Article
Stuart Salzer · Nov 10, 2016 17m read

Yes, you can use RMS file on OpenVMS

This article explains the basics of OpenVMS file structures, from a Caché prospective, and what you need to know to read and write any OpenVMS file with Caché, even the files with difficult file structures.

On UNIX and Windows all files are just a stream of octets (eight-bit bytes). Assigning meaning to those octets is done entirely by convention. Those conventions are rather simple, even if slightly different on UNIX and Windows. Meanwhile, the authors of OpenVMS thought that file structure was so important, they built it into the Operating System in the form of the Record Management Services (RMS). While RMS does provide for file structures that are almost 100% compatible with UNIX and Windows, most native OpenVMS applications prefer file structures that are more efficient.

There is a good chance you have encountered a file that appears to contain valid commands for some program, but nevertheless produces errors. That file almost certainly contained invisible characters (humorously called gremlins). This is true on OpenVMS as it is on UNIX and Windows, but on OpenVMS files can not only have characters that you can ' t see, files can have structure you can ' t see. Actually, you can see both if you know how.

The first important difference between OpenVMS and those other operating systems is that on OpenVMS, a file is usually a collection of records, and each of those records is a (usually much smaller) stream of octets. These records can have a fixed or variable length, (or even a combination of the two). The records can also have a sequence, but not necessarily a single sequence based upon how the records were written to the file. The file can also define a method for inserting line termination characters between records that may or may not be stored in the records, so that one can reconstruct a stream from the record oriented file.

Consider that you have an OpenVMS file that you wish to read. In general, the same read commands will read any file. There are some exotic file organizations, for which you may want to do special processing, or maybe not, but you should at least know how to identify one of those files. Here is a useful command given against any file:

```
$ DIR/FULL filename.ext
…
File organization:  Indexed, Prolog: 3, Using 1 key
Shelved state:      Online
Caching attribute:  Writethrough
File attributes:    Allocation: 16, Extend: 0, Maximum bucket size: 2, Global buffer count: 0, No version limit
Record format:      Fixed length 132 byte records
Record attributes:  Fortran carriage control
…
```

The important output appears in the middle. This tells us the file is indexed, that means that now matter what order you write records to the file, they will later read back sorted based upon key columns (that are also part of the file definition, but not shown here). We are also told that the file has fixed 132 bytes records. That means that you have to write 132 byte records to the file, other sizes don ' t work. It also tells us the file has Fortran carriage control. Unlike the Indexed file organization, and enforcement of fixed 132 byte records, you are responsible for dealing with record attributes like Fortran carriage control.

Whatever you do, don ' t try to parse the output from DIR/FULL. There is a better option. Parse the output from ANALYZE/RMS/FDL. This creates a temporary file in File Definition Language (FDL), an easy to parse language OpenVMS uses to describe file structures [1]. The code below is incomplete, and finishing it is left as an exercise for the reader. Specifically wrong, is that it has no real error handling, and it doesn ' t properly handle multiple numbered areas, keys, and so on.

```
; Format
;  DO FDL^routine(file,.fdl)
```

```
;
; Arguments
;  file The name of a file, whose structure you wish to retrieve.
;  fdl (passed by reference), An array in which to store the parsed structure of file.
;
FDL(file,fdl)   []
  {
  SET tmp=##CLASS(%File).TempFilename("FDL")
  DO $ZF(-1,"ANALYZE/RMS/FDL/OUTPUT="tmp"_"file)
  OPEN tmp:"R":1 IF '$TEST { QUIT }
  SET io=$IO USE tmp
  TRY {
    FOR  {
      READ x CONTINUE:x=""
      SET x=$TRANSLATE(x,$CHAR(9)," ")_"
      IF $EXTRACT(x)=" " {
        SET p=$FIND(x," ",2),key2=$EXTRACT(x,2,p-2)
        SET fdl(key1,key2)=$ZSTRIP($EXTRACT(x,p,*),"<>W")
      } ELSE {
        SET p=$FIND(x," "),key1=$EXTRACT(x,1,p-2)
        SET fdl(key1)=$ZSTRIP($EXTRACT(x,p,*),"<>W")
      }
    }
  } CATCH err { }
  CLOSE tmp USE io
  DO ##CLASS(%File).Delete(tmp)
  QUIT
  }
```

The entries that are important to understanding how to read any file are in:

- fdl("FILE","ORGANIZATION")
- fdl("RECORD","BLOCKSPAN")
- fdl("RECORD","CARRIAGECONTROL")
- fdl("RECORD","FORMAT")
- fdl("RECORD","SIZE")

Of these, only fdl("RECORD","FORMAT") is available with a Caché method.

If you want to read a file as a sequence of records, all you need to do is this:

```
  OPEN file:"R"
  USE file
  TRY {
    FOR {
      READ x
      ; Process record
    }
  } CATCH err { }
```

Of course you may want better error handling. You can also read with object methods like this:

```
  SET stream=##CLASS(%File).%New(file)
  DO stream.Open("R")
  WHILE 'stream.AtEnd {
    SET x=stream.ReadLine()
    WRITE x,!
  }
```

Unfortunately, the stream methods have not been tested with exotic file structures, and in many cases do not work.

So for this article we will stick with traditional OPEN, USE, READ, and WRITE, as documented in [2].

The important attribute that affects how you process the records is fdl("RECORD","CARRIAGE_CONTROL"). This can have one of four values:

- none
- carriage_return
- fortran
- print

The common values are "none" and "carriage_return". If the value is "none", the file is as read. If the value is "carriage_return", there is an implicit carriage return line feed $CHAR(13,10) following each record. If your trying to read a file as records, this is an unimportant distinction, but there are occasional files that are marked " none " because their records contain carriage control characters, that Caché doesn ' t strip for you. In fact, if you have applications that use Open mode "WNV", and you use WRITE !, you will get a file with hidden structure, and records that contain only $CHAR(13,10). This was more common before "WNM" was implemented in Caché 5.0 [RFD035]

"fortran" is somewhat exotic, and you would not be far off to treat them just like "carriage_return", but if you want to be correct for "fortran", you should remove the first character from each record and use it to look up the appropriate carriage control.

Character	Prefix record with	Postfix record with
$CHAR(0)	Nothing	Nothing
"0"	$CHAR(10,10)[1]	$CHAR(13)[3]
"1"	$CHAR(12)	$CHAR(13)[3]
"+"	Nothing	$CHAR(13)[3]
$"	$CHAR(10)[2]	Nothing
" ", or any other character	$CHAR(10)[2]	$CHAR(13)[3]

[1] For the first record of the file use $CHAR(13,10) instead.
[2] For the first record of the file user nothing instead.
[3] For the last record of the file use $CHAR(13,10) instead.

The good news is that if you don't actually have any FORTRAN code, or attempting to interact with any third party FORTRAN code, you will probably never see, or have to deal with a "fortran" carriage control file.

Not so with "print" carriage control. While most commonly associated with COBOL programs, OpenVMS batch log also use "print" carriage control. These files have two octet associated with each variable length record, and these two octet describe the carriage control to prefix and postfix each record. When RMS reads one of these records for Caché, these carriage control octets are stripped. Thus, simply ignoring the carriage control is generally the easiest option. If you can ' t ignore the missing carriage control, you have two choices. You can first convert the file to another file, and read that file:

```
SET tmp=##CLASS(%File).TempFilename("TMP")
SET fdlf=##CLASS(%File).TempFilename("FDL")
OPEN fldf:"WNM"
```

```
  WRITE "FILE",!
  WRITE $CHAR(9),"ORGANIZATION",$CHAR(9),"sequential",!!
  WRITE "RECORD",!
  WRITE $CHAR(9),"CARRIAGECONTROL",$CHAR(9),"carriagereturn",!
  WRITE $CHAR(9),"FORMAT",$CHAR(9),"variable",!
  WRITE $CHAR(9),"SIZE",$CHAR(9),reclen,!
  CLOSE fldf
  DO $ZF(-1,"CONVERT "file_ "tmp"/FDL="fdlf"/FIX")
  DO ##CLASS(%File).Delete(fdlf)
  OPEN tmp:"R"
  USE tmp
  TRY {
    FOR  {
      READ x
      SET x=$$CC($ASCII(x,1))_$EXTRACT(x,3,*)_$$CC($ASCII(x,2))
      ; Process record x
    }
  } CATCH err { }
  ...
CC(code)   []
  {
  IF code=0 { QUIT "" }
  IF code<128 { QUIT $TR($J("",code)," ",$C(10))_$C(13) }
  IF code<160 { QUIT $CHAR(code-128) }
  IF code<192 { QUIT "" } ; Reserved
  IF code<208 { QUIT $TR($J("",code-191)," ",$C(11))_$C(13) } ; VFU
  QUIT "" ; Reserved
  }
  ; Warning: Vertical Format Unit (VFU) codes do not translate
  ; unambiguously into ASCII, the example code is a guess.
```

Another option is to read the file bypassing RMS, with

```
  OPEN file:"RU"
  TRY {
    FOR  {
      USE file
      READ len#2 SET len=$ZWASCII(len) QUIT:len<0
      READ x#len SET prefix=$ASCII(x,1),postfix=$ASCII(x,2)
      IF len#2 { READ junk#1 }
      USE $PRINCIPAL
      SET x=$CC(prefix)_$EXTRACT(x,3,*)_$$CC(postfix)
      ; Process record x
    }
  } CATCH err { }
  ; Warning: The above code does not properly handle the
  ; case where len=0, because READ x#0 will result in an
  ; error. Fixing that is left as an exercise for the reader.
```

That covers reading all RMS file structures. Writing is more complicated, because Caché can only natively write a few different file structures. If you need any other file structure, you will have to do a little extra work. Native support is based upon the Caché OPEN mode:

Mode	WRITE by	File Structure (all are ORGANIZATON sequential)
"WNS"	String[1]	CARRIAGECONTROL carriagereturn FORMAT stream SIZE 32767

Mode	WRITE by	File Structure (all are ORGANIZATON sequential)
"WNV"	Record[2]	CARRIAGECONTROL none FORMAT variable SIZE 32767
"WNM"	String[1]	CARRIAGECONTROL carriagereturn FORMAT variable SIZE 32767
"WNB"	Block[3]	CARRIAGECONTROL none FORMAT undefined SIZE 32256
"WNU"	String[1]	CARRIAGECONTROL none FORMAT undefined SIZE 32767

[1] WRITE x, appends x to the current output record. The record is not output until WRITE ! is executed.
[2] WRITE x, outputs a record of x.
[3] WRITE x, outputs $LENGTH(x)+511\512 blocks containing the data x padded with NULs as needed.

If you need to create a file with some other file attributes, there are four techniques you can use:

1. FDL create. In some cases Caché can WRITE to a file, but can ' t create a file with some specific attributes. In those cases it is usually easiest to let FDL create the file with the correct attributes, and then WRITE as needed.

2. Fix attributes. In some cases Caché, cannot WRITE to a file with specific attributes, but can WRITE to a file with similar attributes. In those cases, create the similar file, and fix the attributes, after the file has been written.

3. RMS bypass. In essence a level beyond the fix attributes method. In this case Caché can WRITE the underlying structure that RMS uses on disk, and then change the attributes to match the data. You should use this only for file attributes that have rather simple implementation.

4. Convert. In some cases, we can ' t write the file at all, but OpenVMS provides a Convert utility, so we can write the data to a temporary file, and allow convert to convert the data to the desired file structure.

The table that follows shows the best way to WRITE each RMS attributes combination. There are pros and cons to each method, so what constitutes best might be somewhat subjective. For the purpose of the table, the order in which the methods are presented, is considered best to worst, and if for cases that are arguable, multiple methods might be listed in the table. Finally, the Convert method is always possible.

CARRIAGECONTROL →		none	carriagereturn	fortran	print
↓ FORMAT	ORGANIZATION sequential				
fixed	sequential	FDL create RMS bypass	FDL create RMS bypass	RMS bypass	Unused combination
variable		"WNV" Native	"WNM" Native	Fix attributes RMS bypass	Unused combination
vfc		Unused combination	Unused combination	Unused combination	Fix attributes RMS bypass
stream		Fix attributes	"WNS" Native	Fix attributes	Unused combination
streamcr		RMS bypass	RMS bypass	RMS bypass	Unused combination
streamlf		RMS bypass	RMS bypass	RMS bypass	Unused combination
undefined		"WNU" Native	Unused combination	Unused combination	Unused combination
↓ FORMAT	ORGANIZATION relative				
fixed	relative	FDL create	FDL create	FDL create	Unused combination
variable		FDL create	FDL create	FDL create	Unused combination
vfc		Unused combination	Unused combination	Unused combination	FDL create
stream		Unused combination	Unused combination	Unused combination	Unused combination
streamcr		Unused combination	Unused combination	Unused combination	Unused combination
streamlf		Unused combination	Unused combination	Unused combination	Unused combination
undefined		Unused combination	Unused combination	Unused combination	Unused combination

CARRIAGECONTROL →		none	carriagereturn	fortran	print
↓ FORMAT	ORGANIZATION indexed				
fixed	indexed	Convert	Convert	Convert	Unused combination
variable		Convert	Convert	Convert	Unused combination
vfc		Unused combination	Unused combination	Unused combination	Unused combination
stream		Unused combination	Unused combination	Unused combination	Unused combination
streamcr		Unused combination	Unused combination	Unused combination	Unused combination
streamlf		Unused combination	Unused combination	Unused combination	Unused combination
undefined		Unused combination	Unused combination	Unused combination	Unused combination

Here are some more details of each method, along with an example.

1. FDL create

   In many cases, Caché can write to a file with particular attributes, but it can ' t create such a file. In that case, you can crate the file by first opening a temporary file, writing the FDL for the file, and then call $ZF(-1,"CREATE/FDL="fdl" "file)

```
; Sample code for
;
; FILE
;   ORGANIZATION relative
; RECORD
;   FORMAT fixed
;   CARRIAGECONTROL none
;   SIZE 80
;
SET reclen=80
SET pad=$JUSTIFY("",reclen)
OPEN fdl:"WNM"
USE fdl
WRITE "FILE",!
WRITE $CHAR(9),"ORGANIZATION",$CHAR(9),"RELATIVE",!
WRITE "RECORD",!
WRITE $CHAR(9),"FORMAT",$CHAR(9),"FIXED",!
```

```
  WRITE $CHAR(9),"CARRIAGECONTROL",$CHAR(9),"NONE",!
  WRITE $CHAR(9),"SIZE",$CHAR(9),reclen,!
  CLOSE fdl
  DO $ZF(-1,"CREATE/FDL="fdl" "file)
  DO ##CLASS(%File).Delete(fdl)
  OPEN file:"W"
  FOR  {
    ; Get record
    USE file WRITE $EXTRACT(recordpad,1,reclen) USE $PRINCIPAL
  }
  CLOSE file
```

2.
Fix attributes

Fix attributes is an option where Caché can natively WRITE to a file where only the CARRIAGECONTROL attribute is not what you want. The FORMAT attribute can also be changed from variable to vfc with this method. WRITE the file using similar attributes. Then change attributes by issuing:

```
DO $ZF(-1,"SET FILE/ATTRIB=(RFM:x,RAT:y) "file)
```

Where x, and y are the same as in RMS bypass (see below).

```
  ; Sample code for
  ;
  ; FILE
  ;   ORGANIZATION sequential
  ; RECORD
  ;   BLOCKSPAN yes
  ;   FORMAT stream
  ;   CARRIAGECONTROL none
  ;   SIZE 32767
  ;
  OPEN file:"WNS"
  FOR  {
    ; Get record and QUIT when there are no more.
    USE file WRITE record,! USE $PRINCIPAL
  }
  CLOSE file
  DO $ZF(-1,"SET FILE/ATTRIB=(RAT:NONE) "file)
```

3.
RMS bypass

RMS bypass should only be used for files where you know the structure of the file on disk, and it is relatively easy to record it. That means ORGANIZATION sequential files only. Essentially, OPEN the file with "WNU", and write the records with whatever structure is needed. CLOSE the file, and then issue:

```
DO $ZF(-1,"SET FILE/ATTRIB=(ORG:SEQ,RFM:x,RAT:y,LRL:reclen) "file)
```

Where x is the FORMAT, coded as:

FORMAT	RFM:	What to WRITE
fixed	FIX	WRITE record Required: $LENGTH(record)=reclen Recommended: reclen#2=0

FORMAT	RFM:	What to WRITE
		If CARRIAGECONTROL is fortran, Required: reclen#2=0 There is an option to include an additional attribute RAT:BLK (corresponding to FDL: BLOCKSPAN no), this option only applies if reclen<512, and if selected, after every 512\reclen records, WRITE $TRANSLATE($JUSTIFY("",512#reclen)," ",$CHAR(0))
variable	VAR	WRITE $ZWCHAR($LENGTH(record)) WRITE record WRITE:$LENGTH(record)#2 $CHAR(0)
vfc	VFC	WRITE $ZWCHAR($LENGTH(record)+2) WRITE $CHAR(prefix),$CHAR(postfix),record WRITE:$LENGTH(record)#2 $CHAR(0)
stream	STM	WRITE record,$CHAR(13,10) record must not contain $CHAR(13,10) It is never efficient to use bypass for this FORMAT. This FORMAT is intended to be compatible with Windows standard FORMAT.
streamcr	STMCR	WRITE record,$CHAR(13) record must not contain $CHAR(13) This FORMAT is intended to be compatible with the long disused macOS version 9 and earlier FORMAT.
streamlf	STMLF	WRITE record,$CHAR(10) record must not contain $CHAR(10)

FORMAT	RFM:	What to WRITE
		This FORMAT is intended to be compatible with UNIX.

Where y is the CARRIATECONTROL, coded as

CARRIAGECONTROL	RAT:	How to include carriage control
none	NONE	Intended for files with no implied carriage control.
carriagereturn	CR	Intended for files with implied $CHAR(13,10) after each record.
fortran	FTN	Prefix each record with a format control character. SPACE $CHAR(32) effectively implies $CHAR(13,10).
print	PRN	Prefix each record with two format control characters. $CHAR(1,141) are good defaults.

Where reclen is the length of fixed records, or the maximum length of variable records. OpenVMS imposes a maximum record length of 32767, or slightly less with certain file attributes.

```
; Sample code for
;
; FILE
;   ORGANIZATION sequential
; RECORD
;   BLOCKSPAN no
;   FORMAT fixed
;   CARRIAGECONTROL fortran
;   SIZE 132
;
SET reclen=132
SET pad=$JUSTIFY("",reclen)
OPEN file:"WNU"
FOR i=1:1 {
  ; Get record and QUIT when there are no more.
  USE file
  WRITE $EXTRACT(" "recordpad,1,reclen)
  WRITE:i#(512\reclen)=0 $EXTRACT(pad,1,512#reclen)
  USE $PRINCIPAL
}
CLOSE file
SET cmd="SET FILE/ATTRIB=(ORG:SEQ,RFM:FIX,RAT:BLK,LRL:132) "
DO $ZF(-1,cmdfile)
```

4.

Convert

The convert method will allow you to create any file structure. This is done by first writing to a temporary file, in a default format created with "WNV" or "WNM". Also create a FDL file describing the format that you want. Finally issue:

DO $ZF(-1,"CONVERT "tmp" "file"/FDL="fdl)

There are additional options that you might include in the CONVERT command, depending upon the desired attributes of the file you are creating. The two options that may prove useful are

- /FIXEDCONTROL (abbreviate /FIX) which you can add to indicate that CONVERT should use the first two octets of each source record to create a fixed control area.
- /PAD=%x20 which you can add to indicate that CONVERT should pad short records with spaces. Useful when writing fixed records.

```
; Sample code for
;
; FILE
;   ORGANIZATION indexed
; RECORD
;   FORMAT fixed
;   CARRIAGECONTROL none
;   SIZE 80
; KEY 0
;   NAME sequence
;   TYPE string
;   POSITION 0
;   LENGTH 8
;
SET reclen=80
SET fdl=##CLASS(%File).TempFilename("FDL")
OPEN fdl:"WNM"
USE fdl
WRITE "FILE",!
WRITE $CHAR(9),"ORGANIZATION",$CHAR(9),"INDEXED",!
WRITE "RECORD",!
WRITE $CHAR(9),"FORMAT",$CHAR(9),"FIXED",!
WRITE $CHAR(9),"CARRIAGECONTROL",$CHAR(9),"NONE",!
WRITE $CHAR(9),"SIZE",$CHAR(9),reclen,!
WRITE "KEY 0",!
WRITE $CHAR(9),"NAME",$CHAR(9),"sequence",!
WRITE $CHAR(9),"TYPE",$CHAR(9),"string",!
WRITE $CHAR(9),"POSITION",$CHAR(9),0,!
WRITE $CHAR(9),"LENGTH",$CHAR(9),8,!
USE $PRINCIPAL CLOSE fdl
SET tmp=##CLASS(%File).TempFilename("TMP")
OPEN tmp:"WNM"
FOR  {
  ; Get record and QUIT when there are no more.
  USE tmp WRITE record,! USE $PRINCIPAL
}
DO $ZF(-1,"CONVERT "tmp" "file"/FDL="fdl"/PAD=%X20")
DO ##CLASS(%File).Delete(tmp)
DO ##CLASS(%File).Delete(fdl)
```

References:

[1] The CONVERT ANALYZE/RMS and CREATE/FDL utility, as well as the FDL language are all documented in:
OpenVMS Record Management Utilities Reference Manual
Order Number: AA-PV6QD-TK
<http://h30266.www3.hpe.com/odl/axpos/opsys/vmsos84/6027/6027PRO.HTML>

[2] Caché I/O is documented in:
Caché I/O Device Guide (especially § 7.2)
Book code: GIOD
<http://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=GIOD>

#Interoperability #Caché