

ServerCap is a program that captures FSERVE network packets. It comprises a replacement FSERVE, called CSERVE. And a SuperBASIC program to view and manage the packet captures.

Note that this program will not capture NETI or NETO network packets. It will also not capture any bad packets that may cross the QL network.

Do not try to run FSERVE and CSERVE at the same time, as the two servers are likely to interfere with each other.

CSERVE is a modified version of FSERVE that saves in memory, every packet received and sent to the server.

Before running the ServerCap_bas program, the replacement server needs to be installed.

LRESPR device_ServerCap_bin

The ServerCap_bas program should run on any QL or compatible system that supports the QL network, and in any screen mode.

The ServerCap_bas program runs rather slow on anything less than a Gold card QL. So there is also supplied a TurboCharged version of the program.

After loading and running the ServerCap_bas program, the following screen will be displayed.

```
FSERVE Packet Capture                               Version 1.00
Item  Time      From To  Function Size
-----
Current status  -  None
Start Time     -
Packets captured - 0
Buffer space free - 0

F1 Start  F2 Stop  F3 Save  F4 Load  F5 Quit
```

At the bottom of the screen are five function keys. Any unavailable functions are greyed out.

Initially only Start, Load, and Quit are available.

Main Function keys

- F1 Start This will start a capture session. You will be asked how much memory in kilobytes to use as a buffer for network packets. It will start the CSERVE job, then wait 3 seconds before looking for received packets. Packet transfers are then displayed as they are received and sent.
If the capture buffer fills, packets will no longer be saved by CSERVE, however the server will continue to service requests.
- F2 Stop This will stop a capture session, and remove the CSERVE job.
- F3 Save This will save a capture session to the file name of your choice. Note - If the file already exists, it will be overwritten without warning.
- F4 Load This will load a previously saved capture session. Any existing capture in memory will be lost.
- F5 Quit This will terminate the program, releasing any allocated memory.
- Shift F5/F10 This will redraw the program windows in case of overwritten windows.

Capping in progress

FSERVE Packet Capture				Version 1.00	
Item	Time	From	To	Function	Size
7	00:00:18	1	2	Request	12
8	00:00:18	2	1	Reply	460
9	00:00:19	1	2	Request	12
10	00:00:19	2	1	Reply	524
11	00:00:19	1	2	Request	12
12	00:00:19	2	1	Reply	524
13	00:00:21	1	2	Request	12
14	00:00:21	2	1	Reply	524
15	00:00:21	1	2	Request	12
16	00:00:21	2	1	Reply	524
Current status				- Capping	
Start Time				- 2025 Feb 20 13:22:10	
Packets captured				- 16	
Buffer space free				- 89078	
F1 Start F2 Stop F3 Save F4 Load F5 Quit					

After stopping a capture session, or loading a saved capture session. You will be given the option of selecting and viewing the captured packets.

FSERVE Packet Capture				Version 1.00	
Item	Time	From	To	Function	Size
7	00:00:18	1	2	Request	12
8	00:00:18	2	1	Reply	460
9	00:00:19	1	2	Request	12
10	00:00:19	2	1	Reply	524
11	00:00:19	1	2	Request	12
12	00:00:19	2	1	Reply	524
13	00:00:21	1	2	Request	12
14	00:00:21	2	1	Reply	524
15	00:00:21	1	2	Request	12
16	00:00:21	2	1	Reply	524
Current status - None					
Start Time - 2025 Feb 20 13:22:10					
Packets captured - 16					
Buffer space free - 89078					
Press Enter to Select Packets					
F1 Start F2 Stop F3 Save F4 Load F5 Quit					

Press ENTER to select packets. Then use the up and down cursor keys to select the required packet. The Shift up and down keys, or Page up/down keys will move you 10 packets at a time. The Enter key will display the packet header and a description of the purpose of the data block.

FSERVE Packet Capture				Version 1.00	
Item	Time	From	To	Function	Size
1	00:00:18	1	2	Request	11
2	00:00:18	2	1	Reply	9
3	00:00:18	1	2	Request	12
4	00:00:18	2	1	Reply	12
5	00:00:18	1	2	Request	12
6	00:00:18	2	1	Reply	76
7	00:00:18	1	2	Request	12
8	00:00:18	2	1	Reply	460
9	00:00:19	1	2	Request	12
10	00:00:19	2	1	Reply	524
Current status - None					
Start Time - 2025 Feb 20 13:22:10					
Packets captured - 16					
Buffer space free - 89078					
Press ENTER to view packets,ESC to exit					
F1 Start F2 Stop F3 Save F4 Load F5 Quit					

```

FSEERVE Packet Capture                               Version 1.00
Item  Time      From To  Function Size
Packet 5      Request Header    8 bytes
  HEX      82 01 00 00 4F 0C 84 62
From Station      1      To station      2
Block high        0      Block low        0
Type number       79      Data block size  12
Data checksum 132      Header checksum  98

FS.XINF          TRAP#3    D0 = 4F
D1 = 00000000
D2 = 00000040
A0 = 00310013

TAB for data block  ↑ Up  ↓ Down  ESC Exit

```

You can use the up and down keys as before to move between packet headers, and TAB to switch to the data block.

```

FSEERVE Packet Capture                               Version 1.00
Item  Time      From To  Function Size
Packet 5      Data block      12 bytes
000 00 31 00 13 00 00 00 00      .1.....
008 00 00 00 40      ...@

TAB for header  ↑ Up  ↓ Down  ESC Exit

```

The ServerCap_bas program can be used to load and view saved captures on systems that do not support the QL network like QPC2. You don't need the ServerCap_bin installed for this.

If you run the program in an expanded screen mode, There are two 'windowsOrigin' variables on line 200 of the program. These allow you change the start position of the windows. Allowing you to position the program where you want on the screen.

Technical bits

CSERVE is a special version of FSERVE, which is a function that starts the capture server running, or returns an error if CSERVE, or FSERVE is already running

Start the modified FSERVE with 'result=CSERVE(address)', Where address is an allocated area of memory.

Example of how to start CSERVER

```
bufferSize=32000
bufferBase=ALCHP(bufferSize)
POKE_L bufferBase,bufferSize
result=CSERVE(bufferBase)
```

The format of the capture in memory is as follows

\$00	long	size of the allocated memory area - must be set before starting CSERVE
\$04	long	start time of the capture
\$08	long	address of the start of the last packet added, or start of allocated area + \$0C for an empty buffer
\$0C		start of saved data packet link header
.		.
.		.

Note that the size of the remaining allocated memory area will be adjusted for each data packet saved, to prevent overrunning the buffer.

Format of saved data packets link header. There is one of these for every saved packet.

\$00	long	address of the next packet, or zero if this is the last packet
\$04	long	time the data packet received, or sent
\$08	word	size of data packet + this link header
\$0A	byte	saved packet type, 1 = FSERVE received, 2 = FSERVE sent
\$0B	byte	spare
\$0C		bytes of the data packet header and data block

Saved capture file format

A saved capture file is the same as the packet data stored in memory. Except for the first long word containing the size of the buffer is omitted, and all the address pointers are converted to offsets from the base of the saved area. This allows the saved packet data to be reloaded into a different address.



#8 is used for viewing saved packets, and covers the #5 and #6 areas
#10 is used for saving and loading saved packets

Global variables

windowsOrigin_x	Sets the top left hand position of all drawn windows
windowsOrigin_y	large screens only
capSaved	Current capture flag 1=saved, 0=not saved
FxKeys%(5)	Function key options selection flags array 1=available, 0=not available FxKeys%(1) Start FxKeys%(2) Stop FxKeys%(3) Save FxKeys%(4) Load FxKeys%(5) Quit
packetWindow\$(windowDepth,40)	Arrays for the sliding packet window Packet descriptions for selection and viewing
packetPointers(windowDepth,1)	Two pointers into the saved packets buffer Index 0 - Address of start of a packet link 1 - Saved packets number
windowDepth	Number of text lines in the packets list window #5
actLastPkt	Address of the start of the actual last saved packet link
currLastPkt	Address of the last known saved packet
lineNumber	number of the last known saved packet
bufferBase	Capture buffer address, Zero if no buffer
bufferSize	Capture buffer size, Zero if no buffer

currentStatus	Program status 0=not capping 1=capture in progress 2=loaded buffer 3=saved buffer
startTime	Capture start time
packetsCapped	Number of packets captured
bufferLeft	How much space is left in the capture buffer
	Offsets from the start of the packet buffer
bu_size	Remaining packet buffer
bu_time	Capture start time
bu_last	Address of last packet added
bu_start	Start of saved packets
	Offsets from the start of a packets link header
pk_next	Address of next packet
pk_time	Packet capture time
pk_size	Size of captured packet
pk_type	Saved packet type
pk_spare	Spare byte
pk_data	Start of saved packet
	Offsets from start of packets header
cs_hedr	Packet header
cs_dest	Destination station number
cs_self	Source station number
cs_blk_l	Block number (low)
cs_blk_h	Block number (high)
cs_type	Packet type
cs_nbyt	Number of bytes in data block
cs_dchk	Data checksum
cs_hchk	Header checksum

Packet Types

Type 1	Trap#1	Open & Delete. Delete is signified by D3 being -1	
		Request	Reply
\$00	L	D3 open type	D0 error
\$04		A0 bytes of name	A0 channel Id (optional)
\$08	B		serial only flag (optional)
Type 2	Trap#2	Close	
		Request	Reply
\$00	L	A0 channel ID	D0 error
\$04	L	D1	
\$08	L	D2	
Types 3 to 79	Trap#3	The type number refers to the value of D0 on entry	
		Request	Reply
\$00	L	A0 channel ID	D0 error
\$04	L	D1	D1
\$08	L	D2	D2
\$0C		bytes of data (optional)	bytes of data (optional)
Type 254	Large block		
		Request	Reply
\$00	L	channel ID (A0)	D0 error? FFF6
\$04	L	start address	D1 number bytes sent?
\$08	L	length	D2 number bytes sent?
\$0C		bytes of data	
Type 255	Block		
		Request	Reply
\$00	L	channel ID (A0)	D0 error
\$04	L	D1 position in file	D1 number of bytes fetched
\$08	L	D2 length	D2 length

Notes on compiling with Turbo

You will need at least 640K of RAM to compile the program, and the ServerCap_bin file loaded into memory.

Note that the Object data needs to be 6K.

Pass: 0	Line: 0	Errors: 0	Warnings: 0
© 1987 The Turbo Team		TURBO	© 2007 George Gwilt V5.09
Freeform	< 64K	Include Nos	BRIEF Set 3 windows
Object: mdv3_TurboServerCap_task			
Object data: 6K	++ COMPILE ++		TURBO buffer: 2K
Report: DISPLAY		Task: ServerCap	
Manual	Report \$	List: NO	Sound: YES Pause: NO