displayed data is *not* added to any output data set but is shown at the terminal of the issuer of the DISPLAY command:

```
DISPLAY .ARCBGEN.+0 MODLEVEL
```

Example: This example displays the maintenance level of module ARCWOPEN in the ABARS secondary address space. The displayed data is not added to an output data set, but is shown at the terminal of the issuer of the DISPLAY command:

```
DISPLAY .ARCWOPEN.+0 MOD ABACKUP(agname)
```

Example: This example displays the maintenance levels of all modules in the DFSMShsm primary address space. The displayed data is added to the data set **USERID.DISPLAY.DATA** but is *not* shown at the terminal of the issuer of the DISPLAY command:

```
DISPLAY .ARCCTL.+0 MODLEVEL(ALL) ODS(USERID.DISPLAY.DATA) LOGONLY
```

Figure 11 shows an example of browsing part of the out data set (ODS) on your terminal screen.

```
 BROWSE -- USERID.DISPLAY.DATA ----------- LINE 00000000 COL 001 080
 COMMAND ===>                                          SCROL==> CSR
******************************** TOP OF DATA **********************
DISPLAY .ARCCTL.+0 MODLEVEL(ALL) ODS(USERID.DISPLAY.DATA) LOGONLY
ARCABDAS HDZ1B10 08.255 14:41:43
ARCABMCB HDZ1B10 08.255 14:41:40
ARCABMIG HDZ1B10 08.255 14:41:39
ARCABMSG HDZ1B10 08.240 14:25:58
ARCABNMB HDZ1B10 08.255 14:41:39
ARCABTP1 HDZ1B10 08.255 14:41:39
ARCABTP2 HDZ1B10 08.240 14:25:58
ARCACASS HDZ1B10 08.240 14:26:05
ARCACBER HDZ1B10 08.240 14:26:05
ARCACBVR HDZ1B10 08.240 14:26:05
ARCACERR HDZ1B10 08.240 14:26:05
ARCACLN HDZ1B10 08.240 14:26:07
ARCACMCT HDZ1B10 08.240 14:26:05
ARCACTL HDZ1B10 08.240 14:26:08
ARCACTTC HDZ1B10 08.240 14:26:07
ARCADBKM HDZ1B10 08.255 14:41:39
```

Figure 11. Example of Displaying the Maintenance Levels

Displaying the MVT entry

Example: This example displays the entire MVT entry for each MVT on the SMS and non-SMS MVT chains:

```
DISPLAY .MVT.+0
```

Displaying a storage location within the ABARS secondary address space

Example: This example displays the module ARCWCTL at hex offset 0 with the aggregate name of AGM1.C.C01V0001.

```
DISPLAY .ARCWCTL.+0 ARECOVER(AGM1.C.C01V0001)
```

See Figure 12 on page 104 for an example of the ARCWCTL module as it would appear on your terminal screen.

```
ARC0300I **OPER-  ISSUED===>DISPLAY .ARCWCTL.+0
 ARECOVER(AGM1.C.C01V0001)
ARC0206I 00051968  47F0F02A
ARC0190I DISPLAY COMPLETE
```

Figure 12. Example of Module ARCWCTL

FIXCDS: Displaying, creating, or modifying a record in the MCDS, BCDS, or OCDS

The FIXCDS command displays or modifies records in the migration control data set (MCDS), backup control data set (BCDS), or offline control data set (OCDS). You can display a record in one of the control data sets with FIXCDS and then by issuing FIXCDS again, make changes to that record. A record consists of a key field (44 bytes), followed by a header field (20 bytes), followed by variable data.

Use the FIXCDS command to fix minor problems that occur in the control data sets. For example, you can use the FIXCDS command when the audit process finds a discrepancy between the computing system catalog and the MCDS.

Use the DISPLAY parameter of the FIXCDS command to display the data before you change it. Then, after you change a control data set record, verify that the change was successful by again specifying the DISPLAY parameter of the FIXCDS command.

Although DFSMShsm keeps copies of the following records in working storage, the changes you specify with the FIXCDS command for these types of records are made only to the DASD copy of the following records:

- Backup control record (BCR)
- Daily statistics record (DSR)
- Migration level-2 control record (L2CR)
- Management control record (MCR)
- JES3 volume activity count record (VAC)
- Volume statistics record (VSR)

If you want the virtual storage copy of the above records to keep the changes you make to the records, use the PATCH command. Otherwise, the changes you made to the CDS records with the FIXCDS command are replaced with the unchanged records from the virtual storage copy.

Use the FIXCDS command to make the following changes to the MCDS, BCDS, and OCDS:

- Delete a control data set record (DELETE).
- Expand a control data set record by a specified number of bytes (EXPAND).
- Rename a control data set record by specifying a new key (NEWKEY).
- Change existing data at the specified offset in a control data set record (PATCH).
- Create a new record in one of the control data sets (CREATE).

Use the FIXCDS command to make the following changes only to the MCDS:

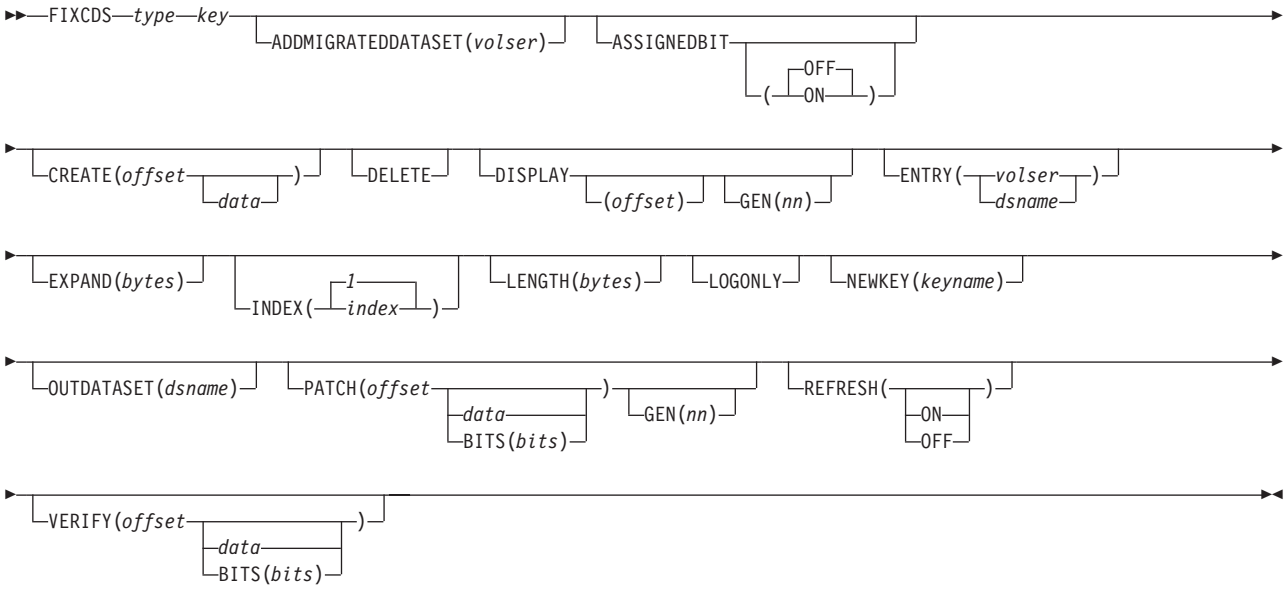
- Add a new MCDS data set record for a migrated data set residing on the volume that you specify (ADDMIGRATEDDATASET).
- Turn the assigned bit on or off in the MCDS for a migrated data set record to show the migration status of a data set (ASSIGNEDBIT).

Use the FIXCDS command to request the refresh of all of the BVRs in the BCDS (REFRESH) during the next DFSMShsm CDS backup.

You can request a printout of the SYSOUT data set.

Note: For information on DFSMShsm supported patches, refer to *z/OS DFSMShsm Implementation and Customization Guide* .

FIXCDS command syntax



Note:

1. You can specify one of ADDMIGRATEDDATASET, ASSIGNEDBIT, CREATE, DELETE, DISPLAY, ENTRY, EXPAND, NEWKEY, REFRESH, VERIFY, or PATCH with each FIXCDS command.
2. If you specify GEN(*nn*) with the DISPLAY or PATCH parameters, the required parameter *type* must be 'C'.

Required parameters of the FIXCDS command

The following sections provide information about the required parameters of the FIXCDS command.

type: Specifying the type of control data set record

Explanation: *type* is a required positional parameter for which you substitute the alphameric 1-character record type identification for the control data set record you want to fix or display. Table 5 shows the record types in the MCDS, BCDS, and OCDS:

Table 5. Records of the Control Data Sets

1 Character Record Type	3 or 4 Character Record Type	Record Type Name
A	MCA	MCDS alias entry record
B	MCB	BCDS data set record
C	MCC	BCDS backup version record

Table 5. Records of the Control Data Sets (continued)

1 Character Record Type	3 or 4 Character Record Type	Record Type Name
D	MCD	MCDS data set record
E	TCN	OCDS tape copy needed record
F	FRB	BCDS fast replication backup record
G	DGN	BCDS dump generation record
H	FRVP	BCDS fast replication volume pairs record
I	FRTV	BCDS fast replication target volume record
J	FRSV	BCDS fast replication source volume record
K	FRD	BCDS fast replication dump record
L	MCL	BCDS backup changed migrated data set record
M	MCM	BCDS move backup version record
N	VAC	MCDS JES3 volume activity count record
O	MCO	MCDS VSAM association record
P	MCP	BCDS eligible volume record
Q	ABR	BCDS aggregate group version record
R	BCR	BCDS backup control record
	BVR	BCDS backup cycle volume record subdivided into spill, unassigned, day of backup cycle (daily)
	DCR	BCDS dump control record
S	L2CR	MCDS migration level 2 control record
	MCR	MCDS management control record
	DSR	MCDS daily statistics record
	MHCR	MCDS multiple-processing-unit control record
	VSR	MCDS volume statistics record
T	TTC	OCDS tape table of contents record subdivided into spill, unassigned, day of backup cycle (daily), and migration level 2
U	MCU	MCDS user record
V	MCV	MCDS volume record
W	DCL	BCDS dump class record
X	MCT	BCDS backup volume record
Y	DVL	BCDS dump volume record
1	MC1	MCDS migration level 1 free space record
Z	MCBR	BCDS data set record for retained backup copies

Abbreviations: None.

Defaults: None.

Note: Because *type* is a required positional parameter, you must specify it immediately after the command name.

key: Specifying the control record key

Explanation: *key* is the required positional parameter specifying the control record key for the control data set record you want to fix or display. The key can be specified in EBCDIC or hexadecimal characters.

Note:

1. Record types E, F, G, H, I, J, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, and 1 have a 1-byte hexadecimal identification as an internal first byte. Do not include the 1-byte hexadecimal identification in the *key* field.
2. For deletion of an L record, specify the user's data set name instead of the key.

The following are keys for types A, B, C, and D records:

Record Type	Key
A	Migrated data set name (non-VSAM) created when DFSMShsm migrates the data set or the name of a VSAM component of a migrated data set eligible for automatic recall.
B	Data set name of a user data set.
C	Backup version data set name created when DFSMShsm backs up the data set.
D	Data set name of a user data set.

A, B, C, and D data set records: The following are examples of the keys used with A, B, C, and D data set records:

```
FIXCDS A HSM.HMIG.T231510.USER.DATA.H4060
FIXCDS A VSAM.PATHNAME
FIXCDS B USER.DATA.NAME
FIXCDS C HSM.BACK.T352016.DATA.NAME.H4323
FIXCDS D USER.DATA.NAME
```

E tape copy needed record: The key for a type E tape copy needed record is either "M" for a migration tape or "B" for a backup tape, followed by a dash (-), then the 6–byte volume serial number, padded with blanks. Examples of the key used with an E tape copy needed record are:

```
FIXCDS E M-A00300
FIXCDS E B-B01401
```

F fast replication backup record: The key for a type F fast replication backup record is the name of the copy pool that the record represents. An example of the key used with a type F fast replication backup record is:

```
FIXCDS F COPYPOOL1
```

G dump generation record: The key for the type G dump generation record is the volume serial number followed by the time of the day (*hhmmssth*) in packed decimal format. The time of day is followed by the year and day (*0cyyddd*F) in packed decimal format, where:

C The digit representing the century.

0=1900–1999

1=2000–2099

YY The last two digits of the year.

DDD The day of the year.

F A four-bit character that allows the data to be unpacked and printed.

Because the time and date are in packed decimal format, you must specify the key in hexadecimal. An example of the key used for a dump generation record on volume DUMP01 created at 12:55 p.m. on day 135 of the year 1997 is:

```
FIXCDS G X'C4E4D4D7F0F1125519490097135F'
```

H fast replication volume pairs record: The following are fields that make up a key for a type H fast replication volume pairs backup record:

Characters	Description
1 to 30	The copy pool name suffixed with periods, if the copy pool name is less then 30 characters in length.
31	The type of record. 'P' if the record is a prepare record. 'B' if the record is a backup record.
32 and 33	The version number in hexadecimal format.
34 to 41	The storage group name suffixed with periods, if the storage group name is less than 8 characters in length.
42	The extent number.

The following is an example of the key (in hexadecimal) that is used for a type H fast replication volume pairs backup record. In this example, we are using a key that consists of the following:

- A copy pool name of COPYPOOL1 (followed by periods to fill field)
- A 'B' for a backup record request
- A version number of 1
- A storage group name of SRCSG1 (followed by periods to fill field)
- An extent number of zero

```
FIXCDS H X'C3D6D7E8D7D6D6D3F14B4B4B4B4B4B4B4B4B4B4B4B4B4B4B4B4B4B4BC20001E2C7D9D7F14B4B4B0001'
```

I fast replication target volume record: The key for a type I fast replication target volume record is the volume serial number. An example of the key used with a type I fast replication target volume record is:

```
FIXCDS I TAR097
```

J fast replication source volume record: The key for a type J fast replication source volume record is the volume serial number. An example of the key used with a type J fast replication source volume record is:

```
FIXCDS J SMS275
```

K fast replication dump record: The following are fields that make up a key for a type K fast replication dump record:

Characters	Description
1 to 30	The copy pool name suffixed with periods, if the copy pool name is less then 30 characters in length.
31 and 32	The version number in hexadecimal format.

The following is an example of the key (in hexadecimal) that is used for a type K fast replication dump record. In this example, we are using a key that consists of the following:

- A copy pool name of COPYPOOL1 (followed by periods to fill field)
- A version number of 1

```
FIXCDS K X'C3D6D7E8D7D6D6D3F14B4B4B4B4B4B4B4B4B4B4B4B4B4B4B4B4B4B4B4B4B0001'
```

L backup migrated data set record: The key for a type L backup migrated data set record is the first 43 characters of the data set name. Under certain circumstances, DFSMShsm changes the 44th byte of the key. As a result, the key to the record might not correspond exactly with the first 43 characters of the data set name. The true 43rd character and the 44th character of the data set name are stored in the data portion of the record. An example of the key used with a type L backup migrated data set record is:

```
FIXCDS L USER.DATA.NAME
```

M move backup version record: The key for a type M move backup version record is 43 characters of the backup version name. DFSMShsm uses all characters except the fourth character of the second qualifier, which is always a K. An example of the key used with an M move backup version record is:

```
FIXCDS M USER.BAC.T280112.JCL.CNTL.H4104
```

N volume activity count record: The key for a type N volume activity count record is the constant VOLCNT followed by a dash (-) and *n*, where *n* is a decimal number that indicates the sequence number of the record. The N record is an internal record used to maintain activity counts on primary volumes in a JES3 environment. An example of the key used with an N data set record is:

```
FIXCDS N VOLCNT-4
```

O VSAM data set record: The key for a type O VSAM data set record is the migrated data set name created by DFSMShsm when it migrates a VSAM data set. An example of the key used with an O data set record is:

```
FIXCDS O HSM.HMIG.T231510.USER.DATA.H4079
```

P, V, X, and Y volume records: The key for types P, V, X, and Y volume records is the volume serial number. Examples of the keys used with P, V, X, and Y volume records are:

```
FIXCDS P TSO456
FIXCDS V TSO123
FIXCDS X TSO789
FIXCDS Y TSO444
```

Q aggregate group version record: The key for type Q aggregate group version record corresponds to the control file name, version, and local copy number. The next available version number for the ABACKUP output files and ABR key is determined by DFSMShsm. If all available version numbers (1–9999) have been used, a message is issued and the ABACKUP command is failed. An example of the key used with Q aggregate backup and recovery record is:

```
FIXCDS Q AGNAME.yyyydddnnnncc
```

Where:

- *agname* is the name of the aggregate group.
- *yyyyddd* indicates the date of the aggregate backup. *yyyy* is the 4 digit year, and *ddd* is the relative (Julian) day of the year.
- *nnnn* indicates the version number of the aggregate group version. This maintains a one-to-one correspondence between an ABR record and an aggregate backup version.
- *cc* indicates the local copy number described by this record.

R backup control record: The key for a type R backup control record is the constant BCR and the processing unit identification in a multiple DFSMShsm-host environment. The processing unit identification is a 1-digit alphameric character. If you omit the processing unit identification, DFSMShsm defaults to the ID of the issuing processing unit. An example of the key used with an R backup control record is:

```
FIXCDS R BCR3
```

R backup cycle volume record: The key for a type R backup cycle volume record when it is used with the REFRESH parameter is BVR. An example of the key used with an R backup cycle volume record to be refreshed is:

```
FIXCDS R BVR REFRESH(ON)
```

For any other FIXCDS command, the key for a type R backup cycle volume record is the constant BVR followed by two characters representing the type of record (01–31 for daily, SP for spill, and UN for unassigned), a dash (-), and a four-character sequence number. An example of the key that is used with an R backup cycle volume record for day 1 of the backup cycle is:

```
FIXCDS R BVR01-0000
```

R dump control record: The key for the type R dump control record is the constant DCR and the processing unit identification in a multiple DFSMShsm-host environment. The processing unit identification is a 1-digit alphameric character. If you omit the processing unit identification, DFSMShsm defaults to the identification of the issuing processing unit. An example of the key used with an R dump control record is:

```
FIXCDS R DCR2
```

S level-2 control record: The key for a type S level 2 control record is the constant L2CR. An example of the key used with an S level 2 control record is:

```
FIXCDS S L2CR
```

S management control record: The key for a type S management control record is the constant MCR and the processing unit identification in a multiple DFSMShsm-host environment. The processing unit identification is a 1-digit alphameric character. If you omit the processing unit identification, DFSMShsm defaults to the ID of the issuing processing unit. An example of the key used with an S management control record is:

```
FIXCDS S MCR2
```

S daily statistics record: The key for a type S daily statistics record is the constant X'C4E2D9' (DSR) followed by the year and day in packed decimal format. The sign code is F. Because the date is in packed decimal format, you must specify the key in hexadecimal. An example of the key used with an S daily statistics record for day 76 of 1984 is:

```
FIXCDS S X'C4E2D984076F'
```

S multiple-processing-unit control record: The key for a type S multiple-processing-unit record is the constant MHCR. An example of the key used with an S multiple-processing-unit record is:

```
FIXCDS S MHCR
```

S volume statistics record: The key for a type S volume statistics record is the constant X'E5E2D9' (VSR) followed by the volume serial number, which is followed by the year and day in packed decimal format. The sign code is F. Because the date is in packed decimal format, you must specify the key in hexadecimal. An example of the key used for volume 123400 for its use on day 121 of 1984 with an S volume statistics record is:

```
FIXCDS S X'E5E2D9F1F2F3F4F0F084121F'
```

T tape table of contents record (migration level-2 volume): The key for a type T tape table of contents record for a tape migration level-2 volume is the constant L2 followed by a dash (-), the volume serial number, a dash (-), and a four-character sequence number. An example of the key used with a T tape table of contents record for the migration level-2 volume TML205 is:

```
FIXCDS T L2-TML205-0000
```

T tape table of contents record (backup volume): The key for a type T tape table of contents record for a tape backup volume is a two-character representation of the volume assignment (01–31 for daily, SP for spill, and UN for unassigned), followed by a dash (-), the volume serial number, a dash (-), and a four-character sequence number. An example of the key used with a T tape table of contents record for spill volume TAPE01 is:

```
FIXCDS T SP-TAPE01-0000
```

U user record: The key for a type U user record is the user identification. An example of the key used with a U user record is:

```
FIXCDS U SLJ2345
```

W dump class record: The key for the type W dump class record is the dump class name, which includes from one to eight alphameric characters. An example of the key used with a W dump class record is:

```
FIXCDS W DCLASS01
```

Z data set record for retained backup copies: The MCBR record is generated from the MCB key using a conversion algorithm. Contact IBM Support for assistance in generating the MCBR key. The MCBR records can also be accessed by using the data set name and INDEX keyword. Refer to the INDEX keyword for more information.

```
FIXCDS Z X'C4C6C8E250B572C0B1018C1C82FFFF'
```

Type 1 record: The key for a type 1 record is the constant L1VOL, a dash (-), and a two-character sequence number representing the record sequence. The record sequence number for the first record is 00, the number for the second record is 01, and so forth. Type 1 records are always created sequentially. An example of the key used with a type 1 migration level-1 free space record is:

```
FIXCDS 1 L1VOL-00
```

Abbreviations: None.

Defaults: None.

Note:

1. Because *key* is a required positional parameter, you must specify it immediately after the *type* parameter.
2. When you specify the key as hexadecimal, use an even number of hexadecimal digits. If you specify an odd number of hexadecimal digits, DFSMShsm inserts a zero to the left of the value to make it an even number of hexadecimal digits. Hexadecimal characters must be in the form *X'n'*.

Optional parameters of the FIXCDS command

The following sections provide information about the optional parameters of the FIXCDS command.

ADDMIGRATEDDATASET: Adding an MCDS record for a migrated data set

Explanation: ADDMIGRATEDDATASET(*volser*) is an optional parameter specifying that a new type D MCDS record be added for a migrated data set residing on the specified volume. The record contains only the minimum information about the data set. See "Adding a migrated data set record to the MCDS" on page 122 for an example of how to use the ADDMIGRATEDDATASET parameter of the FIXCDS command to add an MCDS record.

To make the data set eligible for recall, you must issue the PATCH parameter of the FIXCDS command to patch the data set's fully-qualified migration name (MCDMCANM) padded to the right with blanks.

Field Name	FIXCDS Offset	Explanation
MCDMCANM	156 or X'9C'	The name of the migration copy of the data set that is the key of the MCA record.

In addition, it is necessary to patch the following fields:

Field Name	FIXCDS Offset	Explanation
MCDFRVSN	64 or X'40'	The volume serial number of the primary volume from which the data set has been migrated.
MCDUCBTY	76 or X'4C'	The device type of the primary volume from which the data set has been migrated.

See "Verifying and patching a migrated data set record" on page 123 for examples of how to use the PATCH parameter of the FIXCDS command to patch an MCDS.

If the data set is migrated to tape and the data set spans more than one tape, the following fields must be filled in:

Field Name	FIXCDS Offset	Explanation
MCDTPDEV	90 or X'5A'	The bit setting for migrated copies that reside on a tape device.
MCDNVSN	212 or X'D4'	A 2-byte binary number of tape volumes after the tape volume where the data set started.
MCDNVSNO	214 or X'D6'	A binary number representing the offset from the MCDVSN field to the MCDAVSN field. This value can be calculated by issuing the DISPLAY parameter of the FIXCDS command and using the offsets displayed on the screen.

To add the extra volumes to the record, you must first use the EXPAND parameter of the FIXCDS command to increase the length of the D record. The D record must increase 6 bytes for each additional volume serial number that needs to be included. Enter the volume serial numbers of the other tape volumes in the expanded area. The first volume serial number starts at offset +0 in the expanded area, the second at offset +6, and so on. Pad the volume serial number with blanks to the right.

For *volser*, substitute the volume serial number of the migration level-1 or DASD migration level-2 volume where the migrated data set resides. For tape migration level-2 volumes, substitute the volume serial number of the volume where the migrated data set starts.

Note: This is only true when the data sets are migrated to DASD.

Abbreviations: None.

Defaults: If you do not specify one of the optional parameters of FIXCDS, the default is DISPLAY.

Note:

1. You can specify the ADDMIGRATEDDATASET parameter only for a type D record. The command fails if you issue it for another record type.
2. To recall the migrated data set after you have added the new type D MCDS record, add MCDMCANM padded by blanks, to the D record. This only applies to multifile format using migration level-2 tape.

ASSIGNEDBIT: Turning the assigned bit on or off

Explanation: ASSIGNEDBIT(ON | OFF) is an optional parameter specifying whether to turn the assigned bit on or off in an MCDS type D record. The assigned bit indicates whether the data set has migrated.

ON specifies that you want to turn on the assigned bit in the type D record. This bit indicates that the data set has migrated.

OFF specifies that you want to turn off the assigned bit in the type D record. This bit indicates that DFSMShsm has recalled the data set.

Abbreviations: None.

Defaults: If you do not specify one of the optional parameters of FIXCDS, the default is DISPLAY. If you do not specify ON or OFF with the ASSIGNEDBIT parameter, the default is OFF.

Note:

1. You can specify the ASSIGNEDBIT parameter only for a type D record. The command fails if you issue it for another record type.
2. You can specify either ON or OFF, but not both.

CREATE: Creating a new control data set record

Explanation: CREATE(*offset data*) is an optional parameter specifying that you want to create a new data set. The LENGTH parameter controls the size of the record to be created.

For *offset*, substitute a decimal number for the offset where you want the data inserted. The number you specify can be any number from zero up to a number equal to the length of your CDS record. You specify the data for the variable portion of the record. The FIXCDS command automatically creates the key and header portions of the record.

For *data*, substitute hexadecimal characters or alphameric characters and $, #, or @ to be inserted into the new record. You cannot specify more than 256 bytes (512 hexadecimal digits), and you should specify an even number of hexadecimal digits. If you specify an odd number of hexadecimal digits, DFSMShsm inserts a zero to the left of the value to make it an even number. Hexadecimal characters must be in the form *X'n'*. The new record contains all zeros after the header field, unless you supply data.

Abbreviations: None.

Defaults: If you do not specify one of the optional parameters of FIXCDS, the default is DISPLAY.

Note:

1. The offset of the first byte of the variable data of the record is zero. The offset of the last byte of the variable data of the record is one less than the length of the variable data.
2. If you are creating a tape table of contents (TTOC) extension record, be sure to update the TTCNUM field for the TTOC base record for the volume. This halfword field is the total count of the base and the extension records for the volume.

   If you are creating a migration level-1 free space (MC1) or JES3 volume activity count (VAC) continuation record, be sure to patch the continuation flag in the appropriate lower-level record to binary '1'.
3. If you are creating an MCL record, an associated MCD record must currently exist. If the MCD record does not exist, the FIXCDS command is failed. If the MCD does exist, the MCD record is updated to contain the 43rd character of the newly created MCL record key.
4. If you are creating an MCL record, the first two bytes of the record cannot be changed. This area contains the 43rd and 44th character of the data set name. If this area is specified to be updated, the FIXCDS command is failed.

DELETE: Deleting a control data set record

Explanation: DELETE is an optional parameter specifying that you want a control data set record deleted.

Abbreviations: None.

Defaults: If you do not specify one of the optional parameters of FIXCDS, the default is DISPLAY.

Note:

1. When a type X backup volume record is deleted from the BCDS, the BVR entry for the volume and the TTOC record for a tape backup volume are also deleted.
2. When deleting the base TTOC record (for example, sequence number 0000), the entire TTOC record for the volume is deleted (the base and all extensions). However, if an extension other than the base (0000) is specified, only that specified extension will be deleted.

   If you are deleting a MC1 or JES3 VAC continuation record, be sure to patch the continuation flag in the appropriate lower-level record to binary '0'.

   If you delete a lower-level MC1 or VAC record, all upper-level continuation records remain in the MCDS but are not usable by DFSMShsm.
3. For deletion of an L record, the user's data set name is specified (not the key). If there is only one L record that corresponds to the requested data set name, the L record is deleted. If there is more than one L record for the data set name, only those L records not associated with a D record are deleted. If there is more than one L record for the data set name and all L records need to be deleted, issue the same command twice.

DISPLAY: Requesting a display of the data from the control data sets

Explanation: DISPLAY is an optional parameter that allows you to display the data from the specified record. A record consists of a key field (44 bytes), followed

by a header field (20 bytes), followed by variable data depending on the record type. For each request, you always see the header field but never see the key field. If you specify DISPLAY without specifying (*offset*), the variable data portion of the record is displayed starting at offset zero.

DISPLAY(*offset*) allows you to specify a decimal value anywhere from zero up to a number equal to the length of your CDS record, or an equivalent hexadecimal value, for the offset. Hexadecimal characters must be in the form *X'n'*. The variable data portion of the record is displayed starting at the specified offset.

The LENGTH parameter controls how much record data you want to display.

DISPLAY GEN(*nn*) allows you to specify the original data set name, along with a generation number, instead of specifying the DFSMShsm-generated name for the backup version data set. When using the GEN(*nn*) parameter, the FIXCDS required parameter *type* must be 'C'.

Abbreviations: None.

Defaults: If you do not specify one of the optional parameters of FIXCDS, the default is DISPLAY.

Note:

1. If you specify an offset that goes beyond the end of the record to which you are referring, the FIXCDS command fails.
2. When you display the D record for a migrated VSAM data set, the password field is blanked out.
3. The offset of the first byte of the variable data of the record is zero. The offset of the last byte of the variable data of the record is one less than the length of the variable data.
4. If you are displaying an MCL record, the first MCL record encountered with the correct record key is displayed.

ENTRY: Specifying a particular entry in the BVR, DVL, or TTOC record

Explanation: ENTRY(*volser* | *dsname*) is an optional parameter used to specify a particular entry in the BVR, DVL, or TTOC record. Offset is then the offset in the specified entry. ENTRY may be used with the CREATE, DELETE, DISPLAY, VERIFY, and PATCH parameters. The domain of CREATE ENTRY is the first unused entry in the existing record.

Abbreviations: None.

Defaults: If you do not specify one of the optional parameters of FIXCDS, the default is DISPLAY.

EXPAND: Expanding a control data set record

Explanation: EXPAND(*bytes*) is an optional parameter specifying that a control data set record be expanded by the specified number of bytes. The new portion of the record is set to binary zeros. For *bytes*, substitute a decimal number for the number of bytes by which you want the record to be expanded.

Abbreviations: None.

Defaults: If you do not specify one of the optional parameters of FIXCDS, the default is DISPLAY.

Note:

1. If the expanded record length is more than 2040 bytes, the FIXCDS command fails.
2. If you expand the CDS record length beyond the maximum record length specified when the control data set was defined, your CDS may become unreadable.
3. If a backup cycle volume record (BVR) is being expanded, the specified length must be an even multiple of 12 bytes, the length of the volume entry in the BVR.
4. If a BVR is being expanded, the resultant record length must not be able to contain more than the maximum number of volume entries. If the maximum record length is 1016 bytes in length, 78 entries can be put in each BVR. If the maximum record length is 2040 bytes in length, 164 entries can be put in each BVR.
5. When a BVR is expanded, the number of volume entries is incremented to reflect the new number of volume entries. If you subsequently update these new volume entries with volume serial numbers, you must ensure that the volume full flag is turned off in the BVR by using the appropriate FIXCDS PATCH command.
6. When expanding a record that was created by a previous release of DFSMShsm, the results may not be as expected. The length of CDS records may change from release to release, and DFSMShsm enforces writing and rewriting records to the correct release. After issuing an EXPAND request, DISPLAY the record to ensure that the expected results have been achieved.

INDEX: Processing a MCBR record entry using the MCB record key

Explanation: INDEX allows the FIXCDS DISPLAY, VERIFY, and PATCH of an entry in a MCBR record without having to generate the MCBR key. Specify Z as the record type, and the original data set name (which is also the MCB key) as the key on the FIXCDS command. Indicate the MCBR record that is to be processed with the INDEX parameter. Specify index number 1 to process the most recent MCBR record, index 2 to process the second most recent MCBR record, and so on. Any value from 1 to 65535 can be specified, as there can be multiple MCBR records associated with a single data set and each MCBR record contains up to 100 entries.

Abbreviations: None.

Defaults: The default value of INDEX is 1.

Note: You cannot specify the INDEX keyword with the CREATE keyword. Specify the hex MCBR key on the FIXCDS command when using the CREATE keyword.

LENGTH: Specifying the length of the data or of a new control data set record

Explanation: LENGTH(*bytes*) is an optional parameter specifying the length of the data you want to display or the length of a new control data set record. LENGTH applies to only the variable portion of a record. For a display, the record header is always displayed with the number of bytes you have specified with this parameter. For a new reocrd, FIXCDS alays creates the first 64 bytes that are the key and header fields.

For (*bytes*), sybstitute a decimal number from 1 through 1976 for the number of bytes you want to display or create for the variable portion of the record.

Abbreviations: None.

Defaults: If you do not specify LENGTH or you specify LENGTH without *bytes*, the displayed default length is the remaining length of the current record, or the created default length is the minimum valid length for the variable data portion of the type of record you want to create.

Note:

1. The LENGTH parameter applies only with the DISPLAY and CREATE parameters. If you specify LENGTH when it does not apply, DFSMShsm ignores it.
2. If you specify LENGTH with CREATE, the number of bytes you specify must be at least the minimum length allowed for the variable data portion of the record you are creating.

LOGONLY: Sending the output to the output data set

Explanation: LOGONLY is an optional parameter specifying that you want the output to go only to the output data set.

If LOGONLY is specified when it is not applicable, it is ignored.

If LOGONLY is specified and OUTDATASET(*dsname*) is *not* specified, it implies that printed output for FIXCDS processing is not desired and will *not* be sent to the user and/or console.

Abbreviations: None.

Defaults: None.

NEWKEY: Renaming a control data set record

Explanation: NEWKEY(*keyname*) is an optional parameter specifying that a control data set record be renamed. For *keyname*, substitute 1 to 44 EBCDIC characters or specify the new key as hexadecimal. Do not include the 1-byte hexadecimal identification in the *keyname.* The 1-byte hexadecimal ID is the first byte of the internal format of keys. The types are E, F, G, H, I, J, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, and 1 records.

Abbreviations: None.

Defaults: If you do not specify one of the optional parameters of FIXCDS, the default is DISPLAY.

Note:

1. You must specify the same record type as the one you specified with the *type* parameter. The record with the new key is the same as the record with the old key.
2. When you specify the key as hexadecimal, specify an even number of hexadecimal digits. If you specify an odd number of hexadecimal digits, DFSMShsm inserts a zero to the left of the value to make it an even number. Hexadecimal characters must be in the form *X'n'*.
3. If the FIXCDS NEWKEY parameter is performed on a DFSMShsm record and the new data set name (DSN) that is specified by the NEWKEY parameter

differs from the original data set name, then any subsequent function which references the original data set name will fail.

4. If an MCL record is specified, the FIXCDS command will fail.

OUTDATASET: Specifying the output location for the FIXCDS results

Explanation: OUTDATASET(*dsname*) is an optional parameter specifying the name of the data set where DFSMShsm is to write the output data. For *dsname*, substitute the fully qualified name of the data set that receives the FIXCDS results. If the data set does not exist, DFSMShsm dynamically allocates and catalogs an output data set with the following characteristics:

- The data set name specified (dsname)
- Record format FBA
- Logical record length of 121
- Primary allocation of 20 tracks (see note)
- Secondary allocation of 50 tracks (see note)
- Unit type of SYSALLDA (see note)
- System reblockable

If the data set already exists:

- It must be cataloged and on DASD.
- Its record format must be FBA.
- Its logical record length must be 121.
- The data set must be system reblockable.
- The user can choose the primary space allocation. If DFSMShsm needs additional extents after the primary space allocation, DFSMShsm uses a secondary space allocation of 50 tracks (see note).
- If the data set does not contain data, DFSMShsm starts writing output data at the beginning of the data set.
- If the data set contains data, DFSMShsm writes the output data after the existing data.

Note: You can use the PATCH command to change the unit name, the primary allocation, and the secondary allocation. See the "Tuning DFSMShsm" topic in *z/OS DFSMShsm Implementation and Customization Guide*.

Abbreviations: You can use the abbreviation ODS for OUTDATASET.

Defaults: None.

PATCH: Changing a control data set record

Explanation: PATCH is an optional parameter you specify to change a control data set record.

PATCH(*offset*) allows you to specify a decimal value anywhere from zero up to a number equal to the length of your CDS record, or an equivalent hexadecimal value, for the offset. Hexadecimal characters must be in the form *X'n'*. The variable data portion of the record is patched starting at the specified offset.

PATCH(*data*) allows you to substitute the changes you want made to the data. You can enter the data as hexadecimal characters or alphameric characters and $, #, or @. You can change up to 256 bytes of data.

PATCH(*bits*), allows you to substitute an eight-character string specifying which bit or bits you want turned on or off. A period (.) means do not change bit, a **1** means set bit on, and a **0** means set bit off. For example,

```
0.1.....
```

means set bit 0 off and set bit 2 on and do not change bits 1, 3, 4, 5, 6, and 7.

PATCH GEN(*nn*) allows you to specify the original data set name, along with a generation number, instead of specifying the DFSMShsm-generated name for the backup version data set. When using the GEN(*nn*) parameter, the FIXCDS required parameter *type* must be 'C'.

Abbreviations: None.

Defaults: If you do not specify one of the optional parameters of FIXCDS, the default is DISPLAY.

Note:

1. The offset of the first byte of the variable data of the record is zero.
2. The offset of the last byte of the variable data of the record is one less than the length of the variable data.
3. If you are patching an MCL record, the first two bytes of the record cannot be changed. This area contains the 43rd and 44th character of the data set name. If this area is specified to be patched, the FIXCDS command fails.
4. The results of using the FIXCDS command to patch the CDS records may not be permanent. There are situations when DFSMShsm logic may negate the patched field.

REFRESH: Requesting that the BVRs be recreated and reorganized

Explanation: REFRESH(ON | OFF) is an optional parameter that manipulates DFSMShsm flags to indicate that the BVRs need or do not need to be recreated and reorganized. An ON request indicates that the BVRs should be refreshed during the next DFSMShsm CDS backup, an OFF indicates the BVRs should not be refreshed during the next CDS backup. The REFRESH is done during the next successful DFSMShsm CDS backup.

Abbreviations: None.

Defaults: If you do not specify one of the optional parameters of FIXCDS, the default is DISPLAY. If you do not specify one of the optional REFRESH parameters (ON or OFF), the command fails. If you just specify FIXCDS, the command also fails.

Note:

1. The REFRESH parameter is only valid if you specify the record type as R, and the three letters of the record key are BVR. Do not use BVRSP, BVRUN, or BVR*nn* where *nn* is a backup cycle day.
2. After requesting a REFRESH of the BVRs, the BVRs are recreated and reorganized when DFSMShsm next performs CDS backup and when the CDS backup on this HSMPlex is successful.
3. The REFRESH(OFF) command only turns off a REFRESH request if a REFRESH(ON) command was previously issued but has not yet occurred because the CDS backup has not successfully completed.

VERIFY: Verify that a field in a CDS record matches the given data

Explanation: VERIFY (*offset*[*data* | BITS(*bits*)]) | is an optional parameter used to confirm that the data in the control data set record matches the specified data. If VERIFY is specified, the requested function can only be performed if the VERIFY is successful. VERIFY should always be used with the PATCH parameter: If the verify fails, the patch is not performed.

For *offset*, specify a decimal value anywhere from zero up to a number equal to the length of your CDS record, or an equivalent hexadecimal value, that represents the variable data portion of the record you want to check. If the ENTRY parameter is used, *offset* applies to the particular entry in the record and not to the record itself. Hexadecimal characters must be in the form *X'n'*.

For *data*, substitute the data you want checked. You can enter the data as hexadecimal characters or alphameric characters and $, #, or @. You can verify up to 256 bytes of data.

For *bits*, substitute an eight-character string specifying which bit or bits you want to test for on/off. A period (.) means do not test, a **1** means to verify if the bit is on, and a **0** means to verify if the bit is off. For example,

```
1.0.....
```

means verify if bit 0 is on and bit 2 is off.

Abbreviations: None.

Defaults: If you do not specify one of the optional parameters of FIXCDS, the default is DISPLAY.

Note:

1. VERIFY is not required when you specify the PATCH parameter, but its use is recommended.
2. VERIFY can be used alone, without the PATCH parameter.

Examples of how to code the FIXCDS command

The following table lists examples of different ways to code the FIXCDS command.

Note: Any values specified here are examples only and should not be interpreted as the values to be used for your system.

Displaying a type Z data set record for retained backup copies

Example: This example displays the second most recent type Z data set record using the original data set name and the INDEX keyword.

```
FIXCDS Z ABC787.REPORT.DATA DISPLAY INDEX(2)
```

Displaying a backup control record from the BCDS

Example: In this example, the type R backup control record from the issuing processing unit is displayed from variable data offset 15 to the end of the record. The output goes only to the SYSOUT data set:

```
FIXCDS R BCR DISPLAY(15) LOGONLY
```

Adding a migrated data set record to the MCDS

Example: In this example, a type D data set record for a data set that has migrated is added to the MCDS:

```
FIXCDS D PAC1234.DUMMY.ASM ADDMIGRATEDDATASET(VOL006)
```

Deleting a backup volume record from the BCDS

Example: This example deletes a type X backup volume record from the BCDS:

```
FIXCDS X BAK123 DELETE
```

Expanding a backup data set record in the BCDS

Example: This example expands a type B backup data set record in the BCDS by 256 bytes:

```
FIXCDS B JLT7652.REPORT.DATA EXPAND(256)
```

Creating a management control record in the MCDS

Example: This example creates the type S management control record for processing unit 1, having variable data 512 bytes in length, with data supplied for offset locations 64 and 65 in the variable data portion of the new record:

```
FIXCDS S MCR1 CREATE(64 X'8010') LENGTH(512)
```

Turning on the assigned bit in the MCD record

Example: This example turns on the assigned bit in the MCD record to indicate that the data set has migrated:

```
FIXCDS D VLS325.REPORT.ASM ASSIGNEDBIT(ON)
```

Renaming a migration volume record in the MCDS

Example: This example renames a type V migration volume record to a new key:

```
FIXCDS V MIG011 NEWKEY(SCR002)
```

Modifying a volume statistics record in the MCDS

Example: This example modifies a type S volume statistics record at offset locations 64 and 65 of the variable data, and prints the SYSOUT data set:

```
FIXCDS S X'E5E2D9C2C1D2E5D6D377133F' PATCH(64 X'0700')
```

Deleting a tape volume record from the OCDS

Example: This example deletes all type T tape volume records from the OCDS that are associated with daily backup volume DBV456:

```
FIXCDS T 01-DBV456-0000 DELETE
```

Example: This example deletes all type T tape volume records from the OCDS that are associated with ML2 volume ML2456:

```
FIXCDS T L2-ML2456-0000 DELETE
```

Changing the version number of the multiple backup data set

Example: This example backs up the control data sets to multiple backup data sets that have reached the limit of V9999999. You change the last final qualifier to V0000001:

```
FIXCDS S MHCR PATCH(X'B0' X'E5F0F0F0F0F0F0F1')
```

Displaying part of the type R dump control record

Example: This example displays part of the type R dump control record for a processing unit. The dump cycle (bit string) is at offset 12. The length of the dump cycle is at offset 16. The cycle start date (Julian date in packed decimal format) is at offset 18:

```
FIXCDS R DCR DISPLAY(12) LENGTH(12)
```

Displaying the array of dump generation keys in the type P BCDS volume record

Example: This example displays the array of dump generation keys in the type P BCDS volume record for volume PRIM01. The array begins at offset X'138'. Each key is 14 bytes long. There are a maximum of 100 dump generations:

```
FIXCDS P PRIM01 DISPLAY(X'138') LENGTH(1400)
```

Displaying a type G dump generation record

Example: This example displays the dump generation record for a dump that was done of user volume PRIM01 (EBCDIC ‘D7D9C9D4F0F1’) at midnight on Julian date 86209:

```
FIXCDS G X'D7D9C9D4F0F1000000000086209F' DISPLAY
```

Displaying a type Y dump volume record

Example: This example displays the type Y dump volume record for volume D00001:

```
FIXCDS Y D00001 DISPLAY
```

Displaying a type W dump class record

Example: This example displays the type W dump class record for class WEEKLY:

```
FIXCDS W WEEKLY DISPLAY
```

Verifying and patching a migrated data set record

Example: This example changes a type D data set record RECORD_KEY at offset X'10'. The byte (8 bits) of data is changed *only* if initially bit 7 is off, bits 2 and 3 are on, and bit 0 is off. The data before and after the PATCH is added to the data set USERID.FIXCDS.DATA. The PATCH data sends the output to the terminal that issued the command:

```
FIXCDS D RECORD_KEY PATCH(X'10' BITS(1.00...1))
       VERIFY(X'10' BITS(0.11...0)) ODS(USERID.FIXCDS.DATA)
```

Example: This example is the same as the preceding example, except that the PATCH command does *not* send data output to the terminal that issued the command:

```
FIXCDS D RECORD_KEY PATCH(X'10' BITS(1.00...1))
       VERIFY(X¬10' BITS(0.11...0)) ODS(USERID.FIXCDS.DATA)
       LOGONLY
```

Example: This example is the same as the preceding example, except that the data before and after the PATCH is *not* sent to any output data set:

```
FIXCDS D RECORD_KEY PATCH(X'10' BITS(1.00...1))
       VERIFY(X¬10' BITS(0.11...0)) LOGONLY
```

Creating an entry for a backup volume record

Example: This example creates an ENTRY for volume BACK01 in the type R backup volume record BVR01-0000 if the entry does not already exist:

```
FIXCDS R BVR01-0000 CREATE ENTRY(BACK01)
```

Example: This example is like the previous example. Additionally, it changes the ENTRY area beginning at X'6' with the data specified by xxxxxxxx:

```
FIXCDS R BVR01-0000 CREATE(X'06' X'xxxxxxxx') ENTRY(BACK01)
```

Patching character data

Example: This example changes character data so that the password in the MCB record is changed to NEWPASWD:

```
FIXCDS B ABC656.REPORT.DATA PATCH(X'80' NEWPASWD)
```

Displaying a data set using the original data set name with a generation number

Example: This example displays a type C backup version record using the original data set name and a generation number:

```
FIXCDS C JLT7652.REPORT.DATA DISPLAY GEN(0)
```

Refreshing the BVR records

Example: This example requests a refresh of all of the BVRs on your HSMplex during the next CDS backup via the BACKVOL CDS command or at the beginning of the auto backup window:

```
FIXCDS R BVR REFRESH(ON)
```

Canceling the refresh of the BVR records

Example: This example cancels the request for a refresh of all of the BVRs on your HSMplex during the next CDS backup window. This command is only used to override a previous REFRESH(ON) command that has not yet been processed because the CDS backup has not yet successfully completed:

```
FIXCDS R BVR REFRESH(OFF)
```

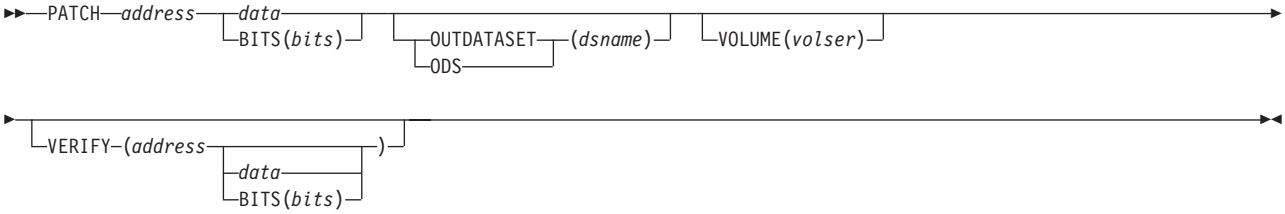
PATCH: Changing storage in the address space of DFSMShsm

The PATCH command changes storage within the DFSMShsm address space under the protect key of the DFSMShsm problem program. You can identify the storage location to be changed with an absolute address or a qualified address.

The PATCH command has a VERIFY parameter that you can specify to be sure that you have correctly identified the location where you want to make a change.

Note: For information on DFSMShsm supported patches, refer to the *z/OS DFSMShsm Implementation and Customization Guide*.

PATCH command syntax



Required parameters of the PATCH command

The following sections provide information about the required parameters of the PATCH command.

Address: Specifying the location to change

Explanation: *address* is a required positional parameter specifying where you want the DFSMShsm address space changed.

For *address*, substitute the absolute address or the qualified address of the location you want to change. Although the addresses contain hexadecimal digits, you do not represent them in the form *X'nn'*. Instead, enter them in the following manner:

- Enter an absolute address as one to eight hexadecimal digits followed by a period (*hhhhhhhh.*).
- Enter a qualified address as a DFSMShsm load module name, followed by a period, a CSECT name, a period, a plus sign, and one to six hexadecimal digits (*loadmodulename.csectname.+hhhhhh*). Because ARCCTL is the only load module that can be displayed, the load name of ARCCTL is always used as the load module name if you supply only the CSECT name and offset (*.csectname.+hhhhhh*).

  Note: The leading period is still required.

The PATCH command considers the .MVT. control block identifier to be a valid control section (CSECT) name.

Abbreviations: None.

Defaults: None.

Note:

1. Because *address* is a required positional parameter, you must specify it immediately after the command name.

2. You cannot use indirect addressing, registers, expressions, and variable names with the PATCH command.
3. You cannot patch addresses in a DFSMShsm secondary address space.

Data: Specifying the data

Explanation: *data* is a required positional parameter specifying the data you want to enter at the specified address.

For *data*, substitute hexadecimal data in the form *X'data'* or substitute character data. If the data contains special characters, put single quotation marks around it. You cannot specify more than 256 bytes (512 hexadecimal digits), and you should specify an even number of hexadecimal digits. If you specify an odd number of hexadecimal digits, DFSMShsm inserts a zero to the left of the value to make it an even number of hexadecimal digits.

Abbreviations: None.

Defaults: None.

Note: Because *data* is a required positional parameter, you must specify it immediately after *address*.

BITS: Specifying the bits to change

For *bits*, substitute an eight-character string specifying which bit or bits are to be turned on or off. A period (**.**) means no change, a **1** means set the bit on, and a **0** means set the bit off. For example,

```
0.1.....
```

means set bit 0 off, set bit 2 on, and leave all other bits unchanged.

Abbreviations: None.

Defaults: None.

Optional parameters of the PATCH command

OUTDATASET: Specifying the output location for the PATCH results

Explanation: OUTDATASET(*dsname*) is an optional parameter specifying the name of the data set where DFSMShsm is to write the output data.

For *dsname*, substitute the fully qualified name of the data set that is to receive the PATCH results. If the data set does not exist, DFSMShsm dynamically allocates and catalogs an output data set with the following characteristics:

- The data set name specified (dsname)
- Record format of FBA
- Logical record length of 121
- Primary allocation of 20 tracks (see note)
- Secondary allocation of 50 tracks (see note)
- Unit type of SYSALLDA (see note)
- System reblockable

If the data set already exists:

- It must be cataloged and on DASD.
- The record format must be FBA.
- The logical record length must be 121.
- The data set must be system reblockable.
- The user can choose the primary space allocation.
- If DFSMShsm needs additional extents after the primary space allocation, DFSMShsm uses a secondary space allocation of 50 tracks (see note).
- If the data set does not contain data, DFSMShsm starts writing output data at the beginning of the data set.
- If the data set contains data, DFSMShsm writes the output data after the existing data.

Note: You can use the PATCH command to change the unit name, the primary allocation, and the secondary allocation. See the "Tuning DFSMShsm" topic in *z/OS DFSMShsm Implementation and Customization Guide*.

Abbreviations: You can use the abbreviation ODS for OUTDATASET.

Defaults: None.

VOLUME: Changing a mounted volume table with a specific volume serial number

Explanation: VOLUME(*volser*) is used to request the modification of a specific mounted volume table (MVT) that contains a matching volume serial number for a DFSMShsm-managed level 0 volume or migration level 1 volume. The .MVT. parameter is required when the control block is being modified.

Abbreviations: None.

Defaults: None.

Note: The VOLUME parameter can be specified only for the MVT control block.

VERIFY: Verifying data before changing It

Explanation: VERIFY(*address* [*data* | BITS(*bits*)]) is an optional parameter requesting that DFSMShsm verify the current data before it makes the change. If the data does not match, DFSMShsm rejects the change.

For *address*, substitute the absolute address or the qualified address, of the location you want to change. Although the addresses contain hexadecimal digits, you do not represent them in the form *X'nn'*. Instead, you enter them in the following manner:

- Enter an absolute address as one to eight hexadecimal digits followed by a period (*hhhhhhhh.*).
- Enter a qualified address as a DFSMShsm load module name, followed by a period, a CSECT name, a period, a plus sign, and one to six hexadecimal digits (*loadmodulename.csectname.+hhhhhh*). Because ARCCTL is the only load module that can be displayed, the load name of ARCCTL is always used as the load module name if you supply only the CSECT name and offset (*.csectname.+hhhhhh*). The leading period is still required.

The PATCH command considers the .MVT. control block identifier to be a valid control section (CSECT) name.

For *data*, specify the data that you want DFSMShsm to verify. You can verify up to 256 bytes of data.

For *bits*, substitute an eight-character string specifying which bit or bits are to be turned on or off. A period (**.**) means do not test, a **1** means test bit for on, and a **0** means test bit for off. For example,

```
1.0.....
```

means test bit 0 for on, test bit 2 for off, and do not test the remaining bits.

Abbreviations: None.

Defaults: None.

Note: You cannot use indirect addressing, registers, and variable names with the PATCH command.

Examples of how to code the PATCH command

The following table lists examples of different ways to code the PATCH command.

Note: Any values specified here are examples only and should not be interpreted as the values to be used for your system.

Changing the data at an absolute address and verifying the change

Example: This example modifies the data at an absolute address and verifies it before processing:

```
PATCH 03B7A341. 'ABCD&' VERIFY(03B7A341. X'0000000000')
```

Changing the data at a qualified address

Example: This example modifies the data at a qualified address without verification:

```
PATCH .ARCCPFC.+4FA X'0700'
```

Changing the date fields

Example: This example changes the last date the following functions have run. This permits these functions to be run again on the current day:

- Automatic backup
- Moving backup versions and backing up migrated data sets (level-1 functions)

To cause the functions to be run again, you can zero the appropriate date field and issue the SETSYS command with the AUTOBACKUPSTART parameters, specifying the appropriate times:

```
PATCH .BCR.+50 X'00000000' /* automatic backup          */
PATCH .BCR.+5C X'00000000' /* move backup versions and   */
                           /* back up migrated data/sets */
```

Allowing migration of password-protected generation data sets

Example: This example sets the bit MCVTFPW in the MCVT to 1 to select a design alternative that allows password-protected generation data sets to be migrated and allows DFSMShsm to ignore the password at the time a generation is rolled off:

```
PATCH .MCVT.+53 BITS(.1......) /* allow migrate of password-protected dataset*/
```

Scratching generation data sets regardless of expiration date at roll-off time

Example: This example changes the bit MCVTFGDG in the MCVT to 1 to select a design alternative that allows DFSMShsm to scratch a generation data set regardless of its expiration date when it is rolled off:

```
PATCH .MCVT.+53 BITS(1.......) /* allow scratch of date-protected data set*/
                               /* at time of roll-off                    */
```

Patching a specific MVT entry

Example: This example modifies the MVT entry corresponding to a volser of PRIM01 without verification. The hex value '000001F4' is placed at offset X'24' into the MVT entry:

```
PATCH .MVT.+24 X'000001F4' VOLUME(PRIM01)
```

Varying dump invocation

Example: The following command sets the bit MCVTALLX in the MCVT to 1 to suppress the ALLEXCP option specified in the full-volume dump invocation of DFSMSdss by DFSMShsm:

```
PATCH .MCVT.+3C3 BITS(1.......) /* suppress ALLEXCP option */
```

Example: This example sets the bit MCVTALLD in the MCVT to 1 to suppress the ALLDATA(*) option specified in the full-volume dump invocation of DFSMSdss by DFSMShsm:

```
PATCH .MCVT.+3C3 BITS(.1.....) /* suppress ALLDATA(*) option */
```

Example: This example sets the bits MCVTALLX and MCVTALLD in the MCVT to 1 to suppress both the ALLEXCP and ALLDATA(*) options:

```
PATCH .MCVT.+3C3 BITS(11......) /* suppress ALLEXCP and ALLDATA(*) option */
```

Verifying and patching data and sending the results to an output data set

Example: This example's command will patch the named module at address +10. The data is changed *only* if the 4 bytes of data at address .MODULE_NAME.+10 currently has a value of X'87654321'. The data before and after the PATCH is added to the data set **USERID.PATCH.DATA** and is also shown at the terminal of the issuer of the PATCH command:

```
PATCH .MODULE_NAME.+10 X'12345678'
      VERIFY(.MODULE_NAME.+10 X'87654321') ODS(USERID.PATCH.DATA)
```

Example: This example uses the PATCH command to change the named module at address +10. The data is changed ONLY if the byte of data at address +10 currently has the following value:

bit 0 = 0

bits 2,3 = 1
bit 7 = 0
bits 1,4,5,6 = not tested

The data before and after the patch is added to the data set **USERID.PATCH.DATA** and is also shown at the terminal of the issuer of the PATCH command:

```
PATCH .MODULE_NAME.+10 BITS(1.00...1)
      VERIFY(.MODULE_NAME.+10 BITS(0.11...0)) ODS(USERID.PATCH.DATA)
```

TRAP: Requesting a dump when a specified error occurs

DFSMShsm takes predetermined actions when it detects certain error conditions. You use the TRAP command to alter DFSMShsm's actions based on your installation's needs. You cannot use TRAP to alter any error conditions for which DFSMShsm issues an abend or a shutdown. Instead, as the issuer of the TRAP command, you may request that DFSMShsm take one of the following actions in response to the specified error condition:

- Abnormally end the task when the error occurs. After the abnormal end, DFSMShsm produces an abnormal end dump (ABEND).
- Write the error in the command activity log (LOG).
- Remove a previous trap (OFF).
- Perform a snap dump (SNAP).

DFSMShsm automatically shuts down because of recurring trap conditions. For example, DFSMShsm automatically shuts down after 200 occurrences of the same error code in the same module under one of the following conditions:

- Automatic traps: 200 occurrences of the same error code in the same module with an option of ABEND or SNAP.
- TRAP command issued with the SNAP option: 200 occurrences of the same error code in the same module with an option of ABEND or SNAP.
- TRAP command issued with the ABEND option: 200 occurrences of the same error code in the same module with any option except FATAL.

By default, the number of occurrences required before DFSMShsm terminates is 25 for ABENDs, and 200 for SNAPs. These defaults can be changed by patching the MCVT.

Table 6 shows which ARCERP options can be altered by a TRAP option. In general:

- You can alter only the ARCERP options of LOG, DEBUG, and SNAP by a TRAP option.
- You cannot alter the ARCERP options of FATAL, ABEND, and ABEND/NODUMP by a TRAP option.

Table 6. Trap Options

ARCERP Options	TRAP Command Options		
	ABEND	LOG	SNAP
LOG	ABEND/DUMP	LOG	SNAP
DEBUG	ABEND/DUMP	LOG	SNAP
SNAP	ABEND/DUMP	LOG	SNAP

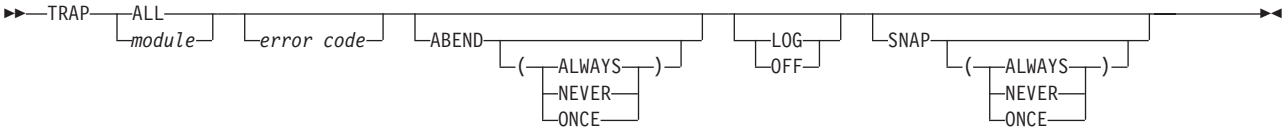
Table 6. Trap Options (continued)

ARCERP Options	TRAP Command Options		
	ABEND	LOG	SNAP
ABEND	ABEND/DUMP	ABEND/DUMP	ABEND/DUMP
ABEND/NODUMP	ABEND/DUMP	ABEND/DUMP	ABEND/DUMP
FATAL	ABEND/DUMP Shut down DFSMShsm	ABEND/DUMP Shut down DFSMShsm	ABEND/DUMP Shut down DFSMShsm

See Chapter 8, "Diagnosing from return codes and reason codes," on page 37 for more information on ARCERP.

TRAP command syntax

TRAP command syntax



Required parameters of the TRAP command

The following sections provide information about the required parameters of the TRAP command.

ALL and *module*: Specifying the location of the trap

Explanation: ALL and *module* are mutually exclusive, required positional parameters that specify the location of the trap.

ALL specifies that you want to test all DFSMShsm modules for errors.

For *module*, substitute the name of the DFSMShsm module that you want to test for errors. A list of modules and codes that DFSMShsm can trap are described in Chapter 8, "Diagnosing from return codes and reason codes," on page 37.

Abbreviations: None.

Defaults: None.

Note: Because *module* is a required positional parameter, you must specify it immediately after the command name.

Optional parameters of the TRAP command

The following sections provide information about the optional parameters of the TRAP command.

error code: Specifying the error code to test for

Explanation: *error code* is an optional positional parameter specifying the error return code to be tested when it is returned by the module you specify. For example, if you specify TRAP ARCGODS 13, DFSMShsm checks for return code 13 from module ARCGODS. For *error code*, substitute a code number. If you substitute a zero for the error code, DFSMShsm tests for any error in the specified module.

Abbreviations: None.

Defaults: When you do not substitute an error code, *error code* defaults to 0. This means that DFSMShsm tests for any error in the module you specify.

Note: Because *error code* is an optional positional parameter, you must specify it immediately after *module.*

ABEND, LOG, OFF, and SNAP: Specifying What DFSMShsm should do if an error occurs

Explanation: ABEND, LOG, OFF, and SNAP are mutually exclusive, optional parameters specifying what DFSMShsm should do and how many times it should do it when the specified error occurs in the specified module.

ABEND specifies that DFSMShsm abnormally ends the task when the specified error condition occurs in the specified module. DFSMShsm abnormally ends the task by issuing an ABEND macro. DFSMShsm almost always tries to restart the task that abnormally ended. During the abnormal end, DFSMShsm produces an abnormal end dump.

ALWAYS, NEVER, and ONCE are mutually exclusive, optional subparameters of the ABEND parameter, specifying how many times DFSMShsm abnormally ends the task when the specified error condition occurs in the specified module.

- ALWAYS specifies that DFSMShsm abnormally end the task every time the specified error condition occurs in the specified module.
- NEVER specifies that DFSMShsm should never abnormally end the task when the specified error condition occurs in the specified module.
- ONCE specifies that DFSMShsm should abnormally end the task the first time the specified error condition occurs in the specified module.

LOG specifies that you want DFSMShsm to write an entry in the DFSMShsm log when the specified error condition occurs in the specified module. TRAP LOG avoids DFSMShsm shutdown because the errors do not increment the counter for the maximum number of occurrences. That is, TRAP LOG logs the errors but does not count them.

OFF specifies that you want DFSMShsm to remove a trap that you specified with a previous TRAP command. For example, you specified TRAP ARCGODS 13 SNAP(ONCE), causing DFSMShsm to produce a snap dump the first time error condition 13 occurs in module ARCGODS. To prevent DFSMShsm from further checking for that error condition, you specify TRAP ARCGODS 13 OFF. This command returns DFSMShsm to the process level that IBM previously specified for just SNAP. See ARCGODS in Chapter 8, “Diagnosing from return codes and reason codes,” on page 37.

SNAP specifies that you want DFSMShsm to produce a snap dump when the specified error condition occurs in the specified module. DFSMShsm produces a snap dump by issuing a SNAP macro. If you specify the SYS1DUMP parameter of the SETSYS command, the dump is written to a system dump data set. When a TRAP occurs, DFSMShsm produces a snap dump directed to the SYSOUT class specified in the SETSYS SYSOUT command. DFSMShsm continues processing after producing a snap dump.

ALWAYS | NEVER | ONCE are mutually exclusive, optional subparameters of the SNAP parameter, specifying how many times DFSMShsm produces a snap dump when the specified error condition occurs in the specified module.

- ALWAYS specifies that DFSMShsm produce a snap dump every time the specified error condition occurs in the specified module.
- NEVER specifies that DFSMShsm never produce a snap dump when the specified error condition occurs in the specified module.
- ONCE specifies that DFSMShsm produce a snap dump the first time the specified error condition occurs in the specified module.

Abbreviations: None.

Defaults: If you do not specify ABEND, LOG, OFF, or SNAP, the default is SNAP. If you do not specify ALWAYS, NEVER, or ONCE, the default is ONCE.

Examples of how to code the TRAP command

The following table lists examples of different ways to code the TRAP command. Any values specified here are examples only and should not be interpreted as the values to be used for your system.

Trapping an error condition with a snap dump

Example: This example traps all the error conditions in module ARCACREL, and DFSMShsm produces a snap dump the first time the error occurs:

```
TRAP ARCACREL 0 SNAP
```

Trapping an error condition with an abnormal end dump

Example: This example sets a trap for error condition 1 in module ARCCPOP, and DFSMShsm prints an abnormal end dump the first time the error occurs:

```
TRAP ARCCPOP 1 ABEND
```

Trapping an error condition with a log entry

Example: This example enters all the error conditions in module ARCBELIG in the DFSMShsm log:

```
TRAP ARCBELIG 0 LOG
```

Removing a previous trap

Example: This example removes a trap that was previously specified for error condition 3 in module ARCPROPN:

```
TRAP ARCPROPN 3 OFF
```

Chapter 11. Introduction to data areas and control blocks

For descriptions of the DFSMShsm control data set records, see *z/OS DFSMShsm Data Areas*, which is available online at the z/OS Internet Library at http://www.ibm.com/systems/z/os/zos/bkserv/.

Appendix. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html) 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

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS V2R2 ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that the screen reader is set to read out

punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the

default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

*** indicates an optional syntax element that is repeatable**

The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3* , 3 HOST, 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write HOST STATE, but you cannot write HOST HOST.
3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

+ indicates a syntax element that must be included

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

This information was developed for products and services offered in the U.S.A. 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 give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A

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.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites 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:

Site Counsel
IBM Corporation
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 information 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.

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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

COPYRIGHT LICENSE:

This information might contain 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.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, 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 (http://www.ibm.com/software/support/systemsz/lifecycle/)
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming interface information

This publication documents information **not** intended to be used as programming interfaces of DFSMShsm.

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 at Copyright and Trademark information (http://www.ibm.com/legal/copytrade.shtml).

Index

F

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

Product Number: 5650-ZOS

Printed in USA

GC52-1387-01