VMS

VMS Analyze/RMS_File Utility Manual

Order Number AA–LA79A–TE

VMS Analyze/RMS_File Utility Manual

Order Number: AA–LA79A–TE

April 1988

This manual describes the VMS Analyze/RMS_File Utility.

Revision/Update Information:	This manual supersedes the *VAX/VMS Analyze/RMS_File Utility Reference Manual,* Version 4.4.
Software Version:	VMS Version 5.0

digital equipment corporation
maynard, massachusetts

April 1988

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*	**CANADA**	**INTERNATIONAL**
Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061	Digital Equipment of Canada Ltd. 100 Herzberg Road Kanata, Ontario K2K 2A6 Attn: Direct Order Desk	Digital Equipment Corporation PSG Business Manager c/o Digital§ local subsidiary or approved distributor

In Continental USA and Puerto Rico call 800-258-1710.
In New Hampshire, Alaska, and Hawaii call 603-884-6660.
In Canada call 800-267-6215.
*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).
Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

Production Note

This book was produced with the VAX DOCUMENT electronic publishing system, a software tool developed and sold by DIGITAL. In this system, writers use an ASCII text editor to create source files containing text and English-like code; this code labels the structural elements of the document, such as chapters, paragraphs, and tables. The VAX DOCUMENT software, which runs on the VMS operating system, interprets the code to format the text, generate a table of contents and index, and paginate the entire document. Writers can print the document on the terminal or line printer, or they can use DIGITAL-supported devices, such as the LN03 laser printer and PostScript™ printers (PrintServer 40 or LN03R ScriptPrinter), to produce a typeset-quality copy containing integrated graphics.

™ PostScript is a trademark of Adobe Systems, Inc.

Contents

Preface

Intended Audience

This manual is intended for all programmers who use VMS RMS data files, including high-level language programmers who use only their language's input/output statements.

Document Structure

This document consists of the following five sections:

- Description—Provides a full description of the Analyze/RMS_File Utility (ANALYZE/RMS_FILE).
- Usage Summary—Outlines the following ANALYZE/RMS_FILE information:

  -Invoking the utility
  -Exiting from the utility
  -Directing output
  -Restrictions or privileges required
- Qualifiers—Describes ANALYZE/RMS_FILE qualifiers, including format, parameters, and examples.
- Commands—Describes ANALYZE/RMS_FILE commands, including format, parameters, and examples.
- Examples—Provides additional ANALYZE/RMS_FILE examples.

Associated Documents

To use the Analyze/RMS_File Utility, you should be familiar with the following manuals:

- *Guide to VMS File Applications*
- *VMS Convert and Convert/Reclaim Utility Manual*
- *VMS File Definition Language Facility Manual*
- *VAX RMS Journaling Manual*

Conventions

Convention	Meaning
$\boxed{\text{RET}}$	In examples, a key name (usually abbreviated) shown within a box indicates that you press a key on the keyboard; in text, a key name is not enclosed in a box. In this example, the key is the RETURN key. (Note that the RETURN key is not usually shown in syntax statements or in all examples; however, assume that you must press the RETURN key after entering a command or responding to a prompt.)
CTRL/C	A key combination, shown in uppercase with a slash separating two key names, indicates that you hold down the first key while you press the second key. For example, the key combination CTRL/C indicates that you hold down the key labeled CTRL while you press the key labeled C. In examples, a key combination is enclosed in a box.

`$ SHOW TIME`
`05-JUN-1988 11:55:22`
	In examples, system output (what the system displays) is shown in black. User input (what you enter) is shown in red.

`$ TYPE MYFILE.DAT` . . .	In examples, a vertical series of periods, or ellipsis, means either that not all the data that the system would display in response to a command is shown or that not all the data a user would enter is shown.
input-file, . . .	In examples, a horizontal ellipsis indicates that additional parameters, values, or other information can be entered, that preceding items can be repeated one or more times, or that optional arguments in a statement have been omitted.
[logical-name]	Brackets indicate that the enclosed item is optional. (Brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
quotation marks apostrophes	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe (') is used to refer to a single quotation mark.

New and Changed Features

The Analyze/RMS_File Utility (ANALYZE/RMS_FILE) has been enhanced for VMS Version 5.0 to provide support for analyzing VAX RMS Journaling files. This document provides essential details about this support, but refer to the *VAX RMS Journaling Manual* for a complete description.

ANALYZE/RMS_FILE Description

The Analyze/RMS_File Utility (ANALYZE/RMS_FILE) allows you to examine, either with or without an interactive terminal dialogue, the internal structure of a VMS Record Management Services (RMS) file. ANALYZE/RMS_FILE can check the structure of the file for errors, generate a statistical report on the structure and use of the file, or generate a File Definition Language (FDL) file from a data file.

The ANALYZE/RMS_FILE command provides information about VAX RMS Journaling files, including information about files marked for after-image journaling and before-image journaling. Where applicable, the ANALYZE/RMS_FILE command also provides information about the state of recovery units affecting VAX RMS Journaling files.

ANALYZE/RMS_FILE provides a set of commands that you may use to analyze a file interactively.

FDL files created with ANALYZE/RMS_FILE have special sections that contain statistics about the structure of the specified data file. You can use FDL files created with ANALYZE/RMS_FILE in conjunction with other VMS RMS utilities.

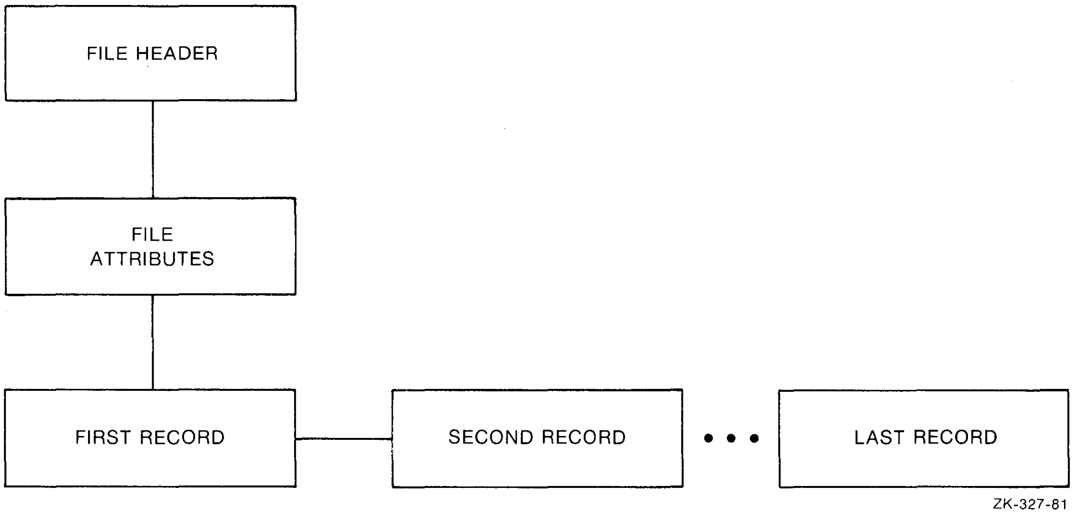
1 Analyzing VMS RMS File Structure Interactively

One of the most useful features of the Analyze/RMS_File Utility is its interactive mode. You enter the interactive mode by specifying the /INTERACTIVE qualifier to the ANALYZE/RMS_FILE command; you then begin an interactive session during which you can examine the structure of a VMS RMS file.

ANALYZE/RMS_FILE treats the internal VMS RMS file as a multilevel entity. All VMS RMS files are identical, relative to the first two levels. Level 1 contains the file header and level 2 contains the file attributes. For files marked for RMS Journaling, level 2 includes information relative to before-image journaling and after-image journaling, where applicable.

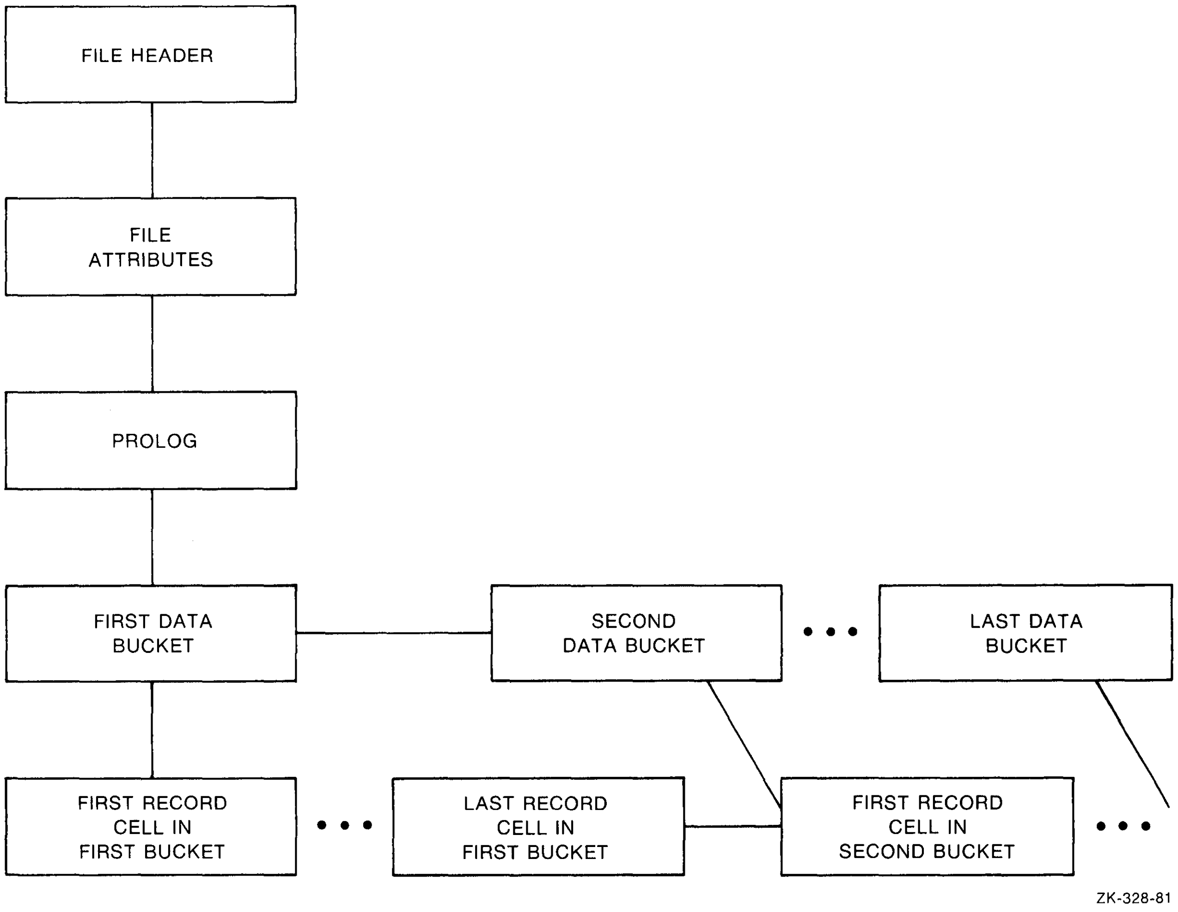
Some differentiation occurs at level 3. For sequential files, level 3 is the lowest level and it contains data records (see Figure ARMS–1) that ANALYZE/RMS_FILE can display individually. For relative files and indexed files, level 3 contains the file prolog.

Figure ARMS–1 Structure of Sequential Files



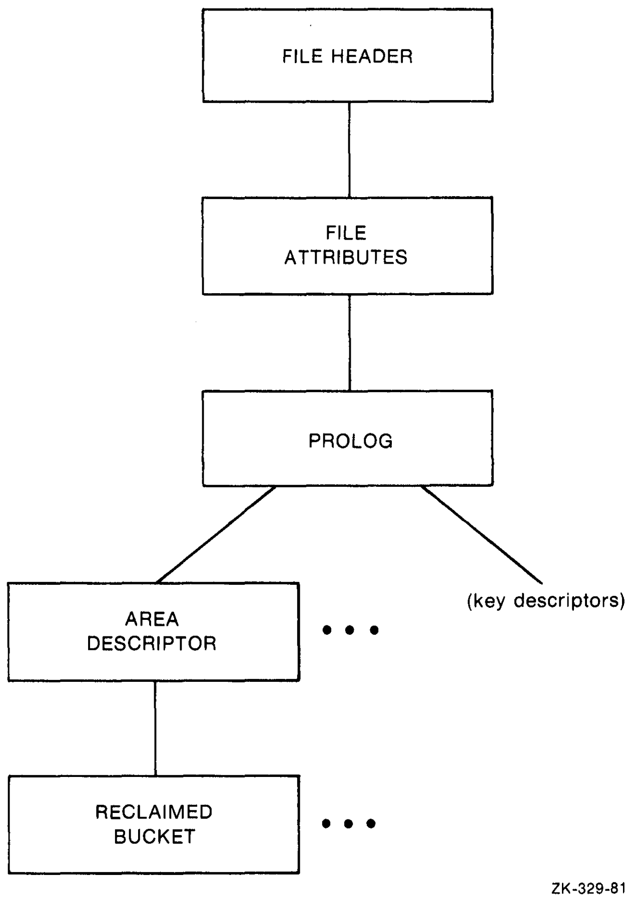
For relative files, level 4 contains data buckets that are accessible individually (see Figure ARMS–2). Using ANALYZE/RMS_FILE, you can view the contents of each individual data bucket.

Figure ARMS–2 Structure of Relative Files


FILE HEADER
FILE ATTRIBUTES
PROLOG
FIRST DATA BUCKET
SECOND DATA BUCKET
LAST DATA BUCKET
FIRST RECORD CELL IN FIRST BUCKET
LAST RECORD CELL IN FIRST BUCKET
FIRST RECORD CELL IN SECOND BUCKET
ZK-328-81

For indexed files, ANALYZE/RMS_FILE presents you with two options at level 4. You can proceed down the path that begins with a level of area descriptors (see Figure ARMS–3) or you can choose the path that begins with a level of key descriptors (see Figure ARMS–4).

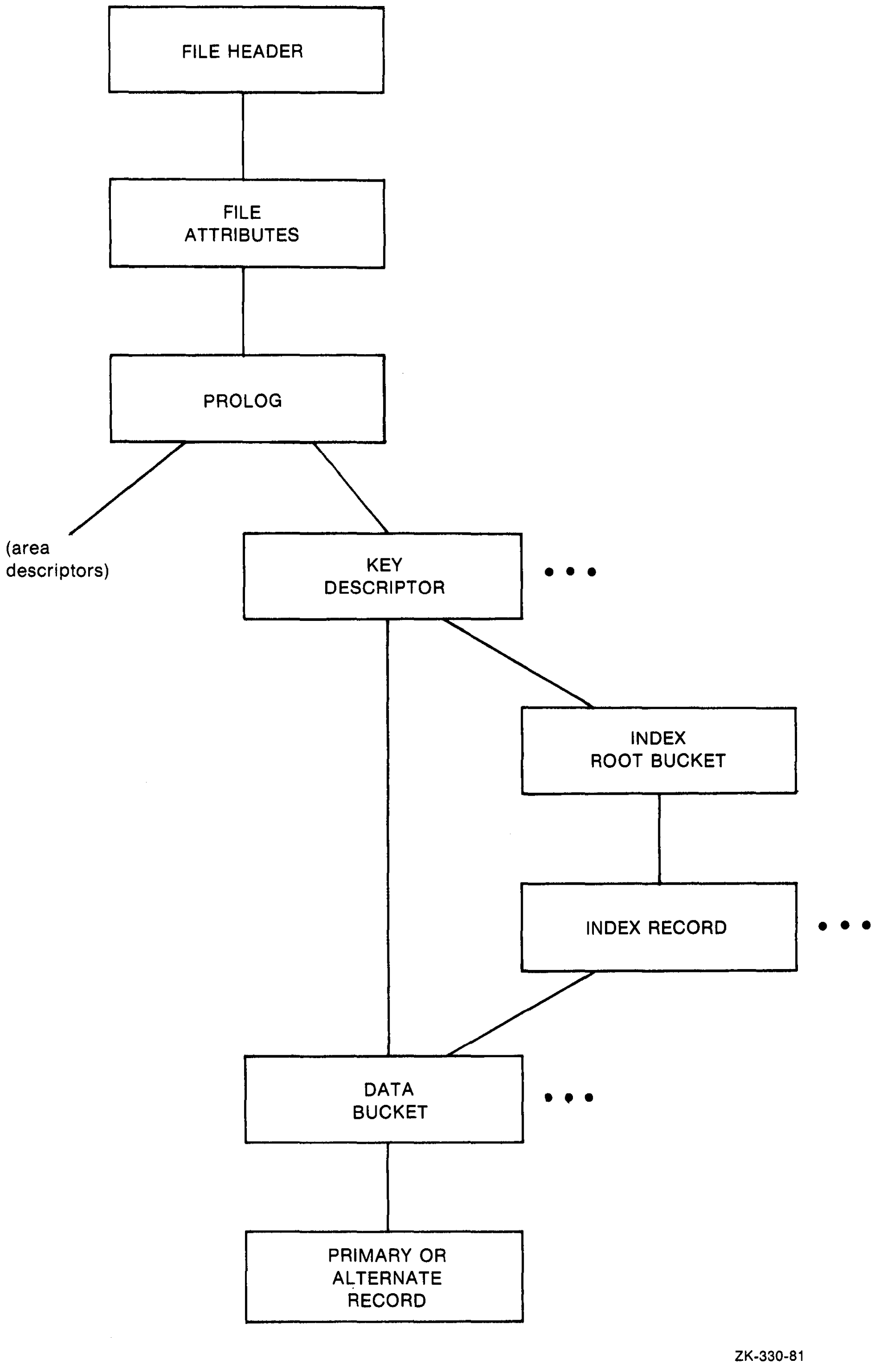
Figure ARMS–3 AREA DESCRIPTOR Path



Level 5 is the final level in a relative file. This level contains the record cells that are accessible individually. For indexed files, the contents of level 5 depend on whether you have chosen the area descriptor path or the key descriptor path:

- If you select the area descriptor path, level 5 is the lowest level and it contains reclaimed data records—that is, records that are effectively empty and are available for storing new data.
- If you select the key descriptor path, ANALYZE/RMS_FILE gives you the option of proceeding to view the index root bucket or the data bucket for the selected key, or you may choose to traverse the level laterally and view another key.

Figure ARMS–4 KEY DESCRIPTOR Path



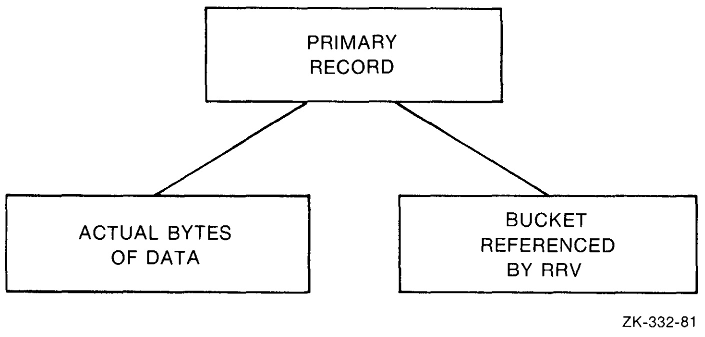
When you choose to view the index root bucket, the next level down contains the index record for the selected key. After viewing the index record, ANALYZE/RMS_FILE provides you with direct access to the first data bucket for the selected key.

At the data bucket level, you may choose to view the data record or you may traverse the data level laterally and select another data bucket for the selected key.

The structure of an indexed file is more complicated than that of sequential and relative files. From the PROLOG level, the structure branches to the AREA DESCRIPTORs and the KEY DESCRIPTORs. Each AREA DESCRIPTOR describes the attributes and the virtual block numbers for the different file areas. The KEY DESCRIPTOR path contains the primary index structures (and data records) as well as the alternate index structures.

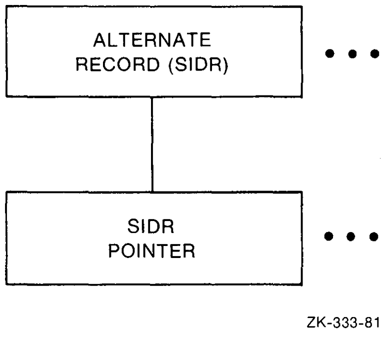
There are two types of record structures — primary records and alternate records. If you follow the primary index structure (key = 0), you find the primary record structures, which contain the actual data records (see Figure ARMS–5). You can examine the actual bytes of data in the record. If the record has been moved to another bucket as a result of a bucket split, you can examine the bucket to which the record reference vector (RRV) points. An RRV is a forwarding pointer that a record leaves behind in its former bucket location when it moves to a new bucket.

Figure ARMS–5 Structure of Primary Records



If you follow any of the alternate index structures, you find the alternate record structures, which contain the secondary index data records (SIDRs). A SIDR consists of an alternate key value and one or more pointers to the actual data records in the primary index structure (see Figure ARMS–6).

Figure ARMS–6 Data Buckets in the Alternate Index Structures



2 Using ANALYZE/RMS_FILE with DECnet–VAX

The ANALYZE/RMS_FILE command is supported only for the examination of files generated by VMS RMS or by RMS–11.

You use the ANALYZE/RMS_FILE command over a network to analyze the internal structure of a remote VMS RMS or RMS–11 file in exactly the same way that you use it to analyze the internal structure of a local VMS RMS or RMS–11 file. For example, you can specify the /FDL qualifier to generate an FDL file from the data file. Using other qualifiers, you can check the file structure for errors, generate a statistical report on the file's structure and use, or enter interactive mode to explore the structure of the file. However, you can specify only one of these qualifiers in each command.

Note that you need the NETMBX privilege to execute the ANALYZE/RMS_FILE command over a network.

3 Handling Error Conditions

This section describes the way you handle ANALYZE/RMS_FILE errors for two general error categories: nonjournaling errors and journaling errors. Even if you do not have VAX RMS Journaling software, you may find that you have imported files marked for VAX RMS Journaling from another system or from other nodes within the VAXcluster.

3.1 Nonjournaling Errors

If you receive any ANALYZE/RMS_FILE error messages while analyzing a file interactively, the file has been corrupted by a serious error. Note that ANALYZE/RMS_FILE errors are not listed in the *VMS Message Utility Manual* because in all cases the user response to errors signaled by ANALYZE/RMS_FILE is identical, as described in the following paragraphs.

If the application program encounters errors during noninteractive analysis, ANALYZE/RMS_FILE returns to the program, as *exit status*, the first occurrence of the most severe error it encounters. For example, if a warning (A) and two errors (B and C) are signaled, then the first error (B) is placed in the DCL symbol $STATUS at image exit.

If you have had a hardware problem (for example, a power or disk head failure), then the hardware most likely caused the corruption.

If you have no hardware problems, then a software error may have been introduced through either the user software or the system software. First, verify the user software and computer operations. Possibly, data files may have been corrupted by a process being stopped abnormally, for example, if a STOP/ID or DELETE/ENTRY occurs in the midst of data processing.

One test of whether or not the problem is in the system software is to note the situations where errors occur. For example, if a particular application uses an unusual I/O sequence that seems to result in file corruption, it may be that the problem is in the system software. In a situation like this, you should attempt to reproduce the problem and note precisely the I/O sequence. This information, together with appropriate supporting information, should be submitted with a Software Performance Report (SPR).

In either case, try to fix the problem with the Convert Utility, using the same file specification for both the input file and the output file. If this procedure does not yield the result you want, use the Backup Utility to restore a backup copy of the file.

3.2 Journaling Errors

If VAX RMS Journaling software is not installed on your system, and you attempt to write to a file marked for journaling, the system issues the following error message:

```
%RMS-F-JNS, operation not supported by RMS Journaling
```

If VAX RMS Journaling software is installed and you receive this message, you attempted an operation that is not supported by VAX RMS Journaling. For more information on handling VAX RMS Journaling errors, see the *VAX RMS Journaling Manual.*

To turn off journaling in either case, use the following DCL command:

```
$ SET FILE/NOAI_JOURNAL/NOBI_JOURNAL/NORU_JOURNAL
```

Note that the SET FILE commands for turning off journaling are available to users who do not have VAX RMS Journaling software as well as to users who do.

Another error condition may occur because if you import a file marked for recovery-unit journaling that has active recovery units. This can happen when a file is not recovered properly before the volume is moved to your system.

If you try to back up the file, VMS RMS issues the following error message:

```
%BACKUP-E-OPENIN, error opening DISK$DATA:[USER]FILE.DAT;1 as input
-SYSTEM-F-RUCONFLICT, another facility has active recovery units on file
```

To turn off the active recovery units, use the following DCL command:

```
$ SET FILE/RU_FACILITY=RMS/NORU_ACTIVE
```

Note that this may leave the file in an inconsistent state with respect to recovery units because active recovery units are not rolled back (aborted).

ANALYZE/RMS_FILE Usage Summary

The Analyze/RMS_File Utility (ANALYZE/RMS_FILE) allows you to examine the internal structure of a VMS RMS file by performing the following functions:

- Checking the structure of a file for errors.
- Generating a statistical report on the file's structure and use.
- Entering an interactive mode through which you can explore a file's structure. This analysis can determine whether or not the file is properly designed for its application and can point out improvements to make in the file's File Definition Language (FDL) specification.
- Generating an FDL file from a data file.
- Generating a summary report on the file's structure and use.
- Generating information related to the file's journaling status.

FORMAT

ANALYZE/RMS_FILE *filespec[,...]*

PARAMETER

filespec

Specifies the data file to be analyzed. The default file type is DAT. You can use multiple file specifications and wildcard characters with the /CHECK qualifier, the /RU_JOURNAL qualifier, the /STATISTICS qualifier, and the /SUMMARY qualifier, but not with the /FDL qualifier or the /INTERACTIVE qualifier.

usage summary

Invoke the utility by entering the ANALYZE/RMS_FILE command at the DCL command level. This command can perform only one of the utility functions at a time; in other words, you must enter a new ANALYZE/RMS_FILE command each time you choose a different function.

If you specify the /INTERACTIVE qualifier, exit ANALYZE/RMS_FILE by entering the EXIT command. Otherwise, let the utility run to successful completion.

If ANALYZE/RMS_FILE terminates with an error message, you should try converting the file and then running the utility again. If the error condition persists, verify the integrity of the hardware and software. If the hardware and software appear to be functioning properly, submit a Software Performance Report (SPR) about the condition.

You can control ANALYZE/RMS_FILE output by using the /OUTPUT qualifier. For a more detailed explanation of the /OUTPUT qualifier, refer to the ANALYZE/RMS_FILE Qualifiers section.

ANALYZE/RMS_FILE Usage Summary

During the time that you are using ANALYZE/RMS_FILE to examine the system authorization file (SYSUAF.DAT), you prevent other users from logging in to the system. Similarly, while you are analyzing your mail file, you cannot receive mail. So if you need to analyze these or other shared files, you may want to make a copy of the file and analyze the copy to avoid this problem.

Note: **If you want to analyze files over a network, you need the NETMBX privilege. If you want to analyze journal files using the /RU_JOURNAL qualifier, you must have CMEXEC privilege and you must have access to the [SYSJNL] directory.**

ANALYZE/RMS_FILE QUALIFIERS

This section describes the ANALYZE/RMS_FILE qualifiers and how you use them to select the utility functions. An additional qualifier, /OUTPUT, permits you to specify an output file for storing the results of the specified function.

/CHECK

Checks the integrity of the file and generates a report of any errors in its structure.

FORMAT

/CHECK

PARAMETERS

None.

DESCRIPTION

The report produced by the /CHECK qualifier includes a list of any errors and a summary of the file's structure. If you do not specify an output file, the report is written to the current SYS$OUTPUT device, which is generally your terminal. You can use wildcards and multiple file specifications. If you specify /NOOUTPUT, no report is generated; instead, only the message indicating whether the file has errors is displayed.

The check function is active by default when you use the ANALYZE/RMS_FILE command without any qualifiers. The /CHECK qualifier is not compatible with the /FDL qualifier, the /INTERACTIVE qualifier, the /STATISTICS qualifier, or the /SUMMARY qualifier.

EXAMPLE

```
$ ANALYZE/RMS_FILE/CHECK CUSTFILE
```

This command checks the file CUSTFILE.DAT for errors and displays the report on the terminal.

/FDL

Generates an FDL file describing the VMS RMS data file being analyzed.

FORMAT

/FDL

PARAMETERS

None.

DESCRIPTION

By default, the /FDL qualifier creates a file with the file type FDL and the same file name as the input data file. To assign a different type or name to the FDL file, use the /OUTPUT qualifier. If the data file is corrupted, the FDL file contains ANALYZE/RMS_FILE error messages.

For indexed files, the FDL file contains special analysis sections you can use with the EDIT/FDL Optimize script to make better design decisions when you reorganize the file. For more information on these special analysis sections, refer to the descriptions of the ANALYSIS_OF_AREA and the ANALYSIS_OF_KEY sections of the *VMS File Definition Language Facility Manual.*

You cannot use wildcards or multiple file specifications with the /FDL qualifier. The /FDL qualifier is not compatible with the /CHECK qualifier, the /INTERACTIVE qualifier, the /STATISTICS qualifier, or the /SUMMARY qualifier.

EXAMPLE

```
$ ANALYZE/RMS_FILE/FDL ADDRFILE
```

This command generates an FDL file named ADDRFILE.FDL from the data file ADDRFILE.DAT.

/INTERACTIVE

Begins an interactive examination of the file's structure. You cannot use wildcards or multiple file specifications. For a list of interactive commands, see the ANALYZE/RMS_FILE Commands section.

FORMAT

/INTERACTIVE

PARAMETERS

None.

EXAMPLE

```
$ ANALYZE/RMS_FILE/INTERACTIVE SUPPLIERS.DAT
```

This command begins an interactive session during which you can examine the structure of the data file SUPPLIERS.DAT.

/OUTPUT

Identifies the destination file for the results of the analysis. The /NOOUTPUT qualifier specifies that no output file is to be created. In all cases, ANALYZE/RMS_FILE displays a message indicating whether the data file has errors.

FORMAT

/OUTPUT *[=output-filespec]*
/NOOUTPUT

QUALIFIER VALUE

output-filespec
Identifies the output file for the results of the analysis. The use of the output file depends on which of the other qualifiers you specify.

/CHECK	Places the integrity report in the output file. The default file type is ANL, and the default file name is ANALYZE. If you omit the **output-filespec** parameter, output is written to the current SYS$OUTPUT device, which is generally your terminal.
/FDL	Places the resulting FDL specification in the output file. The default file type is FDL, and the default file name is that of the input file.
/INTERACTIVE	Places a transcript of the interactive session in the output file. The default file type is ANL, and the default file name is ANALYZE. If you omit the **output-filespec** parameter, no transcript of your interactive session is produced.
/RU_JOURNAL	Places the recovery-unit journal information in the output file. The default file type is ANL, and the default file name is ANALYZE. If you omit the **output-filespec** parameter, output is written to the current SYS$OUTPUT device, which is generally your terminal.
/STATISTICS	Places the statistics report in the output file. The default file type is ANL, and the default file name is ANALYZE. If you omit the **output-filespec** parameter, output is written to the current SYS$OUTPUT device, which is generally your terminal.
/SUMMARY	Places the summary report in the output file. The default file type is ANL, and the default file name is ANALYZE. If you omit the **output-filespec** parameter, output is written to the current SYS$OUTPUT device, which is generally your terminal.

EXAMPLES

1
```
$ ANALYZE/RMS_FILE/STATISTICS/OUTPUT=.TXT SEQ.ADD
```

This command generates a statistics report named ANALYZE.TXT from the data file SEQ.ADD.

2
```
$ ANALYZE/RMS_FILE/NOOUTPUT/CHECK PARTS_INVENTORY.DAT
```

This command checks the structure of the data file PARTS_INVENTORY.DAT. No output is produced except the message telling whether the data file contains errors.

/RU_JOURNAL

Provides information about recovery-unit journaling, where applicable.

FORMAT

/RU_JOURNAL

PARAMETERS

None.

DESCRIPTION

You can use the /RU_JOURNAL qualifier on any file, but it is inoperative on files not marked for recovery-unit journaling.

This qualifier provides the only way of accessing a file that would otherwise be inaccessible because of unresolved recovery units. This situation might be the result of an unavailable recovery-unit journal file or because of unavailable data files that were included in the recovery unit.

To use the /RU_JOURNAL qualifier, your process must have both CMEXEC privilege and access to the [SYSJNL] directory (either SYSPRV privilege, or access for UIC [1,4]).

This qualifier is compatible with all of the ANALYZE/RMS_FILE qualifiers, and you can use it with wildcards and multiple file specifications.

When you specify the /RU_JOURNAL qualifier, ANALYZE/RMS_FILE provides you with the following data for each active recovery unit:

- The journal file specification and the journal creation date
- The recovery-unit identification, recovery-unit start time, cluster system identification number (CSID), and process identification (PID)
- Information about the files involved in the recovery unit, including the file specification, the name of the volume where the file resides, the file identification, the date and time the file was created, and the current status of the file
- The state of the recovery unit — active, none, started, committed, or not available (for more information, see the *VAX RMS Journaling Manual*)
- An error statement

EXAMPLE

```
$ ANALYZE/RMS_FILE/RU_JOURNAL SAVINGS.DAT
```

This command generates information regarding the journaling status of the data file SAVINGS.DAT.

/STATISTICS

Specifies that a report is to be produced containing statistics about the file.

FORMAT

/STATISTICS

PARAMETERS

None.

DESCRIPTION

The /STATISTICS qualifier is used mainly on indexed files.

By default, if you do not specify an output file with the /OUTPUT qualifier, the statistics report is written to the current SYS$OUTPUT device, which is generally your terminal.

The /STATISTICS qualifier is not compatible with the /CHECK qualifier, the /FDL qualifier, the /INTERACTIVE qualifier, or the /SUMMARY qualifier.

EXAMPLE

```
$ ANALYZE/RMS_FILE/STATISTICS SEQ.DAT
```

This command generates a statistics report from the data file SEQ.DAT and displays the report on the current SYS$OUTPUT device, which is generally your terminal.

/SUMMARY

Specifies that a summary report is to be produced containing information about the file's structure and use.

FORMAT

/SUMMARY

PARAMETERS

None.

DESCRIPTION

The /SUMMARY qualifier generates a summary report containing information about the file's structure and use.

If the file has no errors, the output generated from the /SUMMARY qualifier is identical to that produced by the /CHECK qualifier. Unlike the /CHECK qualifier, however, the /SUMMARY qualifier does not check the structure of your file, so output is generated more quickly.

Do not use this qualifier with the /CHECK qualifier, the /FDL qualifier, the /INTERACTIVE qualifier, or the /STATISTICS qualifier.

EXAMPLE

```
$ ANALYZE/RMS_FILE/SUMMARY INVENTORY.DAT
```

This command generates a summary report from the data file INVENTORY.DAT and displays the report on the current SYS$OUTPUT device, which is generally your terminal.

ANALYZE/RMS_FILE COMMANDS

This section describes the ANALYZE/RMS_FILE commands and shows you how to use them in the interactive mode.

In the interactive mode, you use various commands to move through the file structure, examining its various components. Interactive sessions always begin at the FILE HEADER level.

AGAIN

Redisplays the structure you are currently viewing.

FORMAT AGAIN

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLE

```
FIXED PROLOG

        Number of Areas: 8, VBN of First Descriptor: 3
        Prolog Version : 3

ANALYZE> AGAIN

FIXED PROLOG

        Number of Areas: 8, VBN of First Descriptor: 3
        Prolog Version : 3
```

This command redisplays the FIXED PROLOG structure.

BACK

Displays a previous structure at the current level, if one exists.

FORMAT

BACK *[n]*

PARAMETER

n
Specifies the number of times that the structure pointer moves back.

QUALIFIERS

None.

DESCRIPTION

You can use the optional parameter *n* instead of entering multiple BACK commands. For example, the command BACK 6 has the same effect as six BACK commands.

EXAMPLES

1

```
ANALYZE> BACK
```

This command displays the previous structure at the current level. For example, if you are currently viewing the second key descriptor of the primary key, this command displays the primary key descriptor.

2

```
ANALYZE> BACK 3
```

This command displays the third structure back at the current level.

DOWN

Moves the structure pointer down to the next level. From the FILE HEADER level, the first command you enter is the DOWN command, which moves the structure pointer to the FILE ATTRIBUTE level.

FORMAT

DOWN *[branch]*

PARAMETER

branch

Specifies the branch you want to follow when the current level has several branches. If there are several branches from the current level and you do not specify a value for the **branch** parameter, ANALYZE/RMS_FILE prompts you by displaying a list of possible branches.

You can also use a question mark after the DOWN command to obtain a list of the possible branches.

QUALIFIERS

None.

EXAMPLES

1

```
ANALYZE> DOWN ?
%ANLRMS-I-DOWNHELP, The following is a list of paths down from this structure:
%ANLRMS-I-DOWNPATH, AREAS         Area descriptors
%ANLRMS-I-DOWNPATH, KEYS          Key descriptors
```

This command displays the branches available to you from the current location in the file structure. In this case, you can specify the AREAS branch or the KEYS branch.

2

```
ANALYZE> DOWN AREAS
AREA DESCRIPTOR #0 (VBN 3, offset %X'0000')

        Bucket Size: 1
        Reclaimed Bucket VBN: 0
        Current Extent Start: 1, Blocks: 9, Used: 4, Next: 5
        Default Extend Quantity: 2
        Total Allocation: 9
```

This command displays information about the descriptor structure for the first area in the file.

DUMP

Displays a hexadecimal dump of the specified virtual block.

FORMAT

DUMP *n*

PARAMETER

n
Specifies the virtual block number from which you want a dump. The number can be decimal or hexadecimal. The format for a hexadecimal number is %X*n*.

QUALIFIERS

None.

EXAMPLE

```
ANALYZE> DUMP 10
DUMP OF VIRTUAL BLOCK 10:
          7  6  5  4  3  2  1  0             01234567
         ------------------------            --------
         73 20 73 27 65 6C 69 66|  0000     |file's s|
         65 72 75 74 63 75 72 74|  0008     |tructure|
         20 75 6F 59 00 43 00 2E|  0010     |..C.You |
         20 65 73 75 20 6E 61 63|  0018     |can use |
         66 20 4C 44 46 20 6E 61|  0020     |an FDL f|
         64 6F 72 70 20 65 6C 69|  0028     |ile prod|
         20 79 62 20 64 65 63 75|  0030     |uced by |
         2F 45 5A 59 4C 41 4E 41|  0038     |ANALYZE/|
         45 4C 49 46 5F 53 4D 52|  0040     |RMS_FILE|
         74 6F 20 68 74 69 77 20|  0048     | with ot|
         20 53 4D 56 20 72 65 68|  0050     |her VMS |
         6C 69 74 75 20 53 4D 52|  0058     |RMS util|
         20 20 20 00 65 69 74 69|  0060     |ities.  |
```

This command shows the first part of a dump of virtual block number (VBN) 10. The left column shows the bytes of the block in hexadecimal, read from right to left. The middle column shows the byte offset in hexadecimal from the beginning of the blocks. In the right column, the character equivalents of each byte are displayed. Nonprintable characters are represented by a period (.).

EXIT

Ends an interactive session.

FORMAT

EXIT

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLE

```
ANALYZE> EXIT
$
```

This command terminates the interactive session and returns you to the DCL command level.

FIRST

Displays the first structure on the current level.

FORMAT

FIRST

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLE

```
ANALYZE> FIRST
```

If you are examining the primary and alternate key descriptors, this command displays the first key descriptor.

HELP

Displays help information about the interactive commands.

FORMAT **HELP** *[keyword...]*

PARAMETER

keyword
Specifies the interactive command you want help with.

QUALIFIERS *None.*

EXAMPLE

```
ANALYZE> HELP

  Information available:

  AGAIN    BACK     DOWN     DUMP     EXIT     File_Structure
  FIRST    HELP     New_features      NEXT     POSITION   Radix
  REST     TOP      UP
```

This command shows the available help topics.

```
Topic? AGAIN
AGAIN
  This command displays the current structure one more time.
Topic?
```

This command displays information about the AGAIN command.

NEXT

Displays the next structure at the current level, if one exists. Because NEXT is the default command, pressing the RETURN key is equivalent to executing a NEXT command.

FORMAT

NEXT *[n]*

PARAMETER

n
Specifies the number of times the structure pointer moves forward.

QUALIFIERS

None.

DESCRIPTION

You can use the optional parameter *n* instead of entering multiple NEXT commands. For example, the command NEXT 6 has the same effect as six NEXT commands (or pressing the RETURN key six times).

EXAMPLES

1
```
ANALYZE> NEXT
```

This command displays the next structure at the current level. For example, if you are viewing key descriptors, this command displays the next key descriptor.

2
```
ANALYZE> NEXT 3
```

This command moves the location pointer forward three times. For example, if you are viewing the first structure at the current level, this command displays the fourth structure.

POSITION/BUCKET

Positions the structure pointer to a specific bucket of an indexed file or a relative file.

FORMAT

POSITION/BUCKET *bucket_vbn [/INDEX=n]*

PARAMETER

bucket_vbn

The virtual block number (VBN) of the bucket at which the pointer is to be positioned. If the bucket has a length greater than one block, use the VBN of the beginning of the bucket.

QUALIFIER

[/INDEX=n]

Specifies the relative key for the bucket of an indexed file. The /INDEX qualifier is necessary only when the index number information is unavailable in the bucket header. For example, you use this qualifier to analyze a Prolog 1 or Prolog 2 file (no bucket header) or to analyze a Prolog 3 file with a corrupted bucket header. You can also use this qualifier to override the index number in a Prolog 3 file bucket header.

The number you use specifies the key. For example, */INDEX=0* specifies that the bucket is a primary index or primary data bucket, and */INDEX=1* specifies that the bucket is found in the first alternate index structure.

DESCRIPTION

The POSITION/BUCKET command lets you position the structure pointer to a specific bucket of your file. You can use this command to bypass step-by-step positioning. You can also use it to position the structure pointer at a bucket that is inaccessible because of structural errors in the file.

When the structure pointer is positioned at the beginning of the bucket, you can step forward or down through the index structure using the NEXT or DOWN command. If you enter an UP command when the structure pointer is positioned at the beginning of the bucket, ANALYZE/RMS_FILE positions the pointer to the bucket's key descriptor. If you enter a BACK command when the structure pointer is positioned at the beginning of the bucket, ANALYZE/RMS_FILE displays an appropriate error message and the pointer remains stationary.

Using the POSITION/BUCKET command allows you to specify a particular bucket header from which key descriptor information and valid path information are derived. ANALYZE/RMS_FILE does not verify that the specified VBN is at the beginning block of a bucket. If ANALYZE/RMS_FILE displays a series of error messages when you enter the POSITION/BUCKET command, it may be that the structure pointer is not positioned at the beginning of the bucket, or it may be that you specified an incorrect index number with the /INDEX qualifier.

EXAMPLE

```
ANALYZE> POSITION/BUCKET 4
BUCKET HEADER (VBN 4)

          Check Character: %X'93'
          Key of Reference: 0
          VBN Sample: 4
          Free Space Offset: %X'0055'
          Free Record ID: 24
          Next Bucket VBN: 36
          Level: 0
          Bucket Header Flags:
             (0)  BKT$V_LASTBKT    0
```

This command displays the information for the bucket that begins at VBN4. Because this is a Prolog 3 file, you do not have to specify the key using the /INDEX=n qualifier. In a Prolog 3 file, the key information is available in the bucket header (Key of Reference: 0).

POSITION/RECORD

Positions the pointer at a specific record in an indexed or relative file.

FORMAT

POSITION/RECORD *record-offset*

PARAMETER

record-offset
The offset (in bytes) from the beginning of the bucket to the desired record. By default, the offset is a decimal number. If you want to use hexadecimal notation to specify the offset, use the format %Xn.

QUALIFIERS

None.

DESCRIPTION

Use this command to display a specific record in the bucket. When the structure pointer is positioned at the desired record, you can move it down and forward to display the various records in the bucket; you cannot display previous records.

The POSITION/RECORD command is valid only when you are positioned at a bucket header. The command positions the structure pointer at the specified byte offset, whether that position is or is not the beginning of a valid record. If the pointer is not positioned at the beginning of a valid record, a series of error messages is generated.

EXAMPLE

```
ANALYZE> POSITION/RECORD %XE
PRIMARY DATA RECORD (VBN 4, offset %X'000E')

                Record Control Flags:
                       (2)  IRC$V_DELETED    0
                       (3)  IRC$V_RRV        0
                       (4)  IRC$V_NOPTRSZ    0
                       (5)  IRC$V_RU_DELETE  0
                       (6)  IRC$V_RU_UPDATE  0
                Record ID: 11
                RRV ID: 11, 4-Byte Bucket Pointer: 4
                Key:
                        7  6  5  4  3  2  1  0          01234567
                       ------------------------         --------
                        00 00 00 00 00 00 00 02|  0000  |........|
```

This command positions the pointer at byte offset %XE, which is the location of the beginning of a record. This command is valid because the pointer was positioned at a bucket header before the POSITION/RECORD %XE command was entered.

REST

Sequentially displays structures at the current level.

FORMAT

REST

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLE

```
ANALYZE> REST
```

This command displays each structure at the current level. For example, if you are viewing the primary and alternate key descriptors, the REST command displays each key descriptor sequentially.

TOP

Displays the FILE HEADER level.

FORMAT

TOP

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLE

```
ANALYZE> TOP
FILE HEADER

        File Spec: DISK$:[JONES.PROGRAM]INVENTORY.DAT;6
        File ID: (6367,16,1)
        Owner UIC: [DOC,DOE]
        Protection: System: RWE, Owner: RWED, Group: R, World:
        Creation Date:    13-NOV-1988 09:10:29.83
        Revision Date:    16-DEC-1988 14:10:37:16, Number: 4
        Expiration Date: none specified
        Backup Date:     none posted
        Contiguity Options:  none
        Performance Options: none
        Reliability Options: none
        Journaling Enabled: none
```

This command displays the file header information for the file INVENTORY.DAT.

UP

Displays the data structures at the next higher level.

FORMAT

UP

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLE

```
ANALYZE> UP
```

This command positions the pointer at the next higher level of the file's structure. For example, if you are currently examining the RMS FILE ATTRIBUTES level, entering the UP command positions you at the FILE HEADER level and displays that level.

ANALYZE/RMS_FILE EXAMPLES

1
```
$ ANALYZE/RMS_FILE/INTERACTIVE/OUTPUT=INVENTORY INVENTORY.DAT
```

This command begins an interactive session during which you can examine the structure of the data file INVENTORY.DAT. A transcript of the session is placed in the output file INVENTORY.ANL.

2
```
$ ANALYZE/RMS_FILE/NOOUTPUT *.*;*
```

This command verifies the structural integrity of all files in the current default directory.

3
```
$ ANALYZE/RMS_FILE/FDL PARTS.DAT
```

This command produces the FDL file PARTS.FDL from the data file PARTS.DAT. Assuming that PARTS.DAT is an indexed file, the new FDL file contains two special sections that FDL files created with the Edit/FDL Utility do not have: ANALYSIS_OF_AREA and ANALYSIS_OF_KEY. You can use these sections with the EDIT/FDL Optimize script to tune your original data file PARTS.DAT. To complete the tuning cycle, enter the following DCL commands:

```
$ EDIT/FDL/ANALYSIS=PARTS/SCRIPT=OPTIMIZE PARTS
$ CONVERT/FDL=PARTS PARTS.DAT *
```

4
```
$ ANALYZE/RMS_FILE DENVER::DB1:[PROD]RUN.DAT
```

This command analyzes the structure of the file RUN.DAT residing at remote node DENVER.

5
```
$ ANALYZE/RMS_FILE/FDL/OUTPUT=TEST.FDL
$_File(s): DENVER::DB1:[PROD]RUN.DAT
```

This command analyzes the structure of the file RUN.DAT at remote node DENVER and generates the FDL file TEST.FDL at the local node.

Index

Reader's Comments

VMS Analyze/RMS_File Utility Manual
AA–LA79A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less ______________________________

What I like best about this manual is ______________________________

What I like least about this manual is ______________________________

I found the following errors in this manual:

Page	Description

Additional comments or suggestions to improve this manual:

I am using **Version** ______ of the software this manual describes.

Name/Title ______________________________ Dept. ______________

Company ______________________________ Date ______________

Mailing Address ______________________________

______________________________ Phone ______________

– Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

– Do Not Tear - Fold Here

Cut Along Dotted Line