VAX/VMS Analyze/RMS_File Utility Reference Manual

Order Number: AA–Z404B–TE

April 1986

This document describes the VAX/VMS Analyze/RMS_File Utility.

Revision/Update Information: This manual supercedes the *VAX/VMS Analyze/RMS_File Utility Reference Manual* Version 4.0.

Software Version: VAX/VMS Version 4.4

digital equipment corporation
maynard, massachusetts

April 1986

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem–10	PDP	VT
DECSYSTEM–20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital

ZK–3031

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire
03061

CANADA

Digital Equipment
of Canada Ltd.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

Digital Equipment Corporation
PSG Business Manager
c/o Digital's 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.

This document was prepared using an in-house documentation production system. All page composition and make-up was performed by TEX, the typesetting system developed by Donald E. Knuth at Stanford University. TEX is a registered trademark of the American Mathematical Society.

ANALYZE/RMS_FILE Contents

Preface

Intended Audience

This manual is intended for all programmers using VAX RMS data files. This audience includes high-level language users who use only their language's input/output statements.

Structure of This Document

This document is composed of five major sections.

The Format Section is an overview of the Analyze/RMS_File Utility and is intended as a quick reference guide. DCL commands that invoke the Analyze /RMS_File Utility as well as all command qualifiers and parameters are listed in the format summary section. The usage summary describes how to invoke and exit from the Analyze/RMS_File Utility, how to direct output, and any restrictions you should be aware of.

The Description Section explains how to use the Analyze/RMS_File Utility.

The Qualifier Section describes each DCL command qualifier. Qualifiers appear in alphabetical order.

The Command Section describes each interactive Analyze/RMS_File Utility command. Commands appear in alphabetical order.

The Examples Section contains examples of common operations that you perform with the Analyze/RMS_File Utility.

Associated Documents

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

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

Conventions Used in This Document

Convention	Meaning
RET	A symbol with a one- to three-character abbreviation indicates that you press a key on the terminal, for example, RET .
CTRL/x	The phrase CTRL/x indicates that you must press the key labeled CTRL while you simultaneously press another key, for example, CTRL/C, CTRL/Y, CTRL/O.
$ SHOW TIME 05-JUN-1985 11:55:22	Command examples show all output lines or prompting characters that the system prints or displays in black letters. All user-entered commands are shown in red letters.
$ TYPE MYFILE.DAT . . .	Vertical series of periods, or ellipsis, mean either that not all the data that the system would display in response to the particular command is shown or that not all the data a user would enter is shown.
file-spec,...	Horizontal ellipsis indicates that additional parameters, values, or information can be entered.
[logical-name]	Square brackets indicate that the enclosed item is optional. (Square 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 following new interactive commands have been added to the Analyze /RMS_file Utility, and their descriptions are included in this manual:

- The BACK command moves the structure pointer to the previous node if one exists within the current level, and displays that node. You can also specify the number of structures that the pointer is to be moved by using the optional parameter BACK n, where n is an integer.
- The NEXT command now accepts the optional parameter n (NEXT n) to specify the number of structures that the pointer is to be moved forward.
- The POSITION/BUCKET command lets you position the pointer to a specific bucket of your file. You can use this command to bypass the step-by-step positioning, and you can also use it to position the pointer at a bucket that would otherwise be inaccessible due to structural errors in the file.
- The POSITION/RECORD command lets you position the pointer at a specific record in the current bucket, allowing you to access subsequent structures more easily.

ANALYZE/RMS_FILE

The Analyze/RMS_File Utility (ANALYZE/RMS_FILE) allows you to examine the internal structure of a VAX RMS file.

ANALYZE/RMS_FILE can perform the following five functions:

- Check the structure of a file for errors.
- Generate a statistical report on the file's structure and use.
- Enter 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.
- Generate an FDL file from a data file.
- Generate a summary report on the file's structure and use.

FORMAT

ANALYZE/RMS_FILE *file-spec[,...]*

Command Qualifiers	Defaults
/CHECK	*See text.*
/FDL	*See text.*
/INTERACTIVE	*See text.*
/[NO]OUTPUT[=file-spec]	*/OUTPUT*
/STATISTICS	*See text.*
/SUMMARY	*See text.*

Command Parameters

file-spec

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

usage summary

Invoking

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

Exiting

Exit the utility by typing the EXIT command if the /INTERACTIVE qualifier was specified. Otherwise, let the utility run to successful completion.

Directing Output

You can control ANALYZE/RMS_FILE output with the /OUTPUT command qualifier. For a more detailed explanation of the /OUTPUT qualifier, refer to the Command Qualifiers Section.

Privileges/Restrictions
During the time that you use ANALYZE/RMS_FILE to examine the system authorization file (SYSUAF.DAT), you prevent other users from logging into 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.

NETMBX privilege is required if you want to analyze a file over a network.

commands

Syntax
ANALYZE> command [parameter]

ANALYZE/RMS_FILE Commands
AGAIN
BACK [n]
DOWN [branch]
DUMP n
EXIT
FIRST
HELP [keyword...]
NEXT [n]
POSITION/BUCKET bucket-vbn [/INDEX=n]
POSITION/RECORD record-offset
REST
TOP
UP

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 VAX RMS file. It can check the structure of the file for errors, generate a statistical report on the structure and use of the file, or generate an FDL file from a data file.

The commands listed above can be used when you analyze a file interactively. You can use these commands by specifying the /INTERACTIVE qualifier in your DCL command line.

FDL files created with ANALYZE/RMS_FILE have special sections that contain statistics about the data file's structure. You can use an FDL file produced by ANALYZE/RMS_FILE with the other VAX RMS utilities.

1 Analyzing VAX RMS File Structure Interactively

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

ANALYZE/RMS_FILE imposes a hierarchical tree structure on the internal VAX RMS file structure. Each data structure in the file (each element in the tree) is a *node*, with a branch for each pointer in the data structure. The file header is always the root node. Each of the three file organizations (sequential, relative, and indexed) has its own tree structure.

The tree structures for sequential and relative files are shown in Figures ARMS–1 and ARMS–2.

Figure ARMS–1 Tree Structure for Sequential Files

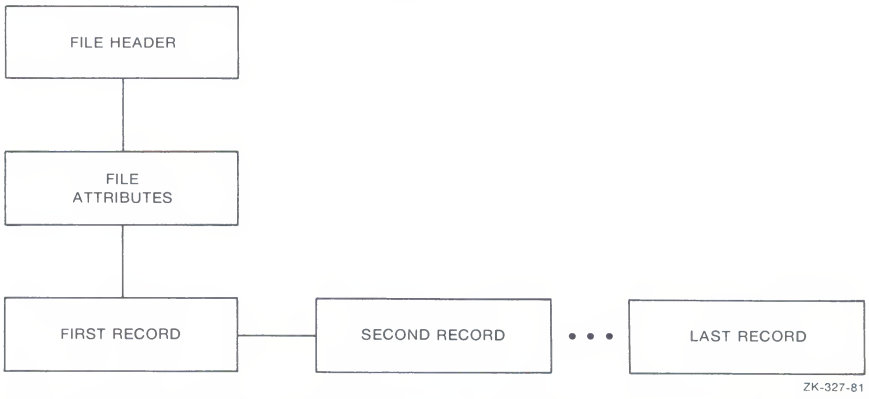
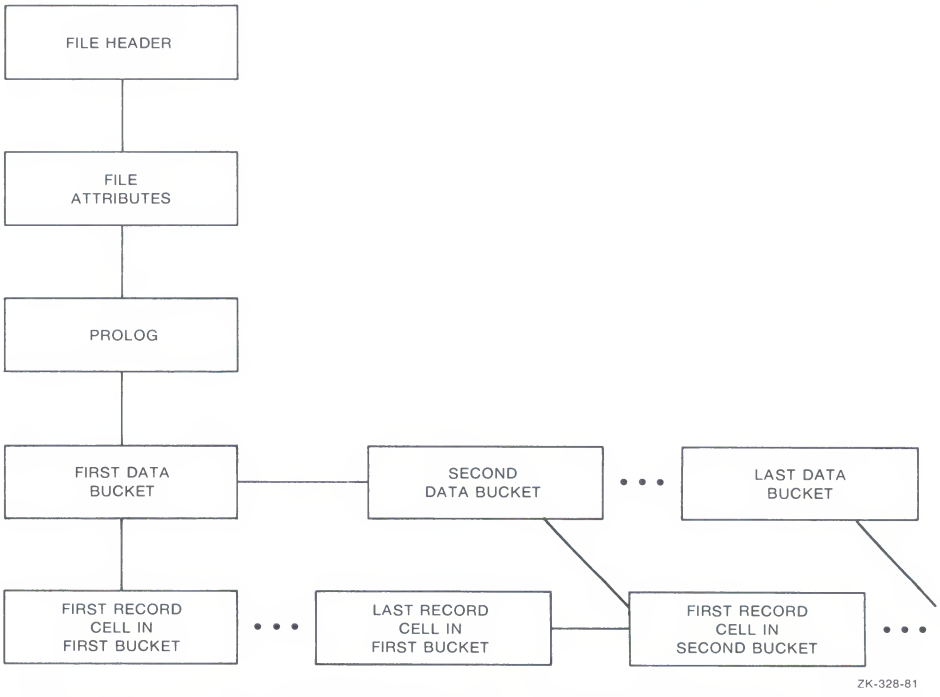


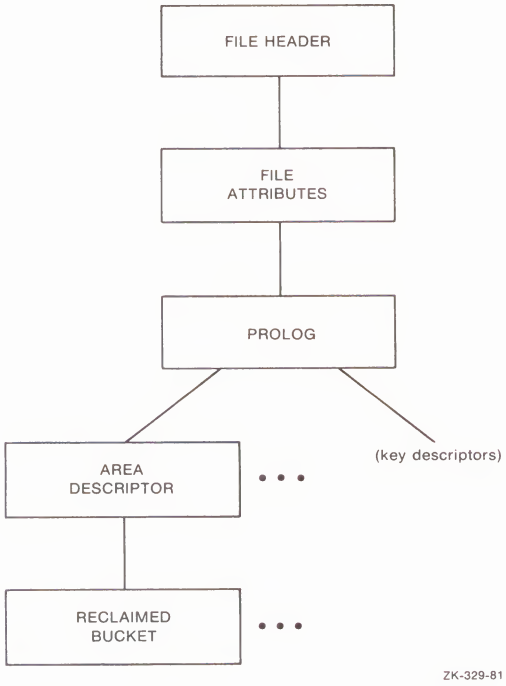
Figure ARMS–2 Tree Structure for Relative Files



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.

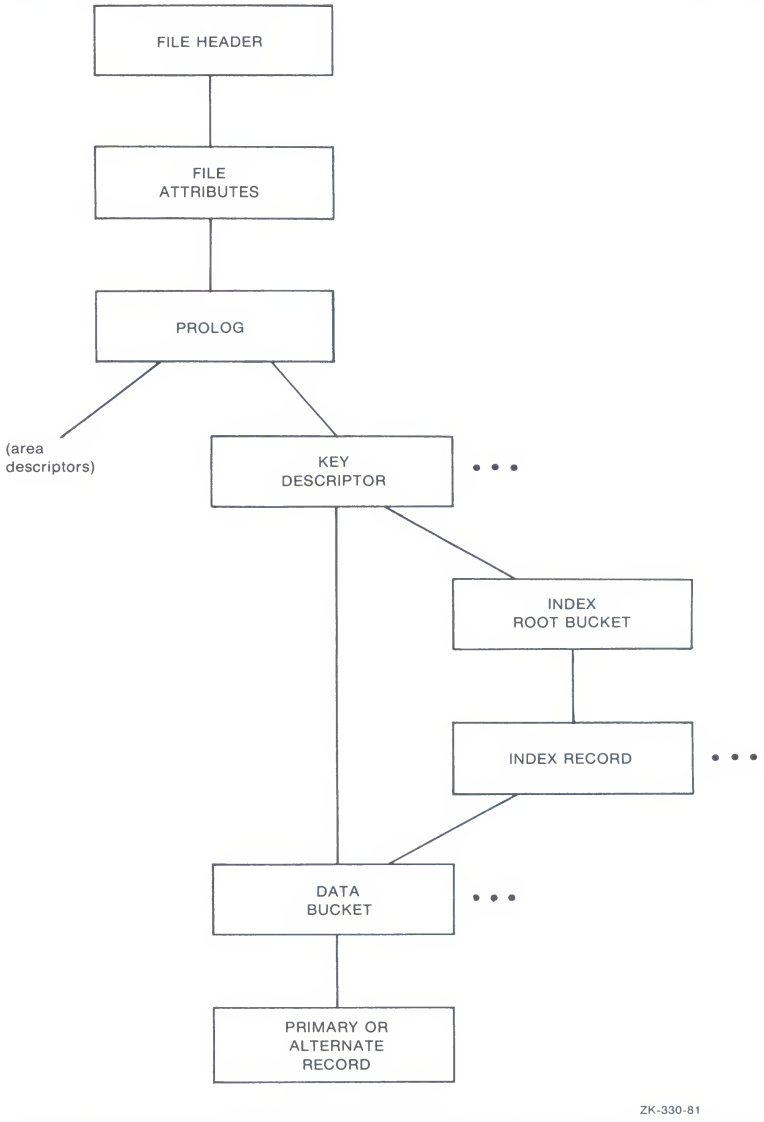
The full structure following the AREA DESCRIPTOR path is shown in Figure ARMS–3.

Figure ARMS–3 AREA DESCRIPTOR Path



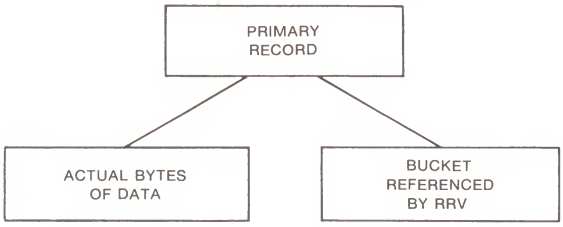
The structure following the KEY DESCRIPTOR path appears in Figure ARMS–4.

Figure ARMS–4 KEY DESCRIPTOR Path



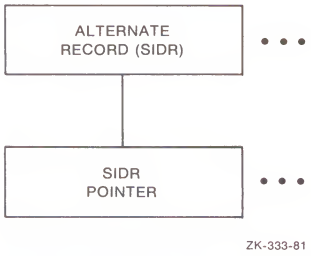
There are two types of record structures—primary records and alternate records. If you follow the primary index structure (key = 0), you find the primary record structures, which contain the actual data records (see Figure ARMS–5). You can examine the actual bytes of data in the record. If the record 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.

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

You can use the ANALYZE/RMS_FILE command over a network to analyze the internal structure of a remote VAX RMS or RMS–11 file. This command is, however, supported only for the examination of files generated by VAX RMS or RMS–11. Otherwise, using ANALYZE/RMS_FILE over a network is no different than using it on your home node. You can, for instance, specify the /FDL command qualifier to generate an FDL (File Definition Language) file from the data file. Using other command 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 command qualifiers in each command.

Remember that NETMBX privilege is required to execute ANALYZE/RMS_FILE over a network.

3 Handling Exception Conditions

If you receive any ANALYZE/RMS_FILE exception conditions, the file has been corrupted by a serious error. If you have had a hardware problem (for example, a power failure or a disk head crash), then the hardware may be more likely to have caused the corruption. If you have not had any hardware problems, then a software error may be more likely to have caused the problem.

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 that you want, use the Backup Utility to restore the backup copy of the file.

Also, if ANALYZE/RMS_FILE finds an error while you are analyzing your file noninteractively, it returns the first error of the worst severity that it discovers as the image exit status. In other words, 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.

COMMAND QUALIFIERS

The ANALYZE/RMS_FILE command qualifiers allow you to choose which of the five functions you want the utility to perform.

/CHECK

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

FORMAT /CHECK

qualifier values *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 telling whether or not the file has errors is displayed.

The /CHECK function is the default when you issue the ANALYZE/RMS_FILE command without any qualifiers. Do not use this qualifier with /FDL, /INTERACTIVE, /STATISTICS, or /SUMMARY; the function of /CHECK with these qualifiers is not compatible.

EXAMPLE

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

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

/FDL

Generates an FDL file that describes the VAX RMS data file being analyzed.

FORMAT /FDL

qualifier values *None.*

DESCRIPTION

By default, the /FDL qualifier creates a file with the FDL file type 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 has been corrupted, then the FDL file generated will contain ANALYZE/RMS_FILE error messages.

For indexed files, the FDL file contains special analysis sections that 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 *VAX/VMS File Definition Language Facility Reference Manual.*

You cannot use wildcards or multiple file specifications. Also, do not use this qualifier with /CHECK, /INTERACTIVE, /STATISTICS, or /SUMMARY; the functions of /FDL are incompatible with these qualifiers.

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 Command Section.

FORMAT /INTERACTIVE

qualifier values *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 output file in which to place the results of the analysis.

FORMAT

/OUTPUT *[=output-file-spec]*
/NOOUTPUT

qualifier value

output-file-spec

Identifies the output file in which to place the results of the analysis. The use of the output file depends on which of the other qualifiers you specified.

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

DESCRIPTION

/NOOUTPUT specifies that no output is to be produced. However, a message indicating whether or not the file has errors is still displayed.

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 had any errors.

/STATISTICS

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

FORMAT /STATISTICS

qualifier values *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.

Do not use this qualifier with /CHECK, /FDL, INTERACTIVE, or /SUMMARY. The function of /STATISTICS is incompatible with these qualifiers.

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 containing information about the file's structure and use is to be produced.

FORMAT /SUMMARY

qualifier values *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, /FDL, /INTERACTIVE, or /STATISTICS qualifiers. The function of /SUMMARY is incompatible with these qualifiers.

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).

COMMANDS

To use the interactive mode, you issue commands to move the structure pointer to nodes within the tree structure. The utility will display the contents of the current node, changing its display as you change the current location. Sessions always begin at the FILE HEADER.

From the header level, the first command to give is always the DOWN command, which moves the current location to the FILE ATTRIBUTE level. The DOWN command moves you down one level in the total structure, the UP command moves you up one level, and the TOP command returns you immediately to the FILE HEADER level from anywhere in the structure.

When the current location is set to a level with multiple nodes on it, you can use the REST command to display the nodes in sequence, or you can use the NEXT command to display them one at a time. The FIRST command moves the current location back to the first structure on a level. The AGAIN command displays the current structure again.

The DUMP command will display a hexadecimal dump of a specified bucket, and the EXIT command ends the interactive session.

AGAIN

Displays the current structure one more time.

FORMAT

AGAIN

command parameters

None.

command 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 displays the current structure one more time.

BACK

Moves the structure pointer to the previous node if one exists within the current level, and displays that node.

FORMAT

BACK *[n]*

command parameter

n
Specifies the number of structures that the pointer is to be moved backward.

command qualifiers

None.

DESCRIPTION

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

EXAMPLES

1
```
ANALYZE> BACK
```

This command positions you at the previous node of the level you are currently examining. For example, if you are examining the second key descriptor of your primary key, this command positions you at, and displays, the primary key descriptor.

2
```
ANALYZE> BACK 3
```

This command positions you at the third node back from the one that you are examining without displaying the intermediate nodes.

DOWN

Moves the structure pointer down to the next level.

FORMAT

DOWN *[branch]*

command parameter

branch

Specifies the branch that you wish to follow if the current node has more than one branch. ANALYZE/RMS_FILE prompts you with a list of possible branches if you do not specify a required branch value.

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

command qualifiers

None.

EXAMPLE

```
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
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 first displays the available branches or paths you can follow from your current position in the file structure. In this case, you can either examine areas or keys.

After you have specified that you want to examine the area descriptors of your file, ANALYZE/RMS_FILE displays information about the first area.

DUMP

Displays a hexadecimal dump of the specified virtual block.

FORMAT

DUMP *n*

command parameter

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

command 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|
         68 74 20 68 74 69 77 20|  0048  | with th|
         2D 72 65 68 74 6F 20 65|  0050  |e other-|
         31 31 2D 58 41 56 00 15|  0058  |..VAX-11|
         69 74 75 20 53 4D 52 20|  0060  | RMS uti|
         52 2E 73 65 69 74 69 6C|  0068  |lities.R|
```

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 the interactive session.

FORMAT

EXIT

command parameters

None.

command qualifiers

None.

EXAMPLE

```
ANALYZE> EXIT
$
```

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

FIRST

Moves the structure pointer to the first node on the current level and displays the node.

FORMAT

FIRST

command parameters

None.

command qualifiers

None.

EXAMPLE

```
ANALYZE> FIRST
```

This command returns you to the first node of the level you are currently examining. For example, if you are examining the primary and alternate key descriptors, this command positions you at, and displays, the first key descriptor.

HELP

Displays help information about the interactive commands.

FORMAT

HELP *[keyword...]*

command parameter

keyword
Specifies the interactive command about which you want more information.

command qualifiers

None.

EXAMPLE

```
ANALYZE> HELP
  Information available:
  AGAIN     BACK      DOWN      DUMP      EXIT      File_Structure
  FIRST     HELP      New_features        NEXT      POSITION    Radix
  REST      TOP       UP
Topic? AGAIN
AGAIN
  This command displays the current structure one more time.
Topic?
```

This command shows the available help topics. It then displays the help information about the interactive command AGAIN.

NEXT

Moves the structure pointer to the next node if one exists within the current level, and then displays that node. Since NEXT is the default command, pressing the RETURN key is equivalent to executing a NEXT command.

FORMAT

NEXT *[n]*

command parameter

n
Specifies the number of structures that the pointer is to be moved forward

command qualifiers

None.

DESCRIPTION

You can use the optional parameter *n* instead of typing many NEXT commands. For example, the command NEXT 6 has the same effect as six consecutive NEXT commands (or six consecutive RETURNs).

EXAMPLES

1
```
ANALYZE> NEXT
```

This command positions you at the next node of the level you are currently examining. For example, if you are examining your primary and alternate key descriptors, this command positions you at, and displays, the next key descriptor.

2
```
ANALYZE> NEXT 3
```

This command positions you three nodes from the one that you are examining, without displaying the intermediate nodes. For example, if you were at the first node, then giving this command (NEXT 3) would position you at and display the fourth node.

POSITION/BUCKET

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

FORMAT

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

command 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.

command qualifiers

[/INDEX=n]

Specifies the relative key for the bucket for an indexed file. This qualifier is necessary only when the index number information is unavailable in the bucket header. For example, you would use this qualifier if you are analyzing a PROLOG 1 or 2 file, if the bucket header in a PROLOG 3 file has been corrupted, or if you wanted to override the index number in a PROLOG 3 file bucket header.

The number that 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 pointer to a specific bucket of your file. You can use this command to bypass the step-by-step positioning, and you can also use it to position the pointer at a bucket that would otherwise be inaccessible due to structural errors in the file.

Once you are positioned at the beginning of the bucket, you can step forward or down through the index structure using the NEXT or DOWN commands. However, if you issue an UP command when positioned at the beginning of the bucket, the pointer will go to the key descriptor of the bucket. If you issue a BACK command, an error message is displayed and the pointer remains at the beginning of the bucket.

Using the POSITION/BUCKET command allows you to specify a particular bucket header, from which key descriptor and valid path information are derived. However, no check is made to verify that the VBN you specify is, in fact, the beginning block of a bucket. If a series of error messages is displayed after you issue the POSITION/BUCKET command, it is probably due either to specifying a VBN that is not a beginning of a bucket, or specifying 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 positions you at VBN 4, and displays the information for the bucket that begins at that VBN. The /INDEX=n qualifier is not used because the file is PROLOG 3 and, therefore, that information is already 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*

command parameter

record-offset

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

command qualifiers

None.

DESCRIPTION

Use this command to position the pointer at a specific record in the current bucket. Once you are positioned at the record, you can move down and forward through structures; however, you cannot move to previous records within that bucket.

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

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

REST

Moves the structure pointer along the remainder of the nodes on the current level, and displays each in turn.

FORMAT

REST

command parameters

None.

command qualifiers

None.

EXAMPLE

```
ANALYZE> REST
```

This command positions you at each successive node of the structure you are currently examining. For example, suppose you are examining your primary and alternate key descriptors. The REST command positions you at and displays each key descriptor, and it then moves on to the next key descriptor, if it exists.

TOP

Moves the structure pointer up to the file header. The file header is displayed.

FORMAT

TOP

command parameters

None.

command qualifiers

None.

EXAMPLE

```
ANALYZE> TOP
FILE HEADER
        File Spec: DISK$:[JONES.PROGRAM]INVENTORY.DAT;6
        File ID: (6367,16,1)
        Owner UIC: [277,377]
        Protection: System: RWE, Owner: RWED, Group: R, World:
        Creation Date:      5-JUL-1986 09:10:29.83
        Revision Date:      5-JUL-1986 09:10:37:16, Number: 4
        Expiration Date: none specified
        Backup Date:      none posted
        Contiguity Options:  none
        Performance Options: none
        Reliability Options: none
```

This command positions you at and displays the file header of your file.

UP

Moves the structure pointer up to the next higher level. The node at that level is displayed.

FORMAT

UP

command parameters

None.

command qualifiers

None.

EXAMPLE

```
ANALYZE> UP
```

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

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—ANALYSIS_OF_AREA and ANALYSIS_OF_KEY—that FDL files created with the Edit/FDL Utility do not have. These sections can be used with the EDIT/FDL Optimize script to tune your original data file PARTS.DAT. To complete the tuning cycle, you would issue the following 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

A

B

C

D

E

F

H

I

K

N

O

P

R

S

T

U

V

W

X

VAX/VMS
Analyze/RMS_File
Utility Reference Manual
AA–Z404B–TE

READER'S COMMENTS

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent:

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

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

SSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03062-2698

Do Not Tear - Fold Here

Cut Along Dotted Line

VAX/VMS
Analyze/RMS_File
Utility Reference Manual
AA–Z404B–TE

READER'S COMMENTS

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent:

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or Country

— — 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

SSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03062-2698

— — Do Not Tear - Fold Here —

Cut Along Dotted Line