VMS

digital

VMS Convert and Convert/Reclaim Utility Manual

Order Number AA–LA80A–TE

VMS Convert and Convert/Reclaim Utility Manual

Order Number: AA–LA80A–TE

April 1988

This document describes the VMS Convert Utility and the VMS Convert/Reclaim Utility.

Revision/Update Information: This manual supersedes the *VAX/VMS Convert and Convert/Reclaim Utility Reference Manual,* Version 4.0.

Software Version: VMS Version 5.0

digital equipment corporation
maynard, massachusetts

April 1988

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™

ZK4537

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.

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 four sections:

- Description—Provides a full description of the Convert Utility (CONVERT) and the Convert/Reclaim Utility (CONVERT/RECLAIM).
- Usage Summary—Outlines the following CONVERT and CONVERT /RECLAIM information:

  -Invoking the utility
  -Exiting from the utility
  -Directing output
  -Restrictions or privileges required

- Qualifiers—Describes the CONVERT and CONVERT/RECLAIM qualifiers, including format, parameters, and examples.
- Examples—Provides additional CONVERT and CONVERT/RECLAIM examples.

Associated Documents

To use CONVERT and CONVERT/RECLAIM, you should be familiar with the following manuals:

- *Guide to VMS File Applications*
- *VMS Analyze/RMS_File Utility Manual*
- *VMS File Definition Language Facility Manual*

Conventions

Convention	Meaning
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

No enhancements have been made to the Convert Utility (CONVERT) or to the Convert/Reclaim Utility (CONVERT/RECLAIM) for VMS Version 5.0. However, support has been added for VMS Record Management Services (RMS) indexed files with collated keys. For more information about collated keys, see the *VMS Record Management Services Manual.*

You do not have to take explicit action to use the CONVERT support for collated keys because CONVERT recognizes indexed files with collated keys and processes them transparently. However, you should be aware of a situation that may develop when you use CONVERT with indexed files that have collated keys.

Collating sequences are identified by a character string that is processed as part of the named collating sequence. Both the National Character Set Utility (NCS) and VMS RMS use the name string for identifying the collating sequence. For more information about NCS, see the *VMS National Character Set Utility Manual.*

Because the collating sequences are physically stored in indexed files that use collated keys, the name is also in the indexed file. When you use CONVERT as shown in the following command line, and the input index file has one or more collated keys, the collating sequences from the input file are passed to the output file, and the records used to populate the output file are sorted accordingly:

```
$ CONVERT input-index-file output-index-file
```

But, if you use a File Definition Language (FDL) file containing the name of a collating sequence as input to a CONVERT process, CONVERT invokes NCS to fetch the collating sequence named in the FDL file from the local system's NCS library. For example:

```
$ CONVERT/FDL=filename input-index-file output-index-file
```

If the collating sequence named in the input indexed file and the collating sequence found in the NCS library have the same name but are different, the output file may be sorted improperly. Typically, this might happen when the input indexed file is created on one system and is later transported to another system that already has a collating sequence with the same name.

You can avoid this by renaming collating sequences that have conflicting names.

CONVERT Description

The Convert Utility (CONVERT) copies records from one or more source data files to a second output data file, which can differ in file organization and format from the first.

You can also use CONVERT to reformat an indexed file in which you have deleted or inserted many records. The file specification of the indexed file is used as both the input and the output file specification. In this case, the output file has a version number one greater than the highest existing version of the input file. During such reorganizations, CONVERT establishes new record file addresses (RFAs) for the records.

On the other hand, the Convert/Reclaim Utility (CONVERT/RECLAIM) finds empty buckets in a Prolog 3 indexed file and makes them usable again by reclaiming them. As opposed to CONVERT, CONVERT/RECLAIM preserves existing RFAs for the file. In general, CONVERT, rather than CONVERT/RECLAIM, produces the most efficient indexed file reorganizations.

A set of library routines can also perform the functions of both the Convert and Convert/Reclaim Utilities from within a program. For more information, refer to the CONVERT routines in the *VMS Utility Routines Manual.*

Sections 1 through 5 cover the following topics. Section 1 explains how to produce output files with CONVERT. Section 2 describes converting between carriage control formats. Section 3 discusses using CONVERT with DECnet–VAX. Section 4 explains how to handle CONVERT exception conditions. Section 5 explains how to use the Convert/Reclaim Utility.

1 Output Files

There are two ways to generate a reformatted and reorganized output file. CONVERT can either create an output file or load an existing one.

If you want to create an output file with characteristics different from the input file, you specify a File Definition Language (FDL) file specification in the command line. To create an output file with the same characteristics as the first input file, you omit the /FDL qualifier.

If the output file exists, it can either be empty or it can contain records. If the output file is sequential, then specifying the /APPEND qualifier causes new records from the input file to be added sequentially to the end of the output file.

However, if the existing output file is indexed and contains records, then specifying the /MERGE qualifier causes new records from the input file to be merged in their proper order.

In addition, sorting the records from an input indexed file can be costly in terms of processing time and disk space. If the records in the input file are already sequentially ordered by the primary key (KEY=0), then specify the /NOSORT qualifier with the CONVERT command. For more information about sorting indexed files, see both the /FAST_LOAD and the /SORT qualifiers in the CONVERT Qualifiers section.

2 Converting Carriage Control Formats

A file can have one of four carriage control formats:

- CARRIAGE_RETURN
- FORTRAN
- PRINT
- NONE

These formats are all represented differently, so when you are converting a file from one carriage control format to another, the carriage control information has to be translated.

This translation is especially important when you are converting to or from a file with the FORTRAN format. Records with the FORTRAN format contain one byte of carriage control information at the beginning of each record.

For most conversions, the FORTRAN carriage control information is preserved as the first data byte of the record, and the printing characteristics are lost. However, certain conversions can preserve the printing characteristics of the FORTRAN carriage control information. When FORTRAN carriage control is converted to the equivalent PRINT carriage control, the information preceding each FORTRAN record is changed but not lost.

When PRINT carriage control is converted to FORTRAN carriage control, certain characters that supply carriage control information to the printer cannot be translated exactly. These untranslatable characters are represented as a single-spaced FORTRAN record.

When FORTRAN carriage control is converted to STREAM, control characters affecting carriage returns (<CR>), line feeds (<LF>), and form feeds (<FF>) are prefixed and appended to each FORTRAN record. These characters may affect the STREAM output because they are considered record delimiters for stream files. As a result, you may have a different number of records in the STREAM output file, and some of the records may be null.

The following chart shows how FORTRAN carriage control information translates to STREAM.

		STREAM Format Equivalent	
FORTRAN Format	**Hex Code Equivalent**	**Characters Prefixed**	**Characters Appended**
1	31	<FF>	<CR>
0	30	<LF> <LF>	<CR>
space	20	<LF>	<CR>
$	24	<LF>	Nothing appended
+	2B	Nothing prefixed	<CR>
null	00	Nothing prefixed	Nothing appended

All other conversions *from* FORTRAN preserve the carriage control information as data. All other conversions *to* FORTRAN prefix the converted records with the ASCII space character to obtain single spacing.

For more information about carriage control, see the description of the FDL Facility in the *VMS File Definition Language Facility Manual.*

3 Using CONVERT with DECnet–VAX

You can use the CONVERT command to transfer files to and from a remote node, either with or without modifying file attributes. If the output file exists, the Convert Utility changes the organization and format of the input data file to that of the output file. If the output file does not exist, CONVERT creates it from the file attributes specified in an FDL file.

You can also use the Convert Utility to copy files to or from a remote node without modifying file attributes. CONVERT transfers the file record by record, just as it does on a single node. However, you must have NETMBX privilege to execute CONVERT over a network.

Similarly, you can use CONVERT/RECLAIM to reclaim empty buckets in Prolog 3 indexed files from a remote node.

4 Exception Conditions

Certain conversions cause exception conditions. An exception condition occurs when a record from the input file cannot be placed in the output file because of some format incompatibility. CONVERT sends a warning error message to SYS$ERROR upon encountering a record that causes an exception condition.

For example, an exception condition occurs when the length of the input records exceeds the length you specified for fixed-length output records. You can avoid this exception condition by specifying the /TRUNCATE qualifier. Converting short fixed-length records into longer fixed-length records also causes an exception. To avoid this exception condition, use the /PAD qualifier to fill in the output records. The /PAD qualifier allows you to specify your choice of pad character.

To keep a copy of the exception records, you create an exceptions file with the /EXCEPTIONS_FILE qualifier. The exceptions file is a sequential file with variable-length records; it receives a copy of any record that cannot be placed in the output data file. Exceptions files have the file type EXC, by default.

5 Using the Convert/Reclaim Utility

Unlike CONVERT, which changes the organization and record format of a file, CONVERT/RECLAIM reclaims empty buckets in an existing Prolog 3 indexed file. The organization and record format of the file are not changed.

When you delete all the records in a bucket, the bucket still retains its position within the database because it has a certain range of primary key values associated with it. When you write new records to the file, those records whose primary key falls within that range are written to that bucket.

CONVERT Description

If your application has buckets with records that do not use a primary key left over from a deleted record, empty buckets cannot be reused unless you reclaim them. To reclaim a bucket, CONVERT/RECLAIM deletes the old pointers to it and puts it on a list of free buckets. When an application adds records and needs a bucket, VMS RMS goes to the free bucket list and sets up pointers to a bucket from the list. By reclaiming buckets, you can often avoid extending the file, which causes inefficient processing.

In addition, CONVERT/RECLAIM preserves RFA (record file address) access to the file.

You cannot use CONVERT/RECLAIM on Prolog 1 or Prolog 2 indexed files. To reclaim empty buckets in a Prolog 1 or Prolog 2 indexed file, you must first reorganize the file by using the Convert Utility. This reorganization creates a new version of the file. However, unlike CONVERT/RECLAIM, CONVERT establishes new RFAs for the records.

To invoke the CONVERT/RECLAIM functions from within a program, use the CONV$RECLAIM routine. For more information, refer to the *VMS Utility Routines Manual*.

CONVERT Usage Summary

The Convert Utility (CONVERT) copies records from one or more files to an output file, changing the record format and file organization to those of the output file.

The Convert/Reclaim Utility (CONVERT/RECLAIM) reclaims empty buckets in Prolog 3 indexed files so that new records can be written in those buckets. It does not change the record format and file organization of these files.

FORMAT

CONVERT *input-filespec[,...] output-filespec*

PARAMETERS

input-filespec [,...]

Specifies the file or files to be converted. The specifications cannot contain wildcard characters. Multiple input files are concatenated to form a single output file. You may specify up to 10 input files.

output-filespec

Specifies the file into which the converted records are to be written. If the file specification does not include a file type, CONVERT assigns the output file the file type of the first input file.

FORMAT

CONVERT/RECLAIM *filespec*

PARAMETER

filespec

Specifies the Prolog 3 indexed file in which you want to reclaim buckets. When you use the CONVERT/RECLAIM command, the file cannot be opened for shared access.

usage summary

Invoke the Convert Utility by typing the CONVERT command at the DCL level.

Likewise, invoke the Convert/Reclaim Utility by typing the CONVERT/RECLAIM command at the DCL level. Exit both the Convert and the Convert/Reclaim Utilities by letting the utility run to successful completion.

Output from the Convert Utility is directed to the file you indicate with the **output-filespec** parameter. For more information, see Section 1.

The Convert/Reclaim Utility, however, produces no output unless you specify the /STATISTICS command qualifier. If you want to execute either CONVERT or CONVERT/RECLAIM over a network, you need NETMBX privilege.

CONVERT QUALIFIERS

This section describes the CONVERT qualifiers that you use to select the organization and format of your output file.

/APPEND

Controls whether records are to be appended to the end of an existing sequential file.

FORMAT

/APPEND
/NOAPPEND

PARAMETERS

None.

DESCRIPTION

The /APPEND qualifier is useful when you want to attach one file to the end of another. When the file's organization is relative or indexed, this qualifier is ignored.

The default is /NOAPPEND. You should use this option when you are loading records into a sequential file that already contains records, or when you are creating a new sequential file.

If you specify both /APPEND and /CREATE, /APPEND overrides the /CREATE qualifier.

EXAMPLE

```
$ CONVERT/APPEND N_Z_FILE.DAT A_M_FILE.DAT
```

This command causes the sequential input file N_Z_FILE.DAT to be attached to the end of the sequential file A_M_FILE.DAT.

/CREATE

Determines whether CONVERT creates a file or uses an existing file for output.

FORMAT

/CREATE
/NOCREATE

PARAMETERS

None.

DESCRIPTION

The /CREATE qualifier causes CONVERT to create an output file instead of using an existing file for output.

If the output file is to have different characteristics from the input file, you must also specify the /FDL qualifier. To create an output file with the same characteristics as the input file, omit the /FDL qualifier.

The /NOCREATE qualifier causes CONVERT to use an existing file for output. You would use this option, for instance, to load records into a data file that you created previously with the Create/FDL Utility.

The default is /CREATE.

EXAMPLES

1
```
$ CONVERT/CREATE OLDFILE.DAT NEWFILE.DAT
```

This command creates the new output file NEWFILE.DAT and loads it with the records from OLDFILE.DAT.

2
```
$ CONVERT/CREATE/FDL=UPDATE.FDL OLDFILE.DAT NEWFILE.DAT
```

This command creates the new output file NEWFILE.DAT and loads it with the OLDFILE.DAT records that have been reformatted according to the characteristics in the FDL file UPDATE.

/EXCEPTIONS_FILE

Specifies whether an exceptions file is to be generated during the conversion.

FORMAT

/EXCEPTIONS_FILE *[=filespec]*
/NOEXCEPTIONS_FILE

QUALIFIER VALUE

filespec

Specifies the output file into which you want the exception records to be returned. If you specify /EXCEPTIONS_FILE and omit the **filespec** parameter, the exception records are displayed to SYS$OUTPUT.

The default file type for the exceptions file is EXC. The default is /NOEXCEPTIONS.

EXAMPLE

```
$ CONVERT/EXCEPTIONS_FILE=EXFILE.EXC/FDL=NEWFILE.FDL OLDFILE.DAT NEWFILE.DAT
```

This command loads the records from OLDFILE.DAT into NEWFILE.DAT and writes any records that cause exceptions into the file EXFILE.EXC.

/EXIT

Controls whether CONVERT is to exit when it encounters an exception record. The default operation is to continue processing records.

FORMAT

/EXIT
/NOEXIT

PARAMETERS

None.

EXAMPLE

```
$ CONVERT/FDL=NEWFILE.FDL/EXIT OLDFILE.DAT NEWFILE
```

This command loads the records from OLDFILE.DAT into NEWFILE.DAT and causes CONVERT to exit if an exception record is processed. Because no output file type is specified, CONVERT assigns the output file the same file type as the input file.

/FAST_LOAD

Specifies whether a fast loading algorithm is to be used for indexed files.

FORMAT

/FAST_LOAD
/NOFAST_LOAD

PARAMETERS

None.

DESCRIPTION

The /FAST_LOAD qualifier is one of the most useful features of the Convert Utility. The /FAST_LOAD qualifier and the /NOFAST_LOAD qualifier both sort primary keys, and both qualifiers require multiple scratch disk files.

Essentially, the difference between the /NOFAST_LOAD option and the /FAST_LOAD option is the way records are inserted into an indexed file. The /NOFAST_LOAD qualifier uses the normal VMS RMS Put service to load each record; VMS RMS updates the indexes of both the primary and secondary (alternate) keys as each record is inserted.

The main disadvantage of using the /NOFAST_LOAD option is the slower system performance that results from bucket splits and updates to the index. As each primary key is inserted, any secondary keys for that record are inserted in the order of the primary key. In other words, the secondary keys are not inserted in order of their own keys. These unsorted secondary keys may eventually cause bucket splits; as a result, the index structure for the secondary keys may be less efficient.

The advantage of the /NOFAST_LOAD option is that CONVERT does not attempt to sort secondary keys. Conversely, if you specify the /FAST_LOAD option, CONVERT sorts the primary and the secondary keys.

CONVERT processes a file as follows:

1. The primary keys are sorted. If the input file is on magnetic tape or if you specify multiple input files, the sort work file contains the sorted records. If the input file is on a disk, however, the sort work file contains only pointers to the sorted records.

Note: If your input records are already ordered by the primary key or if the primary key of the input and output indexed files is the same, you should specify /NOSORT. This qualifier ensures that the primary keys are not sorted again. For more information about sorting, see the description of the /SORT qualifier.

2. CONVERT builds the primary data record level from the sorted output file. CONVERT completely fills a bucket with data before it creates the lowest primary index level (the level 1 index). When an index bucket is filled, CONVERT creates an index record in the next highest index level.

3 When CONVERT is finished with the primary key, it updates the associated KEY DESCRIPTOR in the file's prolog, closes any input files, deletes any temporary files, and closes the output file. At this point, CONVERT has created a valid output file with records ordered by the primary key. If you specified no alternate keys, CONVERT is finished processing.

4 CONVERT reopens the input file so the first alternate key (if one has been specified) can be sorted. Again, CONVERT creates a temporary file containing pointers back to the primary data records. These pointers, however, have been sorted according to the first alternate key.

5 CONVERT loads these sorted pointers into the secondary index data record (SIDR) level and adjusts them to point to the records in the primary data level. Again, CONVERT completely fills a bucket with data before it creates the lowest secondary index level. When an index bucket is filled, CONVERT creates an index record in the next highest secondary index level.

6 When CONVERT is finished with this secondary key, it updates the associated KEY DESCRIPTOR in the file's prolog, closes any input files, deletes any temporary files, and closes the output file. At this point, CONVERT has created a valid output file, containing sorted primary keys and secondary keys. If you specified no more alternate keys, CONVERT is finished processing. However, if you specified additional alternate keys, they are processed in the same way.

The primary advantage of using the /FAST_LOAD option is that it is considerably faster than the VMS RMS method used by the /NOFAST_LOAD option. In most cases, you can increase processing speed by a factor of 10. Even greater speed results when you load large files with many keys.

In addition, the index structure can be very efficient because each key is sorted before it is loaded. The only disadvantage is the large amount of disk space needed for the work files. However, you can control the amount of disk space by using the /WORK_FILES qualifier and by reassigning the work files to different devices. See the /WORK_FILES qualifier for more information.

The default is /FAST_LOAD.

EXAMPLES

1
```
$ CONVERT/FAST_LOAD UPDATE.DAT MASTER.DAT
```

This command loads the records from the file UPDATE.DAT into the output file MASTER.DAT using the /FAST_LOAD option. CONVERT attains the added speed by building the indexes directly and then using VMS RMS for block I/O only.

2
```
$ CONVERT/NOFAST_LOAD UPDATE.DAT MASTER.DAT
```

This command loads the records from the file UPDATE.DAT into the output file MASTER.DAT. In this case, the operation takes longer because CONVERT uses VMS RMS Put services to output each individual record.

/FDL

Indicates that an FDL file is to be used in creating the output file.

FORMAT

/FDL=*fdl-filespec*

QUALIFIER VALUE

fdl-filespec
Specifies the FDL file to be used in creating the output file.

DESCRIPTION

The newly created output file will have the name specified by the **output-filespec** command parameter; this name overrides any file name specified in the FDL file.

The default file type for the FDL file is FDL.

EXAMPLE

```
$ CONVERT/FDL=INDEXFILE CUSTSEQ.DAT CUSTIND.DAT
```

This command creates the new file CUSTIND.DAT according to the specifications in the FDL file INDEXFILE.FDL. Records are then loaded from CUSTSEQ.DAT into CUSTIND.DAT.

/FILL_BUCKETS

Controls whether to override the bucket fill percentage parameter associated with the output file.

FORMAT

/FILL_BUCKETS
/NOFILL_BUCKETS

PARAMETERS

None.

DESCRIPTION

If you specify /FILL_BUCKETS, CONVERT fills the buckets of the output file with as many records as possible. This behavior is advantageous if you do not plan to do random file processing, because using fewer buckets saves disk space and processing time.

With /NOFILL_BUCKETS, however, CONVERT does not fill the buckets completely. Therefore, you can add records at a later date without splitting buckets or extending the file.

This option is valid only for indexed output files. The default is /NOFILL_BUCKETS.

EXAMPLE

```
$ CONVERT/FILL_BUCKETS SALES_DATA.DAT CUST_DATA.DAT
```

This command loads the records from the indexed file SALES_DATA.DAT into the indexed file CUST_DATA.DAT, filling the buckets of the output file with as many records as possible.

/FIXED_CONTROL

Controls the behavior of CONVERT in conversions between variable with fixed-length control (VFC) records and records of other formats.

FORMAT

/FIXED_CONTROL
/NOFIXED_CONTROL

PARAMETERS

None.

DESCRIPTION

If you specify /FIXED_CONTROL for VFC input records, then the fixed-control portion of the record is attached to the front of the output record.

If you specify /FIXED_CONTROL for VFC output records, then the fixed-control portion of the output record is taken from the front of the input record. If the input record is not long enough to fill the control portion of the output record, an exception record is generated.

If you specify /NOFIXED_CONTROL with VFC input records, then the fixed-portion of the input record is not copied to the output record.

If you specify /NOFIXED_CONTROL for VFC output records, then the fixed-portion of the output record is set to 0.

This option is not applicable to indexed files. The default is /NOFIXED_CONTROL.

When you use this qualifier, you must account for the size of the fixed-control area when you calculate the maximum size of the output record.

EXAMPLE

```
$ CONVERT/FIXED_CONTROL VFC_FILE.DAT OUTFILE.DAT
```

This command loads the VFC records in the input file VFC_FILE.DAT into the output file OUTFILE.DAT.

/KEY

Directs CONVERT to read records from an indexed file using a specified key of reference, such as the primary key, the first alternate key, or the second alternate key.

FORMAT

/KEY=*n*

QUALIFIER VALUE

n

A numeric value that specifies the key of reference CONVERT uses for reading records from the input indexed file. For example, you can specify the primary key as the key of reference by using the value *0* (/KEY=0), which is the default, or you can specify the first alternate key as the key of reference by using the value *1* (/KEY=1).

DESCRIPTION

The /KEY qualifier is valid for indexed input files only. If you use the /KEY qualifier, you must specify a key value (/KEY=0, /KEY=1, and so on). If you do not specify the /KEY qualifier, the default is the primary key (/KEY=0).

EXAMPLE

```
$ CONVERT/NOCREATE/KEY=1 CUST_INX.DAT CUST_SEQ.DAT
```

This command loads the records from the indexed input file CUST_INX.DAT into the sequential output file CUST_SEQ.DAT. The records in the output file are ordered by the first alternate key in the input file.

/MERGE

Specifies that records are to be inserted into their proper position in an existing indexed file.

FORMAT /MERGE

PARAMETERS *None.*

DESCRIPTION

The /MERGE qualifier is useful when your input records are not sorted, and you do not want them to be sorted as they are loaded into an output file.

If you specify both /MERGE and /CREATE, /MERGE overrides the /CREATE qualifier.

EXAMPLE

```
$ CONVERT/MERGE ACCOUNTS.DAT MASTER_INX.DAT
```

This command loads the records from the input file ACCOUNTS.DAT into the existing indexed output file MASTER_INX.DAT according to primary key values.

/PAD

Determines whether short records are to be padded.

FORMAT

/PAD *[=[%b]x]*
/NOPAD

PARAMETERS

%b

Represents the base of the numeric value specified by *x*. Values for *b* are as follows:

D Indicates that *x* is a decimal value.

O Indicates that *x* is an octal value.

X Indicates that *x* is a hexadecimal value.

If you specify *x* as a numeric value, it is interpreted in the base indicated by *%b*.

x

Specifies that the short records are to be padded with either ASCII characters (A through Z, a through z) or numeric values (0 through 9).

If you specify *x* as an ASCII character, you do not need to specify *%b*. However, if you specify *x* as a numeric value, you must specify the optional base with *%b*.

DESCRIPTION

The /PAD qualifier indicates that short records are to be padded with ASCII characters. A record is too short when it does not contain as many bytes as the record length for fixed-length record format.

If you specify /PAD, the default pad character is the ASCII null character (binary value 0). The /PAD option is valid only for fixed-output record formats.

The default is /NOPAD, which generates an exception record when a short record is encountered.

EXAMPLES

1
```
$ CONVERT/NOCREATE/PAD=%X20 INFILE.DAT OUTFILE
```

This command specifies that any short records in the input file INFILE.DAT are to be padded with an ASCII space character before being loaded into the fixed-length output file OUTFILE.DAT.

2
```
$ CONVERT/FDL=FIXED/PAD=X INFILE.VAR OUTFILE.FIX
```

This command creates the fixed format file OUTFILE.FIX and then loads it with records from the variable input file INFILE.VAR. Any short records from the input file are padded with an ASCII X before they are loaded.

/PROLOG

Specifies the prolog version number of the output indexed file.

FORMAT

/PROLOG=*n*

QUALIFIER VALUE

n

Specifies the prolog number 1, 2, or 3.

If you specify 2 for *n*, the output file will be either a Prolog 1 or Prolog 2 file.

If you specify 3, CONVERT creates a Prolog 3 file for output. Prolog 3 files accept multiple keys (or alternate keys), all data types, and segmented keys. The only restriction on using a Prolog 3 file applies to files containing overlapping key segments for the primary key. In this case, you would have to use a Prolog 2 file.

DESCRIPTION

If you do not specify the /PROLOG qualifier, CONVERT uses the prolog version of the first input file. If the input file is not indexed, CONVERT uses the VMS RMS default. To see what this default is on your system, enter the DCL command SHOW RMS_DEFAULT.

The /PROLOG qualifier overrides the value given with the FDL attribute KEY PROLOG.

EXAMPLE

```
$ CONVERT/PROLOG=3 INFILE_2 OUTFILE_3
```

This command loads the records from the Prolog 2 input file INFILE_2 into the Prolog 3 output file OUTFILE_3. Both the input and output file are indexed files.

/READ_CHECK

Specifies whether each input record is to be read from the file a second time and compared to the record originally read.

The default is /NOREAD_CHECK.

FORMAT

/READ_CHECK
/NOREAD_CHECK

PARAMETERS

None.

EXAMPLE

```
$ CONVERT/READ_CHECK Q3_SALES.DAT YTD_SALES.DAT
```

This command specifies that the records from the input file Q3_SALES.DAT are to be read and checked by the file processor, and then loaded into the output file YTD_SALES.DAT.

/SHARE

Specifies whether the input file is to be opened for sharing with other processes.

FORMAT

/SHARE
/NOSHARE

PARAMETERS

None.

DESCRIPTION

You can use the /SHARE option to generate a rough backup of a file that is always opened for sharing by some applications. However, another process can alter the records at the same time CONVERT is operating. As a result, the consistency of the output file cannot be guaranteed.

EXAMPLE

```
$ CONVERT/SHARE SYSUAF.DAT BACKUP.DAT
```

This command indicates that the input file SYSUAF.DAT is open for sharing with other processes at the same time its records are being loaded into the output file BACKUP.DAT.

/SORT

Specifies whether the input file is to be sorted before being loaded into an indexed file. The sort is done according to the primary key of the output file.

FORMAT

/SORT
/NOSORT

PARAMETERS

None.

DESCRIPTION

Two procedures can improve the sort performance:

- Increasing the size of the working set for the duration of the sort. The general rule is to use as large a working set as allowed by your working set quota. To set this value, use the DCL command SET WORKING_SET. To see what your authorized quota is, enter the SHOW WORKING_SET command.
- Placing the input file, the output file, and the temporary work files on separate disk devices. The default operation is to place the work files on your default device, which could cause CONVERT to run out of disk space. To specify the location of the work files, enter a command in the following form:

  ```
  $ ASSIGN device-name: SORTWORKn
  ```

  The *n* represents the number of the work file, from 0 to 9. The colon is required after the device name. For example, the following two ASSIGN commands would place the work files on disks named TMPD and DEVD:

  ```
  $ ASSIGN TMPD: SORTWORK0
  $ ASSIGN DEVD: SORTWORK1
  ```

  Using more than two work files (the default) is not particularly advantageous unless you have to use many smaller ones in order to fit on crowded disks. You can control the number of work files with the /WORK_FILES qualifier.

  Also, when CONVERT uses SORT32, it may open up to 13 files. If your process open-file limit is reached or if VMS RMS runs out of dynamic memory, SORT32 may fail to open a necessary temporary file.

The default is /SORT. Also, this option is ignored if the output file is not indexed.

For more information about using SORT32 with CONVERT, see the /FAST_LOAD qualifier.

EXAMPLES

1 `$ CONVERT/SORT IN_INX.DAT OUT_INX.DAT`

This command causes the records in the input indexed file IN_INX.DAT to be sorted according to the primary key values before being loaded into the output indexed file OUT_INX.DAT.

2 `$ CONVERT/NOSORT/FDL=REORG INX.DAT INX.DAT`

This command reorganizes the file INX.DAT according to the attributes specified in the FDL file REORG.FDL. The primary keys are not sorted because INX.DAT is already ordered by the primary key, and the primary key definition did not change.

/STATISTICS

Determines whether a set of statistics about the completed conversion is to be displayed. You can use this qualifier with both the CONVERT and the CONVERT/RECLAIM commands.

FORMAT

/STATISTICS
/NOSTATISTICS

PARAMETERS

None.

DESCRIPTION

Both CONVERT and CONVERT/RECLAIM have a /STATISTICS command qualifier. The statistics produced by the Convert Utility are as follows:

- Number of files processed
- Total records processed
- Total exception records
- Total valid records
- Elapsed time
- Buffered I/O count
- Direct I/O count
- Page faults
- CPU time

The statistics produced by the Convert/Reclaim Utility are as follows:

- Total buckets scanned
- Data buckets reclaimed
- Index buckets reclaimed
- Total buckets reclaimed
- Elapsed time
- CPU time

CONVERT/RECLAIM produces no output unless you specify this qualifier.

EXAMPLES

1 `$ CONVERT/STATISTICS Q3_SALES.DAT YTD_SALES.DAT`

This command causes CONVERT to load the records from the input file Q3_SALES.DAT into the output file YTD_SALES.DAT and then to display a set of statistics about the conversion.

2 `$ CONVERT/RECLAIM/STATISTICS CUSTDATA.IDX`

This command causes CONVERT/RECLAIM to reclaim buckets in the indexed file CUSTDATA.IDX and then to display statistics about the reclamation.

/TRUNCATE

Specifies whether long records are to be truncated.

FORMAT

/TRUNCATE
/NOTRUNCATE

PARAMETERS

None.

DESCRIPTION

A record is too long when it exceeds the maximum record length of the file or the record length for fixed-length record format.

If you specify /NOTRUNCATE and a long record is encountered, the record is not written to the output file. If you specify the /EXCEPTIONS_FILE qualifier, the entire record is written to the exceptions file.

EXAMPLES

1 `$ CONVERT/TRUNCATE INFILE.DAT OUTFILE.DAT`

This command causes CONVERT to truncate any records from the input file INFILE.DAT that are too long for the specifications of the output file OUTFILE.DAT.

2 `$ CONVERT/NOTRUNCATE/EXCEPTIONS_FILE=EXFILE INFILE OUTFILE`

This command causes CONVERT to write any records from the input file INFILE that are too long for the specifications of the output file OUTFILE to the exceptions file EXFILE.

/WORK_FILES

Specifies the number of temporary work files to be used during the sort process.

FORMAT

/WORK_FILES=*n*

QUALIFIER VALUE

n

Specifies the number of work files you want. You can specify 0 or any value from 1 through 10. A value of 0 indicates that no work files are necessary because the data will fit into the working set of your process.

The default number of work files used during a sort is 2.

DESCRIPTION

This qualifier is valid when you are fast-loading a file with multiple keys or when you specify the /SORT qualifier. For more information about sorting, see both the /SORT and the /FAST_LOAD qualifiers.

EXAMPLE

```
$ CONVERT/WORK_FILES=0 UPDATE.DAT MASTER.DAT
```

This command loads the records from the input file UPDATE.DAT into the output file MASTER.DAT without using any work files.

/WRITE_CHECK

Specifies whether all writes are to be checked by reading the new records from the disk and comparing the new records with the original records in memory.

FORMAT

/WRITE_CHECK
/NOWRITE_CHECK

PARAMETERS

None.

EXAMPLE

```
$ CONVERT/WRITE_CHECK UPDATE.DAT MASTER.DAT
```

This command causes CONVERT to load the records from the input file UPDATE.DAT into the output file MASTER.DAT and then to compare the written records with the original records for accuracy.

CONVERT
EXAMPLES

1 `$ CONVERT/NOCREATE/TRUNCATE/EXCEPTIONS_FILE=EXFILE VARFILE.DAT FIXFILE.DAT`

This command causes CONVERT to copy records from a file with variable-length records (VARFILE.DAT) to a file with fixed-length records (FIXFILE.DAT). Records longer than the fixed length are truncated, and short records are copied to the exceptions file EXFILE.EXC.

2 `$ CONVERT FILE.IDX FILE.IDX`

This command creates the output file FILE.IDX with a version number one higher than that of the input file. The output file is a copy of the input file, but it is a clean copy without bucket splits, RRVs (record reference vectors), or pointers to deleted records. The performance of the output file is also improved.

Note that CONVERT establishes new RFAs during such reorganizations.

3 `$ CONVERT/RECLAIM/STATISTICS FILE.IDX`

This command reclaims empty buckets in the file FILE.IDX. The RFA values are preserved.

4 `$ CONVERT/FDL=TEST.FDL TRNTO::DBA1:[EXP]SUB.DAT OUT.DAT`

This command creates a new sequential file OUT.DAT with stream record format at the local node, according to the specification in the previously created FDL file TEST.FDL. The input file SUB.DAT at remote node TRNTO is sequential with variable-length record format. The Convert Utility copies records from SUB.DAT to OUT.DAT, changing the format of the records.

The contents of the FDL file TEST.FDL are as follows:

```
SYSTEM
        SOURCE              VAX/VMS

FILE
        ORGANIZATION        SEQUENTIAL

RECORD
        BLOCK_SPAN          YES
        CARRIAGE_CONTROL    CARRIAGE_RETURN
        FORMAT              STREAM
        SIZE                0
```

5 `$ CONVERT MASTER.DAT DENVER::DB1:[PROD]MASTER.SAV`

This command creates a new file called MASTER.SAV at remote node DENVER from the file MASTER.DAT at the local node. Because the /FDL qualifier is not used, the new file has the same file organization and record format as the original file. The action of this CONVERT command is similar to the function performed by the COPY command. However, CONVERT transfers the file record by record and thus does not use block I/O.

6
```
$ CONVERT/APPEND SALES.TMP KANSAS::[200,2]SALES.CMD
```

This command causes records from the file SALES.TMP at the local node to be added sequentially to the end of the output file SALES.CMD at remote node KANSAS. The file SALES.TMP is sequential with variable-length record format, and the file SALES.CMD is sequential with stream record format. When the Convert Utility loads records from the input file to the output file, it changes the record format.

7
```
$ CONVERT/FDL=FIXED/PAD=0/TRUNCATE INFILE.VAR OUTFILE.FIX
```

This command creates the fixed format file OUTFILE.FIX and then loads it with records from the variable input file INFILE.VAR. Before they are loaded, any short records from the input file are padded with an ASCII 0 character, and any long records are truncated.

8
```
$ CONVERT/FDL=SYS$INPUT FORT.DAT STREAM.DAT
FILE
        ORGANIZATION            SEQUENTIAL

RECORD
        CARRIAGE_CONTROL        CARRIAGE_RETURN
        FORMAT                  STREAM
CTRL/Z
```

This command converts the FORTRAN carriage control file FORT.DAT to a stream file that prints or types identically. The number of records may differ, and the FORTRAN carriage control information is removed from the records.

9
```
$ CONVERT/FDL=SYS$INPUT FORT.DAT VAR.DAT
FILE
        ORGANIZATION            SEQUENTIAL

RECORD
        CARRIAGE_CONTROL        CARRIAGE_RETURN
        FORMAT                  VARIABLE
CTRL/Z
```

This command converts the FORTRAN carriage control file FORT.DAT to a variable-length record file. The FORTRAN carriage control information is preserved as the first data byte, and the number of records in the output and input files is the same.

Index

A

B

C

D

E

F

I

K

L

M

N

O

P

Q

R

S

T

V

W

Reader's Comments

VMS Convert and Convert/Reclaim Utility Manual
AA–LA80A–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