**Usage (# fw ctl debug –h) :**

fw ctl debug [–x] [–m <module>] [+ | –] <options | all | 0>
fw ctl debug [–t (NONE | ERR | WRN | NOTICE | INFO)] [–f (RARE | COMMON)]
fw ctl kdebug [–i <file> | [–f] –o <file>] [–b <buffer size>] [–t | –T] [–p fld1[,fld2...] [–m <num> [–s <size>]]

---

**Example :**

```
# fw ctl debug 0                                          // Setting kernel debug default options
if NG : # fw ctl debug –buf 16000                         // Setting kernel debug buffer
if NGX : # fw ctl debug –buf 32000                        // Setting kernel debug buffer
# fw ctl debug –m fw + drop conn vm packet machine        // Setting kernel debug flags
# fw ctl debug –m fw + ld   <===  ONLY if the load on the gateway is not high
# fw ctl debug –m VPN all                                 // VPN debug
# fw ctl debug –m cluster all                             // Cluster debug
# fw ctl debug –m h323 all                                // H.323 debug
# fw ctl debug –m CPAS all                                // CPAS debug
# fw ctl debug –m WS all                                  // Web Intelligence debug
# fw ctl debug –m FG-1 all                                // FloodGate debug (QoS)
# fw ctl debug –m interspect all                          // Interspect debug
# fw ctl debug –m uag all                                 // User Authority debug
# fw ctl debug –m RTM all                                 // Real-Time Monitoring debug
# fw ctl debug –m BOA all                                 // ???
if NG : # fw ctl kdebug –f > /kernel_debug.ctl 2>&1       // output file
if NGX : # fw ctl kdebug –T –f > /kernel_debug.ctl 2>&1   // output file
```

---

**Explanation for debug :**

```
fw ctl debug 0          // defaults (clears) all kernel debugging options
fw ctl debug –x         // disables all kernel debugging options :
                        // de-allocates the buffer & automatically kills "fw ctl debug" process
fw ctl debug –buf       // allocates the buffer (OS will use maximal available buffer) :
                        // MIN value 128kB ; MAX value in NG is 16MB , in VSX NGX is 16MB , in NGX is 32MB
fw ctl debug                    // displays ALL kernel modules and their flags THAT WERE TURNED ON
fw ctl debug –m                 // displays ALL kernel modules and their flags that this machine "understands"
fw ctl debug –m <module>        // displays the flags for this module THAT WERE TURNED ON
```

---

**Explanation for kdebug :**

```
fw ctl kdebug –t / –T   // in NGX ONLY - prints the timestamp (t = seconds ; T = microseconds) :
                        // helps you to synchronize packets with "fw monitor"
fw ctl kdebug –p <field> // prints specific fields :
                        // all | proc | pid | date | mid | type | freq | topic | time | ticks | tid | text | err | host
```

**New in NGX :**

fw ctl kdebug –f –o <file_name> –m <num> –s <size>

| | |
|---|---|
| file_name | = name of the output file |
| num | = maximum number of cyclic files to create |
| size | = maximum size of each cyclic file in kilobytes |

When given <size> is reached (more or less), <file_name> is renamed to <file_name.0>, and a new <file_name> is created. If <file_name.0> already exists, then <file_name> is renamed to <file_name.1>, and so on – until the <number> limit is reached (then the rotation takes place - oldest files are just deleted).

---

**# fw ctl kebug –m <module> <severity list> <subject list>**

List of debug severities:

| | |
|---|---|
| info | informational purposes only |
| warning | warnings: may affect connection behavior |
| error | errors: the connection is probably rejected |
| fatal | fatal errors: may prevent policy installation, etc. |

List of debug severities:

see the table with flags below

If you want to make sure that the firewall accepted the flags, you need to run : *fw ctl debug –m VPN*

| Flag | Explanation |
|---|---|
| cluster | cluster related events |
| comp | compression for encrypted connections |
| counters | various counters (typically for SmartView Monitor) |
| cphwd | hardware acceleration issues |
| driver | driver attachment issues |
| err | errors that should not happen |
| ifnotify | debugs notification of changes in interface status - up or down (received from OS). |
| ike | turns on all IKE kernel debug in respect to moving the IKE to the interface, where it will eventually leave and the modification of the source IP of the IKE packet, depending on the configuration. |
| init | initializes the VPN kernel and kernel data structures, when kernel is up, or when policy is installed - it will also print the values of the flags that are set using CPSET upon policy reload |
| l2tp | L2TP protocol related events |
| mem | hardware-buffer management |
| mspi | information related to creation and destruction of MSA / MSPI |
| nat | NAT issues , cluster IP manipulation (Virtual_IP-to-Member_IP and backwards) |
| packet | events that can happen for every packet, unless covered by more specific options |
| pcktdmp | dumps the encrypted / decrypted packets (before encryption / after decryption) |
| policy | events that can happen only for a special packet in a connection, usually related to policy decisions or logs / traps |
| queue | handling of Security Association (SA) queues |
| rdp | handling of RDP packets |
| ref | information regarding reference counting for MSA / MSPI when storing or deleting SAs |
| resolver | debugs the link selection table manipulation; also debugs the Certificate Revocation List (CRL), which is also part of the peer resolving mechanism |
| sas | printing of keys and SA information |
| sr | SecureClient related issues |
| tagging | sets the VPN policy of a connection according to VPN communities , VPN Policy related info |
| tcpt | TCP Tunnel (Visitor mode) related information |
| tnlmon | tunnel monitoring |
| vin | debugs IPSec NIC interaction (IPSec NIC runs on Windows only) |
| warn | warnings: may affect connection behavior |
| xl | Accelerator cards interaction (AC II / III / IV) |

**Kernel debugging options for Check Point Active Streaming module: CPAS**

If you want to make sure that the firewall accepted the flags, you need to run : *fw ctl debug –m CPAS*

| Flag | Explanation |
|---|---|
| api | interface layer messages |
| conns | detailed description of connections, and connection's limit-related messages |
| error | errors: the connection is probably rejected |
| events | event-related messages |
| ftp | messages of the FTP example server |
| glue | glue layer messages |
| http | messages of the HTTP example server |
| pkts | packets handling messages (allocation, splitting, resizing, etc.) |
| skinny | SCCP (Skinny Client Control Protocol - Cisco proprietary VoIP protocol) |
| tcp | TCP processing messages |
| tcpinfo | TCP processing messages - more detailed description |
| timer | reports of timer ticks (pours many messages, without real content) |
| warning | warnings: may affect connection behavior |

If you want to make sure that the firewall accepted the flags, you need to run : *fw ctl debug –m FG-1*

| Flag | Explanation |
|---|---|
| auth | authenticated QoS feature |
| automatch | report matching process (debug version only) |
| autosched | report scheduling process (debug version only) - a good way to report the rates on rules |
| chain | tracing each packet through FloodGate-1 points in the cookie chain |
| chainq | holding and releasing packets during critical actions (policy install / uninstall) - internal Chain Q mechanism |
| citrix | Citrix processing |
| conn | connection information and identification processing |
| dns | DNS classification mechanism |
| dom | *currently unused* |
| dns | DNS related messages |
| driver | activation of the driver and attaching to the kernel |
| drops | dropped packets due to WFRED policy |
| dropsv | dropped packets due to WFRED policy - with additional debug information (verbose version) |
| error | different error messages (default) |
| general | *currently unused* |
| install | policy installation and building internal data structure (*for future use*) |
| llq | low latency queuing |
| log | logging information |
| ls | load sharing |
| memory | memory allocation issues - memory leak and error detection |
| pkt | packet recording mechanism |
| policy | QoS policy rules matching classification mechanism |
| rates | reporting rule / connection rates - IQ Engine behaviour and status |
| registry | registry error messages |
| rtm | failures in information gathering in RTM module (SmartView Monitor) |
| tcp | TCP streaming (re-transmission detection) mechanism |
| time | *currently unused* |
| timer(s)? | reports of timer ticks (pours many messages, without real content) |
| verbose | used with other flags - for additional information |
| sched | basic scheduling information |
| url | URL and URI for QoS classification mechanism |

**Kernel debugging options for VoIP H323 module: H323**

If you want to make sure that the firewall accepted the flags, you need to run : *fw ctl debug –m h323*

| Flag | Explanation |
|---|---|
| align | VoIP debug general messages (for example, VOIP infrastructure) |
| cpas | CPAS TCP debug messages - since H323 : H225 and H245 are over TCP ; **this flag is not included when debug is run with "*all*" flag ( # fw ctl debug –m h323 all )** |
| decode | H323 decoder messages |
| error | different error messages (default) |
| h225 | H225 call signaling messages (SETUP, CONNECT, RELEASE COMPLETE, etc.) |
| h245 | H245 control signaling messages (OPEN LOGICAL CHANNEL, END SESSION COMMAND, etc.) |
| init | used for internal errors |
| ras | H225 RAS messages (REGISTRATION, ADMISSION, and STATUS REQUEST / RESPONSE) |

**Kernel debugging options for User Authority module: uag**

If you want to make sure that the firewall accepted the flags, you need to run : *fw ctl debug –m uag*

| Flag | Explanation |
|---|---|
| driver | information about UAG, such as IOCTL, connection, NAT |
| error | errors: the connection is probably rejected |
| uag_forward_ip | NULL pointer |
| uag_api_client | *currently unused* - NULL pointer |

• SmartDefense:
  - Network Security:
    - Port scanning issues: **portscan**
    - SynDefender: **synatk**
    - Packet Validation (Packet Sanity, Max Ping Size, Small PMTU, SequenceVerifier): **packval**
  - Application Intelligence:
    - Mail: **mail, smtp**
    - Citrix: **citrix**
    - TFTP: **tftp**
    - DNS: **domain**
    - MS-SQL: **sql**
    - Micorsoft Networks | CIFS: **cifs**
    - MSN over MSNMS: **sip, msnms**
  - Other:
    - Logging: **dynlog, log**
    - Dynamic List/ SAM: **sam**
    - Quarantine: **quarantine**
  - Streaming
    - **tcpstr** (Passive Streaming)
    - Debugging Module CPAS: `fw ctl debug -m CPAS + error warning tcp http`
    - **spii** (INSPECT Streaming)
  - Various Packet Processing: **packet, chain**
  - Policy Installation: **filter, install, asm**
  - Memory: **memory**
  - Zone Policy: **bridge, sam**

• Web Intelligence:
  - Many options are available. Some examples of options:
  - Policy installation: **spii, policy, module**
  - Connection management: **connection, session**
  - Request/Response parsing: **parser, body**
  - Reject / Defense: **policy, body, report_mgr**
  - Adding a capture to the debug: **pkt_dump, address, timestamp**