Demystifying STREAMS memory: VOS release 15.1
by Noah Davids

STCP is being used more and more, by both VOS and user applications. Because STCP is the biggest consumer of STREAMS memory, it's important to understand how to allocate and monitor STREAMS memory to avoid slowdowns and failure. For example, if the system exceeds usage thresholds, memory allocations will fail. This will, at best, trigger a slow down of the system as allocations have to be retried, and, at worst, the failure of things like TCP connections.

What exactly is STREAMS memory? STREAMS memory is regular memory, there is no physical difference. VOS partitions its kernel memory up for various subsystems to use. One of those subsystems is STREAMS. By default 1/8 of kernel memory is available for STREAMS to use. The memory is not pre-allocated so it is possible that the amount of memory currently allocated to STREAMS is much less. In 15.1 the kernel is limited to 1Gig of memory so STREAMS can use up to 128 Meg; in 15.2 the kernel limit goes up to 2 Gig and STREAMS goes up to 256 Meg.

The value of 1/8 is controlled by two parameters that can be displayed by the “list_streams_params” analyze_system request. The two values define the numerator and denominator of a fraction that defines the amount of kernel memory that can be allocated for STREAMS. The default value is 1/8 (see figure 1)

```
as:  list_streams_params sys_numerator

sys max heap numerator [1 - 4095] (sys_numerator)      1

as:  list_streams_params sys_denominator

sys max heap denomin [1 - (num*2)](sys_denominator)    8
```

Figure 1 – system STREAMS memory allocation parameters

Within STREAMS there are 3 thresholds, LO, MED and HI. Again, these thresholds are defined by numerator and denominator values which represent fractions, 1/4 for HI, 1/32 for MED and 1/16 for LO (see figure 2).

```
as:  list_streams_params hi_numerator

HI  max heap numerator [0 - 4095] (hi_numerator)       1

as:  list_streams_params hi_denominator

HI  max heap denomin [1 - num]    (hi_denominator)     4

as:  list_streams_params med_numerator

MED max heap numerator [0 - 4095] (med_numerator)      1

as:  list_streams_params med_denominator

MED max heap denomin [1 - (num*4)](med_denominator)    32
```

```
as:  list_streams_params lo_numerator

LO  max heap numerator [0 - 4095] (lo_numerator)       1

as:  list_streams_params lo_denominator

LO  max heap denomin [1 - (num*4)](lo_denominator)     16
```

Figure 2 – LO, MED, HI STREAMS memory allocation parameters

The HI, MED, and LO thresholds are calculated using the algorithm in figure 3. The 9,961,472 value that is added back in at the end has to do with an adjustment for kernel memory that is pre-allocated. The algorithm will change slightly in release 15.3 and 16.

```
system = Kernel / 8
HI =  system * (1 – 1/4)
MED = HI – system / 32
LO = MED – system / 16
System = system + 9,961,472
HI = HI + 9,961,472
MED = MED + 9,961,472
LO = LO + 9,961,472
```

Figure 3 – HI, MED, LO threshold calculation algorithm

```
Assuming a 1 Gig of kernel memory this gives you
System = 1,073,741,824 / 8 = 134,217,728
HI = 134,217,728 * (1 – 1/4) = 100,663,296
MED = 100,663,296 – 134,217,728 / 32 = 96,468,992
LO = 96,468,992 – 134,217,728 / 16 = 88,080,384
System = 134,217,728 + 9,961,472 = 144,179,200 = 898 0000x
HI = 100,663,296 + 9,961,472 = 110,624,768 = 698 0000x
MED = 96,468,992 + 9,961,472 = 106,430,464 = 658 0000x
LO = 88,080,384 + 9,961,472 = 98,041,856 = 5D8 0000x
```

Figure 4 HI, MED, LO threshold calculation – example

You can change the value of sys_denominator any time after the system has booted and HI, MED, LO thresholds will be recomputed. It may take a few minutes for that to show up in analyze_system. I would not recommend changing any of the other values.

Allocations made during interrupts use the HI threshold. Allocations for STREAMS configuration messages use the MED threshold. Allocations for writing data use the LO threshold for the most part with a few using the MED threshold. Allocations made after STREAMS memory usage exceeds the associated threshold will fail. So, since allocations will start to fail when usage exceeds the LO threshold, that is the number to care about.

When STREAMS memory usage reaches 88% of the LO value you will start to see messages in the syserr_log that look like:

```
Streams allocation is at or above 88% of current BPRI_LO limit.
```

Figure 5 – Syserr_log message indicating STREAMS memory allocation approaching the LO threshold

You can even exceed 100% (figure 6). How can you do that? Even after the LO threshold is reached the system can continue allocating based on the MED and HI thresholds. That can drive the current allocation above the LO threshold which is what is used to calculate the percentage value.

```
Streams allocation is at or above 103% of current BPRI_LO limit.
```

Figure 6 – Syserr_log message indicating LO threshold has been exceeded

If things get really bad you will see messages like the one in figure 7.

```
Streams has been unable to get system memory 20 time(s).
```

Figure 7 – Syserr_log message indicating STREAMS memory allocation problems.

Monitoring STREAMS Memory Usage:
There are two ways to monitor STREAMS memory. The first is to monitor overall usage, how close you are to the LO threshold. The second is to look at the actual modules that are making the allocations.

Monitoring overall usage:
This is the same thing that the system does when it puts out the message in figure 5. However, you can see what the actual value is instead of waiting for a syserr_log message. The values of interest are displayed by the analysze_system request “dump_stream –stm_msg”. This request will display a great deal of data that is not of interest (for this discussion), so use the match feature to trim the data down to a useful size.

The first calculation, pmm_allocated_size/pmm_limit[BPRI_LO], tells you how close you are to reaching the LO threshold. The second calculation, pmm_total_size/pmm_limit[BPRI_LO], tells you what your maximum usage has been. This value can actually go down. If memory allocated to STREAMS is not used in a very long time it can be given back to the system for alternative uses, in that case the max usage can go down.

```
as
VOS Release 15.1.2ah, analyze_system Pre-release
Current process is 1982, ptep 867E54C0, Noah_Davids.CAC
as:  match 'hq_pool[0]->pmm_total_size' ; dump_stream -stm_msg
    hq_pool[0]->pmm_total_size         = 0097C280x
as:  match 'hq_pool[0]->pmm_limit[BPRI_LO]' ; dump_stream -stm_msg
    hq_pool[0]->pmm_limit[BPRI_LO]     = 05D80000x
as:  match 'hq_pool[0]->pmm_allocated_size' ; dump_stream -stm_msg
    hq_pool[0]->pmm_allocated_size     = 001E4480x
as:  q
ready  19:57:13
calc 001E4480x / 05D80000x * 100
2.02323383188503
ready  19:57:51
calc 0097C280x / 05D80000x * 100
```

```
10.1443693599599
ready  19:58:44
```

Figure 8 – using dump_stream –stm_msg to calculate current usage

The macros BPRI_LO_check.cm and BPRI_LO_alert.cm at the end of this article automatically do these calculations for you. BPRI_LO_check prints out the current values. BPRI_LO_alert is run in a started process and continually checks the values. If the current usage exceeds a specified threshold, a message is sent to a user or group of users.

Monitoring which modules are using STREAMS memory:
The analyze_system request “scan_streams_msgs” will display every STREAMS module that has allocated STREAMS memory and the location in the module where the allocation was done. Using this request you can see how much each module has allocated. The locations are release specific and require access to the source code to understand. Still, many of the names are somewhat self explanatory and will give you some idea of what is using your STREAMS memory resources.

```
as:  scan_streams_msgs
   Count     Size  Free Cnt  Free Sz  Caller; Q_INFO

     416 0001A000         0 00000000  genet_recv+833, line 314; 00000000
     384 00018000         0 00000000  udp_receive+1368, line 716; 00000000
      46 00002E00         0 00000000  tcp_stream+29B4, line 839; 00000000
      46 00002E00         0 00000000  tcp_stream+2BDC, line 864; 00000000
      46 00002E00         0 00000000  tcp_stream+2CBC, line 874; 00000000
      45 00002D00         0 00000000  tcp_stream+2B17, line 857; 00000000
      41 00002900         0 00000000  tcp_stream+2DAF, line 888; 00000000
      38 00002600         0 00000000  sthi+6B83, line 3621; 00000000
      38 00002600         0 00000000  sthi+6BBF, line 3624; 00000000
      35 00002300         0 00000000  tcp_stream+2A5F, line 845; 00000000
      32 00002000         0 00000000  udp_receive+1368, line 716; sth (8097C578)
      20 00001E00         0 00000000  udp_stream+14A3, line 567; 00000000
      13 00000D00         0 00000000  fast+322, line 262; 00000000
      13 00000D00         0 00000000  fast+447, line 269; 00000000
      13 00000D00         0 00000000  tcp_ip+321, line 206; 00000000
      13 00000D00         0 00000000  tcp_ip+446, line 213; 00000000
      10 00000A00         0 00000000  pc_async_stream+1EDE, line 1440; 00000000
       8 00000800         0 00000000  open+95A, line 504; 00000000
       7 00000700         0 00000000  sdlmux+27A1, line 299; 00000000
       5 00000500         0 00000000  sdlmux_stm+1F9D, line 380; 00000000
       5 00000500         0 00000000  cpc_lapb_stream+2CA3, line 1449; 00000000
       5 00000500         0 00000000  cpc_lapb_stream+2D18, line 1456; 00000000
       5 00000500         0 00000000  cpc_lapb_stream+2E76, line 1480; 00000000
       1 00000480         0 00000000  loop+B3A, line 186; 00000000
       4 00000400         0 00000000  arp+171E, line 1246; sdlmux_u (BEC7D268)
       4 00000400         0 00000000  cpc_lapb_stream+2DBB, line 1469; 00000000
       4 00000400         0 00000000  cpc_lapb_stream+2E28, line 1476; 00000000
       3 00000300         0 00000000  lat_ai_init+5A1, line 272; 00000000
       3 00000300         0 00000000  sdlmux+27A1, line 299; sdlmux_l (BEC7D228)
       3 00000300         0 00000000  adp+2568, line 1190; sdlmux_u (BEC7D268)
       3 00000300         0 00000000  cpc_lapb_stream+2C51, line 1444; 00000000
       2 00000200         0 00000000  lat_ai_init+547, line 251; 00000000
       1 00000100         0 00000000  sthi+58F9, line 2876; 00000000
```

```
       1 00000100         0 00000000  lat_al_call_side+269A, line 1593; 00000000
       1 00000100         0 00000000  route_table+813, line 347; 00000000
       1 00000100         0 00000000  timech+506, line 249; 00000000
       1 00000100         0 00000000  timech+AA6, line 424; 00000000
       1 00000100         0 00000000  pt_tioc_mod+A10, line 206; 00000000
       1 00000100         0 00000000  pt_tioc_mod+A71, line 218; 00000000
       1 00000100         0 00000000  pt_tioc_mod+AAD, line 224; 00000000
       0 00000000     13949 0066B500  00000000; 00000000
       0 00000000       133 0003DC80  vos_pse+5B3D, line 3136; 00000000
       0 00000000       413 00067380  vos_pse+5B3D, line 3136; sth (8097C578)
       0 00000000        78 00003400  sth+1CA4, line 899; sth (8097C578)
       0 00000000         1 00000100  sthi+6AD3, line 3584; 00000000
       0 00000000         1 00000100  sthi+6AD3, line 3584; sth (8097C578)
       0 00000000        12 00000600  lat_al_buffers+BD9, line 415; 00000000
       0 00000000         3 00000300  lat_al_buffers+169D, line 863; 00000000
       0 00000000        22 00002400  lat_al_buffers+172C, line 877; 00000000
       0 00000000         1 00000880  lat_st_lrput+402, line 136; lat_streams (8
+097E0E0)
       0 00000000         1 00000100  lat_st_lrsrv+1245, line 775; lat_streams (
+8097E0E0)
       0 00000000         1 00000100  stcp_link_service+22C8, line 846; 00000000
       0 00000000         1 00000100  stcp_link_service+2C3B, line 1229; osl (BE
+B86608)
       0 00000000        10 00001500  stcp_link_service+56B6, line 2702; osl (BE
+B86608)
       0 00000000         4 00000400  arp+BF7, line 639; 00000000
       0 00000000         5 00000500  arp+BF7, line 639; arp (BEBD8130)
       0 00000000       639 000A2B00  genet_xmit+DAD, line 288; 00000000
       0 00000000       226 0000E200  genet_xmit+DAD, line 288; gbe (BEC100E8)
       0 00000000         3 00000300  genet_xmit+DAD, line 288; udp (BED4E070)
       0 00000000        40 00002800  genet_xmit+DB9, line 289; 00000000
       0 00000000        28 00000E00  icmp+65F, line 240; 00000000
       0 00000000         1 00000280  icmp+1395, line 777; 00000000
       0 00000000         3 00000300  icmp+18D1, line 958; 00000000
       0 00000000         3 00000180  icmp+1EF7, line 1228; 00000000
       0 00000000        18 00001200  icmp+26BB, line 1497; 00000000
       0 00000000        19 00004380  ip_stream+193D, line 1141; 00000000
       0 00000000         6 00000600  ip_stream+3157, line 1835; ip (BEC52378)
       0 00000000        13 00000D00  ip_stream+383A, line 1987; ip (BEC52378)
       0 00000000         1 00000100  sdlmux+9C58, line 4405; 00000000
       0 00000000         1 00000100  sdlmux+9C58, line 4405; sdlmux_l (BEC7D228
+)
       0 00000000       230 00050280  ack+17AD, line 1033; 00000000
       0 00000000       481 00072880  ack+17AD, line 1033; tcp (BECF8B68)
       0 00000000        12 00000C00  ip_tcp+5C0, line 370; 00000000
       0 00000000        11 00000B00  ip_tcp+5C0, line 370; tcp (BECF8BA8)
       0 00000000        34 00002200  receive+735, line 399; 00000000
       0 00000000         2 00000200  tcb+10A1, line 826; 00000000
       0 00000000         3 00000300  tcb+15B8, line 1044; 00000000
       0 00000000         2 00000200  tcb+15D8, line 1050; 00000000
       0 00000000         1 00000080  tcb+17C0, line 1252; tcp (BECF8B68)
       0 00000000     15718 003D6600  tcp_stream+4E16, line 1646; 00000000
       0 00000000     15718 003D6600  tcp_stream+4E45, line 1651; 00000000
       0 00000000     15716 003D6400  tcp_stream+4E74, line 1656; 00000000
       0 00000000     15715 003D6300  tcp_stream+4EA3, line 1661; 00000000
       0 00000000     15709 003D5D00  tcp_stream+4ED2, line 1666; 00000000
       0 00000000        23 00001700  tcp_stream+AD6E, line 3073; tcp (BECF8B68)
```

```
       0 00000000       275 00019C80  udp_stream+1CAB, line 813; 00000000
       0 00000000         1 00000880  tnmod+1AF9, line 695; 00000000
       0 00000000         1 00000080  tnmod+2523, line 1030; tnmod (BED6F040)
-------- --------  -------- --------
    1319 00053480     95288 01BE6100
```

Figure 9 – scan_streams_msgs output

Dealing with STREAMS Memory Problems:
What can you do if you see the messages in figures 5, 6 or 7?

There are three cases that can cause STREAMS memory problems. The first is a STREAMS memory leak. The second is application problems and the third is an overtaxed system.

A STREAMS leak is caused when the chain of pointers that associate an allocated block of memory to a STREAMS module is broken. There is no way to recover this memory short of rebooting. You can use the analyze_system request “scan_streams_msgs -leaks_only” to locate leaks. This request can take quite a while before it starts to write anything so be patient.

```
as:  scan_streams_msgs -leaks_only
   Count     Size  Free Cnt  Free Sz  Caller; Q_INFO

  914018 0A75C980         0 00000000  tcp_stream+39EC, line 1295; 00000000
      41 00015C80         0 00000000  ad_wfunc+1624, line 820; 00000000
      23 0000C380         0 00000000  ad_dlmux+488, line 176; 00000000
       5 00005280         0 00000000  arp_stream+229C, line 957; 00000000
      44 00002100         0 00000000  route_table+7E8, line 347; 00000000
       1 00000480         0 00000000  loop+B34, line 186; 00000000
       5 000003C0         0 00000000  route_table+6D4, line 280; 00000000
       1 00000100         0 00000000  lat_al_buffers+814, line 311; 00000000
       1 000000C0         0 00000000  timech+4A1, line 247; 00000000
       1 000000C0         0 00000000  timech+9F5, line 424; 00000000
       1 00000080         0 00000000  lat_al_buffers+85C, line 322; tcp (FDA02C1
+8)
-------- --------  -------- --------
  914141 0A786840         0 00000000
```

Figure 10 looking for STREAMS leaks – and finding them

Note the large number of tcp_stream entries. This was caused by the bug stcp-2260, which has been fixed in 14.7.2ah and 15.1.2as releases. Don’t panic if you are running earlier releases, the leak only happens when you attempt to communicate with a host on another network and you do not have any route to that network. If you have a default route there will be no memory leak.

Note that there will always be a small number of reported leaks on a live system. This is because the chains are always changing, as long as the numbers to not grow too large it is not a problem.

```
as:  scan_streams_msgs -leaks_only
   Count     Size  Free Cnt  Free Sz  Caller; Q_INFO

       5 00000500         0 00000000  cpc_lapb_stream+2CA3, line 1449; 00000000
       5 00000500         0 00000000  cpc_lapb_stream+2D18, line 1456; 00000000
```

```
       5 00000500         0 00000000  cpc_lapb_stream+2E76, line 1480; 00000000
       1 00000480         0 00000000  loop+B3A, line 186; 00000000
       4 00000400         0 00000000  cpc_lapb_stream+2C51, line 1444; 00000000
       4 00000400         0 00000000  cpc_lapb_stream+2DBB, line 1469; 00000000
       4 00000400         0 00000000  cpc_lapb_stream+2E28, line 1476; 00000000
       1 00000100         0 00000000  sthi+58F9, line 2876; 00000000
       1 00000100         0 00000000  timech+506, line 249; 00000000
       1 00000100         0 00000000  timech+AA6, line 424; 00000000
       1 00000100         0 00000000  pt_tioc_mod+A10, line 206; 00000000
       1 00000100         0 00000000  pt_tioc_mod+A71, line 218; 00000000
       1 00000100         0 00000000  pt_tioc_mod+AAD, line 224; 00000000
-------- --------  -------- --------
      34 00002580         0 00000000
```

Figure 11 looking for STREAMS leaks – and not finding them

If a “scan_streams_msgs –leaks_only” indicates a real leak it needs to be reported to the CAC – along with the “scan_streams_msgs –leaks_only” output. These kinds of leaks are typically caused by unusual, or at least unanticipated, packet sequences. To locate the leak and fix it requires understanding the packet sequence which requires a detailed description of the system, running applications, application errors and the network environment and any changes to it. If possible don’t reboot your system before calling the CAC.

Applications do not directly allocate STREAMS memory but a misbehaving application can cause STREAMS to consume a lot of memory. For example, an application that creates a TCP socket and then does not read from it causes all the data received on that socket to be buffered – using STREAMS memory. An application that opens many sockets and does this on all of them can cause problems. The good news is that terminating the application will free up the memory – nothing is permanently lost. The netstat command will show bytes ready to be read but the number of bytes may not reflect the actually amount of memory used. The reason is that the incoming packets are buffered in standard sized blocks. Many small packets can create a disproportionate amount of STREAMS memory usage.

Using the analyze_system requests “dump_onetcb” or “dump_stcbq –full” you can display these blocks. The “dump_onetcb” request will do so for one TCP socket, the “dump_stcbq” request will do so for all TCP sockets. By matching on the db_size string you can display just the size, matching on the “STCP TCB” string will display just the socket address so you can tell which “sizes” go with which sockets. In figure 12, netstat shows two sockets with 30,000+ bytes in their receive queues. You can see that two sockets have a large number of “db_size” lines indicating many blocks have been allocated. If you add up the sizes they are slightly larger then what netstat is showing.

```
netstat -numeric -PCB_addr
Active connections
PCB       Proto Recv-Q Send-Q  Local Address      Foreign Address    (state)
. . .
85b8ec80  tcp    31990      0  164.152.77.34:6666 164.152.77.50:3460 ESTABLISHED
860a7380  tcp    30432      0  164.152.77.34:6666 164.152.77.50:3468 ESTABLISHED
. . .

as:  match 'STCP TCB' -or db_size; dump_stcbq -full
*************** STCP TCB @ 85ED7A40                                  ************
*************** STCP TCB @ 85EE1500                                  ************
```

```
*************** STCP TCB @ 85EE3FC0                                ************
     mp->datap->db_size                128
*************** STCP TCB @ 85EE5A40                                ************
     mp->datap->db_size                128
*************** STCP TCB @ 85EE7600                                ************
     mp->datap->db_size                128
*************** STCP TCB @ 85EE91C0                                ************
     mp->datap->db_size                128
*************** STCP TCB @ 85EEACC0                                ************
*************** STCP TCB @ 85EEF2C0                                ************
     mp->datap->db_size                128
*************** STCP TCB @ 85EF0140                                ************
*************** STCP TCB @ 85EFF880                                ************
*************** STCP TCB @ 85F00540                                ************
*************** STCP TCB @ 85F01340                                ************
. . .
*************** STCP TCB @ 860A8100                                ************
*************** STCP TCB @ 860E2080                                ************
*************** STCP TCB @ 85B8EC80                                ************
     mp->datap->db_size                128
     mp->datap->db_size                2048
     mp->datap->db_size                128
     mp->datap->db_size                512
     mp->datap->db_size                2048
     mp->datap->db_size                128
     mp->datap->db_size                1024
     mp->datap->db_size                2048
     mp->datap->db_size                128
     mp->datap->db_size                1024
     mp->datap->db_size                2048
     mp->datap->db_size                128
     mp->datap->db_size                1024
     mp->datap->db_size                2048
     mp->datap->db_size                128
     mp->datap->db_size                1024
     mp->datap->db_size                2048
     mp->datap->db_size                128
     mp->datap->db_size                1024
     mp->datap->db_size                2048
     mp->datap->db_size                128
     mp->datap->db_size                1024
     mp->datap->db_size                2048
     mp->datap->db_size                128
     mp->datap->db_size                1024
     mp->datap->db_size                2048
     mp->datap->db_size                128
     mp->datap->db_size                1024
     mp->datap->db_size                2048
     mp->datap->db_size                128
     mp->datap->db_size                1024
     mp->datap->db_size                2048
*************** STCP TCB @ 860A7380                                ************
     mp->datap->db_size                128
     mp->datap->db_size                128
     mp->datap->db_size                2048
     mp->datap->db_size                128
     mp->datap->db_size                2048
```

```
     mp->datap->db_size              128
     mp->datap->db_size              1024
     mp->datap->db_size              2048
     mp->datap->db_size              128
     mp->datap->db_size              1024
     mp->datap->db_size              2048
     mp->datap->db_size              128
     mp->datap->db_size              1024
     mp->datap->db_size              2048
     mp->datap->db_size              128
     mp->datap->db_size              1024
     mp->datap->db_size              2048
     mp->datap->db_size              128
     mp->datap->db_size              1024
     mp->datap->db_size              2048
     mp->datap->db_size              128
     mp->datap->db_size              1024
     mp->datap->db_size              2048
     mp->datap->db_size              128
     mp->datap->db_size              2048
     mp->datap->db_size              2048
     mp->datap->db_size              128
     mp->datap->db_size              2048
     mp->datap->db_size              1024
     mp->datap->db_size              2048
as:
```

Figure 12 – dumping the amount of queued data associated with a TCP connection

Note that large amounts of data in the send queues can also create problems.

Finally the system can just be overtaxed. Lots of processes using a reasonable amount of STREAMS resources have the same effect of a few processes using a large amount of resources. This can be very hard to identify. Basically, if you have lots of connections all with 0 or a little data queued (0 to a few db_size lines reported in the output from “dump_stcbq –full”, see figure 12), you probably have this condition.

Unfortunately, you cannot use this technique to look for queued UDP messages. The only way to do that is to follow the queue pointers in the UCB structures. Something too detailed to describe here. It is however done in the dump_streams.cm described below.

The macro dump_streams.cm at the end of this article can be executed to collect as much information as possible. It should be executed before anything is done that would change the state of the system, for example stopping processes or the STCP stack. The output will be very large and it may take a while to complete. Once it’s complete an issue can be entered with the CAC and the data analyzed.

Command macros:

BPRI_LO_check.cm

This command macro calculates how close the current STREAMS memory usage and the maximum STREAMS memory usage values are to the STREAMS BPRI_LO threshold as a percentage of the BPRI_LO value. A value of 100% would indicate that the current usage equals the BPRI_LO threshold value.

```
& BPRI_LO_check begins here
&
& BPRI_LO_check.cm
& version 1.1 06-07-21
& Noah Davids Stratus CAC noah.davids@stratus.com
&
&attach_input
!analyze_system
..!attach_default_output (process_dir)>BPRI_LO_check_1
!match 'hq_pool[0]->pmm_total_size' ; dump_stream -stm_msg
..!detach_default_output
&set_string temp (contents (process_dir)>BPRI_LO_check_1 1)
&set equal (calc (index (quote &temp&) =) + 2)
&set_string pmm_total_size_0 (substr (quote &temp&) &equal& 9)
..!attach_default_output (process_dir)>BPRI_LO_check_1
!match 'hq_pool[0]->pmm_limit[BPRI_LO]' ; dump_stream -stm_msg
..!detach_default_output
&set_string temp (contents (process_dir)>BPRI_LO_check_1 1)
&set equal (calc (index (quote &temp&) =) + 2)
&set_string pmm_limit_BRI_LO_0 (substr (quote &temp&) &equal& 9)
..!attach_default_output (process_dir)>BPRI_LO_check_2
!match 'hq_pool[0]->pmm_allocated_size' ; dump_stream -stm_msg
..!detach_default_output
&set_string temp (contents (process_dir)>BPRI_LO_check_2 1)
&set equal (calc (index (quote &temp&) =) + 2)
&set_string pmm_allocated_0 (substr (quote &temp&) &equal& 9)
&
!quit
!display_line
!display_line
!display_line
&set temp (calc &pmm_allocated_0& / &pmm_limit_BRI_LO_0& * 100)
!display_line 'Current BRI_LO usage' &pmm_allocated_0& / &+
 &pmm_limit_BRI_LO_0& * 100 = &temp&%
&set temp (calc &pmm_total_size_0& / &pmm_limit_BRI_LO_0& * 100)
!display_line 'MAX     BRI_LO usage' &pmm_total_size_0& / &+
 &pmm_limit_BRI_LO_0& * 100 = &temp&%
!display_line
!display_line
!display_line
!display_line
!display_line
&
& BPRI_LO_check ends here
```

Example execution of BPRI_LO_check.cm

```
BPRI_LO_check.cm -form -usage
 ------------------------------- BPRI_LO_check -------------------------------
 No arguments required. Press ENTER to continue.
```

```
BPRI_LO_check.cm
VOS Release 15.1.2ah, analyze_system Pre-release
Current process is 619, ptep 87CB6000, Noah_Davids.CAC
as:  as:  as:  as:


Current BRI_LO usage 00202F80x / 05D80000x * 100 = 2.15144029913102%
MAX     BRI_LO usage 01C39580x / 05D80000x * 100 = 30.1860952122326%
```

BPRI_LO_alert.cm

This command macro calculates how close the current STREAMS memory usage is to the STREAMS BPRI_LO threshold as a percentage of the BPRI_LO value. If the percentage is greater than the "threshold" argument a message is sent to all logged in users whose name matches the "users" argument. The check is performed every "sleep" argument seconds.

While you can run this from an interactive process I suspect it will be more useful as a started process

```
& BPRI_LO_alert begins here
&
& BPRI_LO_alert.cm
& version 1.1 06-07-21
& Noah Davids Stratus CAC noah.davids@stratus.com
&
&begin_parameters
     threshold threshold:number,required,min(1),=90
     USERS users:string,required,=*
     SLEEP sleep:number,req,=5
&end_parameters
&
&echo no_command_lines no_input_lines no_macro_lines
&
&attach_input
!analyze_system
&label again
..!attach_default_output (process_dir)>BPRI_LO_1
!match 'hq_pool[0]->pmm_limit[BPRI_LO]' ; dump_stream -stm_msg
..!detach_default_output
&set_string temp (contents (process_dir)>BPRI_LO_1 1)
&set equal (calc (index (quote &temp&) =) + 2)
&set_string pmm_limit_BRI_LO_0 (substr (quote &temp&) &equal& 9)
..!attach_default_output (process_dir)>BPRI_LO_2
!match 'hq_pool[0]->pmm_allocated_size' ; dump_stream -stm_msg
..!detach_default_output
&set_string temp (contents (process_dir)>BPRI_LO_2 1)
&set equal (calc (index (quote &temp&) =) + 2)
&set_string pmm_allocated_0 (substr (quote &temp&) &equal& 9)
&
&set temp (calc &pmm_allocated_0& / &pmm_limit_BRI_LO_0& * 100)
&if &temp& > &threshold&
&then ..!start_process (string send_message &USERS& &+
    (quote BRPI_LO is now &temp&))
```

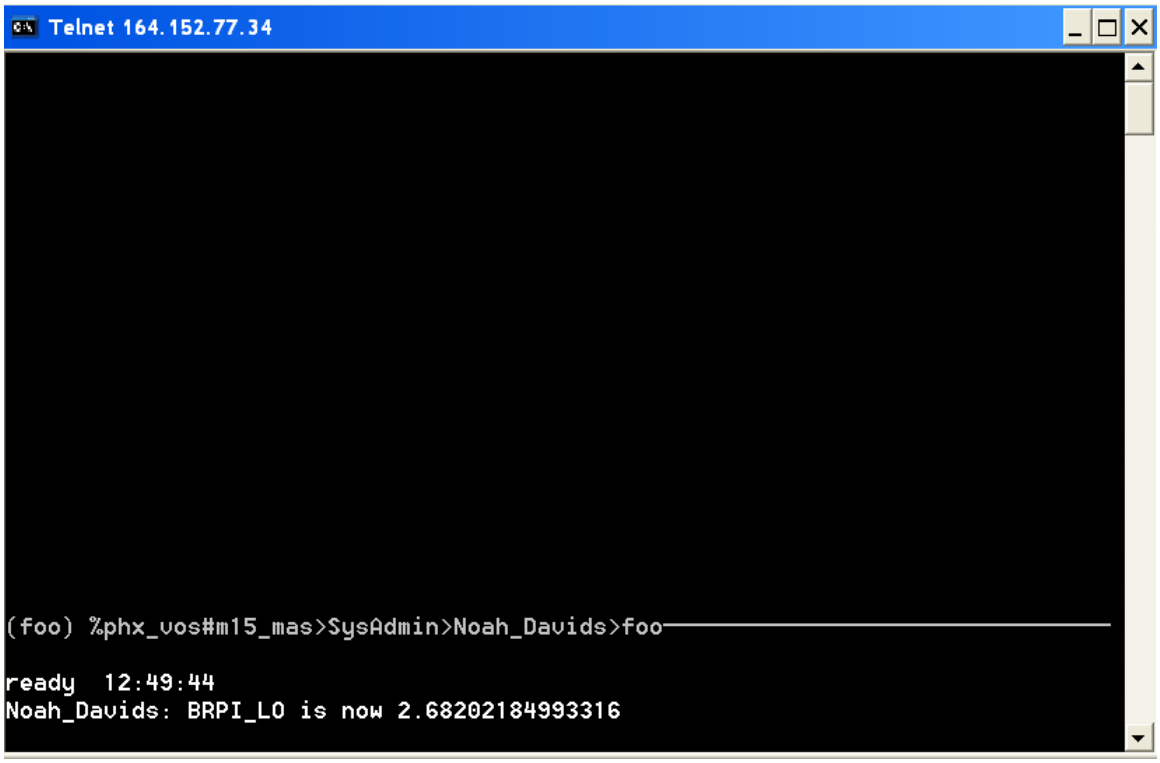
```
&
!sleep -seconds &SLEEP&
&goto again
&
& BPRI_LO_alert ends here
```

Example execution of BPRI_LO_alert.cm.
Note that this example uses a threshold of 2 so that I can show an example of the message without running my module out of STREAMS memory.

```
BPRI_LO_alert –form -usage

------------------------------- BPRI_LO_alert -------------------------------
 threshold: 90
 users:     *
 sleep:     5

start_process 'BPRI_LO_alert 2 *.SysAdmin 5' -privileged
ready  15:44:40
```



dump_streams.cm

This macro tries to gather as much information as possible about the current STREAMS usage on a module. Because a lot of output is generated it places it all in the file named dump_streams.(date).(time). Output is redirected via the attach_default_output command, so if the command macro is interrupted for

any reason you will have to execute the detach_default_output command to redirect output back to the terminal.

```
& dump_streams.cm begins here
&
& dump_streams.cm
& version 1.2 06-08-01
& Noah_Davids Stratus CAC noah.davids@stratus.com
&
&attach_input
&set_string out dump_streams.(date).(time)
attach_default_output &out&
!>system>stcp>command_library>netstat -numeric -all_sockets -PCB_addr
!analyze_system
!scan_streams_msgs -leaks_only
!scan_streams_msgs
!match *** -or db_size ; dump_stcbq -full
..!detach_default_output
..!attach_default_output (process_dir)>map
!match 'pointer to UCB' ; dump_ucbq
..!display_line END
!quit
!detach_default_output
!attach_default_output (process_dir)>map2
!display_line (substr (contents (process_dir)>map 1) 41)
&set line 2
&while (contents (process_dir)>map &line& ) ^= 'as:  END'
!display_line (substr (contents (process_dir)>map &line&) 36)
&set line (calc &line& + 1)
&end
!display_line END
!detach_default_output
analyze_system
..!attach_default_output (process_dir)>map3
&set line 1
&while (contents (process_dir)>map2 &line&) ^= 'END'
&if (contents (process_dir)>map2 &line&) ^= '00000000'
&then &do
..!display_line STCP UCB: (contents (process_dir)>map2 &line&)
!match ub_qptr ; dump_oneucb (contents (process_dir)>map2 &line&)
&end
&set line (calc &line& + 1)
&end
..!display_line END
..!detach_default_output
..!attach_default_output &out& -append
&set line 1
&while (contents (process_dir)>map3 &line&) ^= 'as:  END'
..!display_line (contents (process_dir)>map3 &line&)
&set line (calc &line& + 1)
!match db_size ; dump_stream -queue (substr (contents &+
  (process_dir)>map3 &line&) 41)
&set line (calc &line& + 1)
&end
!dump_stream -long -all_streams -msg_q -msg_dump_size 192
```

```
!quit
!detach_default_output
&
& dump_streams.cm ends here
```

Example execution of dump_streams.cm.

```
dump_streams
as:  VOS Release 15.1.2ah, analyze_system Pre-release
Current process is 215, ptep 86F94740, Noah_Davids.SysAdmin
as:  as:  ready  14:48:49
ls dump*

Files: 2, Blocks: 623

w        622 dump_streams.06-08-01.14:48:45
w          1 dump_streams.cm

Directories: 0

Links: 0
```