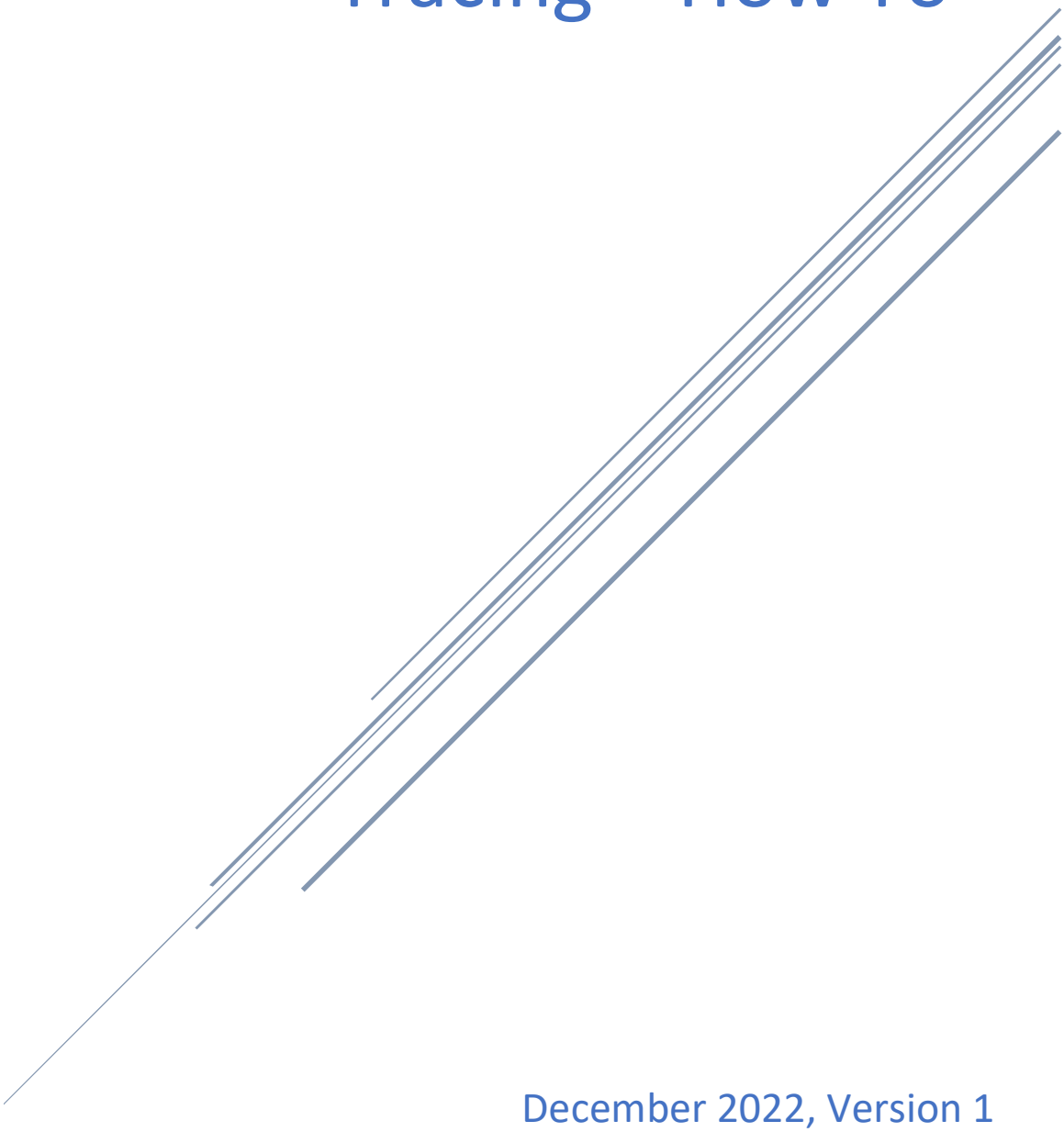
VSI OpenVMS LAN Driver Tracing – How To



December 2022, Version 1

Contents

LAN Driver Tracing

The LAN drivers record trace data of various events during LAN driver operation, including user startup and shutdown, link changes happen, and transmit and receive errors. Also, transmit and receive packet data can be recorded.

The packet data and a few interesting other trace events can be written to a PCAP file for analysis using WireShark.

Tracing is initiated on device configuration using default settings. These settings can be modified by setting or clearing a few flags in the LAN_FLAGS system parameter. These bits are

- Bit24 - Stop tracing when the trace buffer is full.
- Bit27 - Disable tracing during driver initialization (this bit only effective when the device is first configured).
- Bit29 - Enable debug mode. Enable tracing of most events and use a larger trace buffer.
- Bit30 - Enable all tracing. Enable tracing of most events.

The trace data can be displayed using the LAN extension for SDA and by LANCP commands.

LANCP Trace Commands

Setting trace parameters

$ MC LANCP SET DEVICE devname /[NO]TRACE=([NO]MASK=valuelist, [NO]SIZE=n, [NO]STOP=valuelist,keywords) [/ALL]

- valuelist is (high32bits,low32bits) defining a 64-bit trace or stop mask.
- The trace mask selects events to be traced.
- The stop mask selects events to stop tracing. Tracing stops 64 trace entries later. For example, you select a transmit error and you really want to see what happens after the error, so tracing doesn't stop on the matched event – it goes a bit past so you can see the aftermath.

- The keywords cover most events.
    - PAUSE says stop tracing but don't deallocate the trace buffer, so you can look at the trace data without it being disturbed.
    - DEFAULT selects the default trace mask.
    - MALL selects all trace mask bits, MNONE selects no trace mask bits.
    - SALL selects all stop mask bits, SNONE selects no stop mask bits.
    - RTINT selects a common set of trace events - receive done, transmit issued, interrupt.
    - RTFINT selects a common set of trace events - RTINT plus fork start and fork done.

If no trace entries are specified by mask or keyword, transmit and receive full packet data is traced.

If keyword(s) are supplied, only those bits are part of the trace mask, i.e., TRACE=NOMASK,PK is the same as TRACE=PK.

There are keywords for all the other trace bits. In addition, you can specify a bit number such as M00 for the one-second timer trace mask bit or S00 for the equivalent stop mask bit. The small, medium, large, full packet keywords affect how much of each packet is saved in the trace data.

The keywords are listed in the LANCP trace output header - with the bit number and all caps keyword. The active mask and stop bits are listed as M and S respectively in the columns next to the bit number.

```
SystemName Device Trace Data DEVNAME (Device Description) (22-NOV-2021 19:46:20.26):
  Trace mask 00404000,00000000, stop mask 00000000,00000000, trace buffer size 2048 entries
    00000001,0    00 TIMER-One second timer       01000000,0    24 XLARGE-Transmit pk (lg)
    00000002,0    01 INTR-Interrupt               02000000,0    25 XFULL-Transmit pk (all)
    00000004,0    02 FSTART-Fork started          04000000,0    26 VCIACTION-VCI action
    00000008,0    03 FDONE-Fork done              08000000,0    27 LANACTION-LAN action
    00000010,0    04 FERROR-Fork error            10000000,0    28 LANOTHER-LAN other
    00000020,0    05 FSOFT-Fork soft error        20000000,0    29 RDL error
    00000040,0    06 STATE-State change           40000000,0    30 Segment
    00000080,0    07 UINIT-First user start       80000000,0    31 Bit31
    00000100,0    08 UCHANGE-User change          0,00000001    32 Link state
    00000200,0    09 USTOP-Last user stop         0,00000002    33 Transmit timeout
    00000400,0    10 SHUTDOWN-Device shutdown     0,00000004    34 Reset device
    00000800,0    11 RISSUE-Receive issued        0,00000008    35 Setup link
    00001000,0    12 RDONE-Receive done           0,00000010    36 Check link
    00002000,0    13 RERROR-Receive error         0,00000020    37 SET_MAC function
    00004000,0 M  14 RSMALL-Receive pk (sm)       0,00000040    38 Message
    00008000,0    15 RMEDIUM-Receive pk (md)      0,00000080    39 Long fork time
    00010000,0    16 RLARGE-Receive pk (lg)       0,00000100    40 Long transmit
    00020000,0    17 RFULL-Receive pk (all)       0,00000200    41 Long receive
    00040000,0    18 XISSUE-Transmit issued       0,00000400    42 PHY rw error
    00080000,0    19 XQUEUE-Transmit queued       0,00000800    43 Auto-negotiation
    00100000,0    20 XDONE-Transmit done          0,00001000    44 Receive error pkt
    00200000,0    21 XERROR-Transmit error        0,00002000    45 Long second
    00400000,0 M  22 XSMALL-Transmit pk (sm)      0,00004000    46 Other
    00800000,0    23 XMEDIUM-Transmit pk (md)
  MALL,MNONE,SALL,SNONE - M00..M63,S00..S63
```

```
  RTINT - INTR, RDONE, XISSUE, XDONE
  RTFINT - RTINT, FSTART, FDONE
  PK - RFULL, XFULL (also SMALLPK, MEDIUMPK, LARGEPK)
```

The size is the number of 32-byte trace entries to allocate for the trace buffer. The minimum is 512, maximum is 1000000. If not specified, the size is not changed. The initial size is 512. The size if trace entries are selected but no size has been specified is 512.

/ALL applies to the devname. EI/ALL selects all EI devices. E/ALL selects all Ethernet devices. /ALL selects all devices. EIA/ALL selects all EIA devices and since there is only one EIA device, EIA/ALL is equivalent to EIA.

The SET DEVICE/TRACE ... command restarts tracing with a reinitialized trace buffer, i.e., tracing does not resume, it is restarted from scratch.

Extracting trace data for WireShark

```
$ MC LANCP SHOW DEVICE devname /TRACE /SELECT=(valuelist,keywords) /REVERSE
/OUTPUTFILE=textfilename /PCAPFILE=pcapfilename /[NO]HEADER
```

The PCAP file is binary WireShark LIBPCAP format.

/ALL is implied.

Trace output includes the trace header above, followed by a list of trace entries like so:

```
1   27474.8703476       0   17:10:21.15  Transmit issued   xmt 0 rindex 96 plen 42 vtag 0000 vcrp 81929B00
       DA FF-FF-FF-FF-FF-FF  SA 08-00-27-73-13-92  PTY 08-06
       00000000 00000000 00000000 00000000 00000608 92137327 0008FFFF FFFFFFFF  ........'s.....................  :0000
                                                  0000 00000000 00000000  ..........                        :0020
4   27474.8704314     838   17:10:21.15  Interrupt         p1 C800C0 p2 0 p3 30 p4 0
5   27474.8704471     156   17:10:21.15  Fork started      p1 0 p2 0 p3 0 p4 0
6   27474.8704478       6   17:10:21.15  Transmit done     xmt 0 rindex 95 plen 42 vtag 0000 vcrp 81929B00
7   27474.8704618     140   17:10:21.15  Fork done         p1 0 p2 0 p3 0 p4 0
8   27474.8705240     622   17:10:21.15  Interrupt         p1 C800C0 p2 0 p3 31 p4 0
9   27474.8705361     121   17:10:21.15  Fork started      p1 0 p2 0 p3 0 p4 0
10   27474.8705532     170   17:10:21.15  Receive done      rcv 199 rindex 0 plen 64 vtag 0000 vcrp 81916000
       DA 08-00-27-73-13-92  SA 52-54-00-12-35-02  PTY 08-06
       0202000A 02351200 54520200 04060008 01000608 02351200 54529213 73270008  ..'s..RT..5.....RT..5.....  :0000
       07F0B1FD 00000000 00000000 00000000 00000000 00000F02 000A9213 73270008  ..'s............................  :0020
13   27474.8705590      58   17:10:21.15  Fork done         p1 0 p2 0 p3 0 p4 0
```

The first column is the sequence number which is the number of 32-byte chunks used. In this example, the first trace entry is a transmit packet which spans 3 32-byte chunks, the next trace entry is sequence #4.

The 2nd and 4th columns are the current time in seconds since boot (from the scc value) and system time. The 3rd column is the number of 100 nsec time units since the last trace entry.

The fifth column is the name of the trace entry.

The remaining output is trace-entry-specific. There are 4 longwords listed as p1, p2, p3, p4.

The /SELECT qualifier allows display of only the selected trace entries. These are selected using the same keywords as SET DEVICE/TRACE.

The /HEADER qualifier is the default and includes the trace mask and settings. You can specify /NOHEADER to suppress the header output.

Saving trace data

```
$ MC LANCP SHOW DEVICE devname /TRACE /INPUTFILE=filename /DUMPFILE=filename /[NO]HEADER
```

The DUMPFILE qualifier causes the trace data to be written to a dump file. The LAN SDA file extension includes the /DUMPFILE=filename as well, so that trace data can be extracted from a dump (or the running system) using SDA.

The /INPUTFILE qualifier redirects LANCP to read the trace data from the specified dump file rather than the device specified. Note that the trace bit definitions then come from the dump file so it doesn't matter what device you specify to read from.

This is used to display PCAP output for trace data extracted from a system dump file, or trace data saved to a dump file for later perusal. The LAN SDA extension doesn't include PCAP output, so you would save the trace data to a dump file and use LANCP to read the dump file and write the PCAP output.

Some examples

1. Start tracing on device EIA and select full packet data:

   The trace buffer size isn't changed.

   These are synonymous:

   ```
   $ MC LANCP SET DEV/TRACE EIA
   $ MC LANCP SET DEV/TRACE=PK EIA
   $ MC LANCP SET DEV/TRACE=(RFULL,XFULL) EIA
   $ MC LANCP SET DEV/TRACE=MASK=(%x02020000,0) EIA
   ```

2. Start tracing with a common trace selection looking at interrupts and timing of transmit issue, completion, receive completion, fork processing:

   The trace buffer size isn't changed.

   ```
   $ MC LANCP SET DEV/TRACE=RTINT EIA    (without fork start and fork done)
   $ MC LANCP SET DEV/TRACE=RTFINT EIA   (with fork start and fork done)
   ```

3. Start tracing as the last example but include "Receive issued" (receive buffer given to the device):

   These are synonymous:

   ```
   $ MC LANCP SET DEV/TRACE=(RTINT,RISSUE) EIA
   $ MC LANCP SET DEV/TRACE=(RTINT,M11) EIA
   ```

4. Pause tracing:

   ```
   $ MC LANCP SET DEV/TRACE=PAUSE EIA
   ```

5. Stop tracing and deallocate the trace buffer:

   These are synonymous:

   ```
   $ MC LANCP SET DEV/NOTRACE EIA
   ```

```
$ MC LANCP SET DEV/TRACE=SIZE=0 EIA
```

6. Look at trace results:

   Forward or reverse:

   ```
   $ MC LANCP SHOW DEV/TRACE EIA
   $ MC LANCP SHOW DEV/TRACE EIA/REVERSE
   ```

   With PCAP and text output:

   ```
   $ MC LANCP SHOW DEV/TRACE EIA/PCAP=tmp.pcap
   $ MC LANCP SHOW DEV/TRACE EIA/PCAP=tmp.pcap/OUTPUTFILE=trace.txt
   ```

   Selecting particular trace entries:

   ```
   $ MC LANCP SHOW DEV/TRACE=PK EIA
   $ MC LANCP SHOW DEV/TRACE=(STATE,UCHANGE) EIA
   $ MC LANCP SHOW DEV/TRACE=(MASK=(%x40,0),UCHANGE) EIA
   ```

   To not include the trace header:

   ```
   $ MC LANCP SHOW DEV/TRACE EIA/NOHEADER
   ```

   Note that you can use the time difference to see, for example, if you select interrupt entries, what is the interrupt spacing. Or if you select FSTART and FDONE, you can see how long the driver is spending in its fork process. Or if you select XISSUE, you can see how fast transmits are being sent out.

   Note that in the LAN SDA extension, the first two commands are:

   ```
   SDA> LAN TRACE/DEV=EIA
   SDA> LAN TRACE/DEV=EIA/REVERSE
   ```

   Note that for LANCP commands, tracing is paused when data is first requested, and resumed on the last response to LANCP. SDA does not pause tracing like this, but you can pause tracing by doing:

   ```
   $ MC LANCP SET DEV/TRACE=PAUSE EIA
   ```

   then looking at the trace data in SDA. No reason to pause the tracing if you don't mind looking at a changing trace buffer and possibly some bogus entries.