= Fortigate debug and diagnose commands complete cheat sheet :homepage: https://yurisk.info :source-highlighter: rouge :toc:

Author: Yuri Slobodyanyuk, https://yurisk.info

NOTE: To enable debug set by any of the commands below, you need to run *diagnose debug enable*. This is assumed and not reminded any further. Use *dia debug info* to know what debug is enabled, and at what level.

NOTE: To disable and stop immediately any debug, run *dia deb res* which is short for *diagnose debug reset*.

NOTE: All debug will run for 30 minutes by default, to increase use `diagnose debug duration <minutes>`, setting to 0 means unlimited by time. Reboot will reset this setting.

== Security rulebase debug (diagnose debug flow) .Security rulebase diagnostics with `diagnose debug flow` [cols=2, options="header"] |=== |Command |Description

|*diagnose firewall iprope lookup <src IP> <src port> <dst IP> <dst port> <IANA protocol number> <src interface>* |Policy lookup for any combination of IPs and ports - use to see what policy (if any) matches traffic between specific IP addresses and ports. E.g. `dia firewall iprope lookup 10.10.10.1 34567 8.8.8.8 443 6 LAN1`

|*diagnose debug flow filter* |Show the active filter for the flow debug

|*diagnose debug filter clear* |Remove any filtering of the debug output set

|*diagnose debug flow filter <filtering param> / dia debug flow filter6 <param>* | Set filter for security rulebase processing packets output. You can set multiple filters - act as AND, by issuing this command multiple times. Parameters:

`vd` - id number of the vdom. When entering the vdom with `edit vdom`, this number is shown first.

`vd-name` - limit debug to specific VDOM by its name. Fortigate translates the name to VDOM ID (vd).

`proto` - Protocol number.

`addr` - IP address of the packet(s), be it a destination or/and a source.

`saddr` - IP source address of the packet(s).

`daddr` - IP destination address of the packet(s).

`port` - Source or/and destination port in the packet(s).

`sport` - Source port of the packet(s).

`dport` - Destination port of the packet(s).

`negate <parameter>` - negate the match, i.e. match if a packet does NOT contain `<parameter`. Where `parameter` is one of the above: `vd`, `addr`, `saddr`, `port`, `sport`, `dport`

|*diagnose debug filter6 <parameter>* | Same as `diagnose debug filter` but for IPv6 packets. The rest of matching and conditions remain of the same syntax.

|*diagnose debug flow show function-name enable* |Show function names responsible for each step in processing.

|*diagnose debug flow trace start [number]* |Actually start the debug with optional `number` to limit number of packets traced.

|*diagnose debug flow trace start6 [number]* |Start the debug trace for IPv6 traffic, with optional `number` to limit number of packets traced.

|===

== Packet Sniffer (diagnose sniffer packet)

[cols=2, options="header"] |=== |Command |Description

|*dia sni pa if-name/any 'tcpdump syntax filter' verbosity count time-format* a| Network level packet sniffer like tcpdump/tshark/wireshark, presenting captured packets on CLI. It gives definite answers whether a packet reached the Fortigate, whether it was dropped by firewall rules, what was incoming/outgoing interface, and contents of the packet if needed.

`verbosity` - level of detail to present, can be one of:

1 - packets' header, includes IP addresses, ports, and flags if set.

2 - packets' header and data for IP packet, i.e. same as above plus contents of the packet.

3 - same as 2 above plus Ethernet header.

4 - packets' header (no contents) plus incoming/outgoing interface name for each packet. This gives the indication whether the packet passed the Fortigate or was dropped by it.

5 - same data as 4 plus contents of IP packets.

6 - packets' header starting from Ethernet plus contents and incoming/outgoing interface names.

`count` - number of packets to capture, integer. If not set, will be capturing until the SSH/console timeout or until stopped with `CTRL + C`.

`time-format`:

- a - absolute UTC time
- `l` - local time
- *default* - relative to the start of sniffing in seconds.milliseconds.

|*IPv6* |For IPv6 traffic, the command is the same, but use the relevant `filter` clauses instead, e.g. `host 2001:db8::1` or `net 2001:db8::/64` or `icmp6`.

|*set auto-asic-offload disable* |You may need to temporarily disable NPU hardware acceleration offloading, to see accelerated packets. You do so inside a specific firewall policy. This will cause all packets passing on this policy rule to be processed by CPU and thus make packets visible to the sniffer. This may increase the CPU load. E.g. `config firewall policy`, `edit 1`, `set auto-asic-offload disable`. Do not forget to turn it on again: `set auto-asic-offload enable`.

|===

== General Health, CPU, and Memory .General Health, CPU, and Memory loads [cols=2, options="header"] |=== |Command |Description

|*get sys stat* |Get statistics about the Fortigate device: FortiOS used, license status, Operation mode, VDOMs configured, last update dates for AntiVirus, IPS, Application Control databases.

|*get sys performance stat* |Show real-time operational statistics: CPU load per CPU, memory usage, average network/session, uptime.

|*diagnose sys top [refresh] [num-of-processes] [iterations]* |Print list of running processes updated every *refresh* seconds (default 5), for *iterations* times, sorted in descending order by the CPU load. This `top` command does not display all processes by default, to show them all, set *num-of-processes* to high number, for example 100. Press "m" to sort the processes by memory consumption. The displayed table is in this order: `Process id`, `process state: (R)unning, (S)leep, (Z)ombie, (D)isk Sleep, < Means higher priority`, `CPU used`, `Memory used`.

|*dia sys kill signal-id process-id* |Forcefully kill the process with the id of *process-id*, sending it the given *signal-id* (Linux signals, e.g. 9, 11).

|*diagnose debug crashlog read* | Display crash log. Records all daemons crashes and restarts. Some daemons are more critical than others.

|*diagnose debug crashlog clear* | Clear the crash log.

|*dia sys top-mem [num-processes] [detail]* |Show top (default 5) processes by memory usage, optionally set number of processes to show with *num-processes*, and use `detail` to get verbose output (a lot).

|*get hardware memory* | Show memory statistics: free, cached, swap, shared

|*dia hardware sysinfo conserve* |Info whether the conserve mode on or off, total memory available, conserve mode thresholds `red` and `green`

|*execute sensor list* |List current readings of all sensors present on this model of the Fortigate. Larger models (1500 and up) show CPUs voltage, fan speeds, temperature, power supply voltage and more.

|*dia sys flash list* |Show contents of the flash memory holding FortiOS firmware images. One of the images will have `Active` set to `yes`, which means it is the used one.

|*diagnose hardware deviceinfo disk* |Show all storage attached to the firewall, including disk type, volume, free space.

|===

== Session stateful table

[cols=2, options="header"] |=== |Command |Description

|*get system session status / get system session6 status* |Show current number of sessions passing the Fortigate (IPv4/IPv6). Run inside the VDOM in multi-vdom environment to get number of connections /sessions for this specific VDOM.

|*get sys session-info statistics* | Get general statistics on sessions: current number of, global limits, number of clashes (different sessions trying to use the same ports), TCP sessions stats per state

|*get sys session-info ttl* |Show the default TTL setting for the connections in the table, default being 3600 seconds.

|*diagnose sys session filter <filter parameter> <filter value> / diagnose sys session6 filter <filter parameter> <filter value>* | Set filter to show/manipulate only specific connections in the stateful table. Run without any filter parameters this command displays the current filter applied if any. Parameters:

`vd` - id number of the vdom. When entering the vdom with edit vdom, this number is shown first.

`sintf` - source interface.

`dintf` - destination interface.

`proto` - protocol, by IANA protocol number.

`proto-state` - protocol state.

`src` - source IP.

`dst` - destination IP.

`nsrc` - NATed source IP.

`sport` - source port.

`nport` - NATed source port.

`dport` - destination port.

`policy` - policy id.

`duration <from> <to>` - duration.

`expire <from> <to>` - expiration time.

`session-state1 <x>` - session state, where *x* is in hex, state bits.

`negate <parameter>` - negate the match, i.e. match if a connection does NOT contain *parameter*. Where parameter is one of the mentioned above.

|*diagnose sys session clear / dia sys session6 clear* |Clear/delete connections from the session table. IMPORTANT: If no session filter is set (see above) before running this command, ALL connections passing the Fortigate will be deleted! Which means they will be disconnected. So use carefully.

|*diagnose sys session list / dia sys session6 list* |List connections limited to the filter set if any, or all session table if not.

|===

== High Availability Clustering debug .HA Clustering related debug and verification [cols=2, options="header"] |=== |Command |Description

|*get sys ha status* |Show general status and statistics of the clustering - health status, cluster uptime, last cluster state change, reason for selecting the current master, configuration status of each member (`in-`

`sync/out-of-sync`), usage stats (average CPU, memory, session number), status (`up/down`, `duplex /speed`, `packets received/dropped`) for the heartbeat interface(s), HA cluster index (used to enter the secondary member CLI with `exe ha manage`).

|*diagnose sys ha dump-by group* | Print detailed info per cluster group, shows actual uptime of each member in `start_time`, as well monitored links failures, status.

|*diagnose sys ha checksum cluster* |Shows configuration checksum for each cluster member separated in individual VDOMs and *global*. In properly synchronized cluster all member checksums should be identical, look at `all` value.

|*diagnose sys ha checksum recalculate* |Force cluster member to recalculate checksums, often will solve the out of sync problem. No adverse effects. Run on each cluster member.

|*diagnose sys ha checksum show <VDOM/global>* |Print detailed synchronization status for each configuration part. Use after seeing `out-of-sync` in *diagnose sys ha checksum cluster* to know which part of configuration causes members to be out-of-sync. Need to run on each cluster member and compare, long output - use `diff/vimdiff/Notepad++ Compare plugin` to spot the differences.

|*diagnose sys ha checksum show <VDOM/global> <settings part name>* |Show exact setting inside the settings tree that causes out-of-sync. Use output from *diagnose sys ha checksum show* (see above) for *settings part name*. E.g. if `diagnose sys ha checksum show root` indicates that *firewall.vip* is out-of-sync, running `diagnose sys ha checksum show root firewall.vip` will give checksums of each VIP in the root domain to compare with those of secondary member.

|*diagnose debug app hatalk -1* |Enable heartbeat communications debug. It shows in real time if members are talking over sync interfaces. The output will look like `state/chg_time /now=2(work)/1610773657/1617606630`, where the desired `state` is *work*, *chg_time* is last cluster state/failover date in epoch, and *now* is the last time communication occurred on heartbeat interface(s), also in epoch.

|*diag debug application hasync -1* |Real time synchronization between members. As only things that changed get synchronized after 1st sync is established, may take time to produce output. See next.

|*execute ha synchronize stop*

diag debug enable

diag debug application hasync -1

execute ha synchronize start

|Stop, enable debug, then start again HA synchronization process, will produce lots of output.

|*exe ha manage ?*

exe ha manage <id>

|First show index of all Fortigate cluster members, then enter any secondary member CLI via its index.

|*diagnose sys ha reset-uptime* a| Resets uptime of this member making it less than the other member(s)'s uptime and so fails over to those member(s). This is a temporary way to force cluster fail-over to

another member from the current one. NOTE: check that the setting below is present or immediately after the reset and failover, this member will become active again if it has higher HA priority.

---

config sys ha set ha override disable

|===

== IPSEC VPN debug

.IPSEC VPN Debug [cols=2*,options="header"] |=== |Command |Description

| *diagnose vpn ike log-filter <parameter>* a| Filter VPN debug messages using various parameters:

- `list` Display the current filter.
- `clear` Delete the current filter.
- `name` Phase1 name to filter by.
- `src-addr4/src-addr6` IPv4/IPv6 source address range to filter by, usually you filter on Remote peer legal IP.
- `dst-addr4/dst-addr6` IPv4/IPv6 destination address range to filter by.
- `src-port` Source port range
- `dst-port` Destination port range
- `vd` Index of virtual domain. -1 matches all.
- `interface` Interface that IKE connection is negotiated over.
- `negate` Negate the specified filter parameter.

|*diagnose debug application ike -1* | Enable IPSec VPN debug, shows phase 1 and phase 2 negotiations (for IKEv1) and everything for IKEv2. "-1" sets the verbosity level to maximum, any other number will show less output.

|*diagnose vpn ike gateway flush name <vpn_name>* |Flush (delete) all SAs of the given VPN peer only. Identify the peer by its Phase 1 name.

|*diagnose vpn tunnel list [name <Phase1 name>]* | Show operational parameters for all or just specific tunnels: Type (dynamic dial up or static), packets/bytes passed, NAT traversal state, Quick Mode selectors/Proxy Ids, mtu, algorithms used, whether NPU-offloaded or not, lifetime, DPD state.

|*diagnose vpn ike gateway list* | Show each tunnel details, including user for XAuth dial-up connection.

|*dia vpn tunnel shut <Phase2 name> <Phase1 name>* |Bring the named tunnel down by its Phase2 and Phase2 name.

|*dia vpn tunnel up <Phase2 name> <Phase1 name>* |Bring the tunnel up by its name.

|*get vpn ipsec tunnel details* | Detailed info about the tunnels: Rx/Tx packets/bytes, IP addresses of the peers, algorithms used, detailed selectors info, lifetime, whether NAT Traversal is enabled or not.

|*get vpn ipsec stats tunnel* | Short general statistics about tunnels: number, kind, number of selectors, state

|*get vpn ipsec tunnel summary* | Short statistics per each tunnel: number of selectors up/down, number of packets Rx/Tx.

|*get vpn ipsec stats crypto* | Crypto stats per component (ASIC/software) of the Fortigate: encryption algorithm, hashing algorithm. Useful to see if unwanted situation of software encryption/decryption occurs.

|===

== SSL VPN debug .SSL VPN client to site/Remote Access debug [cols=2, options="header"] |=== |Command |Description

|*get vpn ssl monitor* |List logged in SSL VPN users with allocated IP address, username, connection duration.

|*diagnose vpn ssl debug-filter criteria* |Limit debug output according to the *criteria* below:

`src-addr4\|src-addr6` *source-ip-of-client* Source IP of the connecting client

`vd` *VDOM name* Limit debug to a specific VDOM, specify VDOM by its string name, not numerical index.

`negate` Negate the filter.

`clear` Clear the filter.

`list` List active filter.

|*diagnose debug app sslvpn -1* |Debug SSL VPN connection. Shows only SSL protocol negotiation and set up. That is - ciphers used, algorithms and such, does NOT show user names, groups, or any client related info.

|===

== Static Routing Debug

.Static and Policy Based Routing debug & diagnostics [cols=2,options="header"] |=== |Command |Description

|*get router info kernel*

get router info6 kernel

a|View the kernel routing table (FIB). This is the list of resolved routes actually being used by the FortiOS kernel.

`tab` Table number, either 254 for unicast or 255 for multicast.

`vf` Virtual domain index, if no VDOMs are enabled will be 0.

`type` 0 - unspecific, 1 - unicast, 2 - local , 3 - broadcast, 4 - anycast , 5 - multicast, 6 - blackhole, 7 - unreachable , 8 - prohibited.

`proto` Type of installation, i.e. where did it come from: 0 - unspecific, 2 - kernel, 11 zebOS module, 14 - FortiOS, 15 - HA, 16 - authentication based, 17 - HA1

`prio` priority of the route, lower is better.

`pref` preferred next hop for this route.

`Gwy` the address of the gateway this route will use

`dev` outgoing interface index. If VDOMs enabled, VDOM will be included as well, if alias is set it will be shown.

|*get router info routing-table all*

get router info6 routing

|Show RIB - active routing table with installed and actively used routes. It will not show routes with worse priority, multiple routes to the same destination if unused.

|*get router info routing database*

get rotuer info6 routing database |Show ALL routes, the Fortigate knows of - including not currently used.

|*get router info routing-table details <route>* | Show verbose info about specific route, e.g. `get router info routing-table details 0.0.0.0/0`

|*diagnose ip rtcache list* | Show the routes cache table.

|*get firewall proute*

get firewall proute6 | Get all configured Policy Based Routes on the Fortigate.

| *exe traceroute-options [source ip / device ifname / view-settings / use-sdwan yes]*

exe traceroute host | Run traceroute, setting various options if needed.

|*exe tracert6 [-s source-ip] host* | Run IPv6 trace route.

|*exe ping-options* [data-size *bytes* / df-bit / interface *if-name* / interval *seconds* / repeat-count *integer* / reset / view-settings / timeout *seconds* / source *ip* / ttl *integer* / use-sdwan yes] | Set various options before running pings.

|*exe ping host* |Run the IPv4 ping.

|*exe ping6-options see available options above for ipv4* |Set various ping6 options before running it.

|*exe ping6 host* |Run the IPv6 ping.

|===

== Interfaces

.Interafces of all kinds diagnostics [cols=2,options="header"] |=== |Command |Description

|*get hardware nic <inerface name>* |Hardware info of the interface: MAC address, state (up/down), duplex (full, half), Rx/Tx packets, drops.

|*diagnose hardware deviceinfo nic <nic name>* |Same as above.

|*get sys interface transceiver* |List all SFP/SFP+ transceivers installed with info on: vendor name, serial number, temperature, voltage consumed, and, most important - Transmit (TX) and Receive (RX) signal power in dBm.

|*get hardware npu np6 port-list* |Show on which interfaces the NPU offloading is enabled.

|*diagnose npu np6lite port-list* | Same as above but for NP6-lite.

|*fnsysctl ifconfig <interface name>* |Gives the same info as Linux `ifconfig`. The only way to see the actual MTU of the interface.

|*fnsysctl cat /proc/net/dev* |Similar to `netstat` shows errors on the interfaces, drops, packets sent /received.

|*diagnose ip address list* |Show IP addresses configured on all the Fortigate interfaces.

|*diagnose sys gre list* | Show configured GRE tunnles and their state.

|*diag debug application pppoed -1*

dia debug application pppoe -1

dia debug applicaiton ppp -1

|Enable all ADSL/PPPoE-related debug.

|*execute interface pppoe-reconnect* |Force ADSL re-connection.

|*diagnose sys waninfo* |Show WAN interface info: public IP address of the WAN interface, guessed geo location of this IP, and whetehr this IP address is in FortiGuard black list.

|===

== LACP Aggregate Interfaces

[cols=2, options="heade"] |=== |Command |Description

|*diagnose netlink aggregate list* |List all aggregate interfaces in the current VDOM, shows names, state (up/down), LACP mode and algorithm used

|*diagnose netlink aggregate name <aggregate interface name>* |Shows details of the given aggregate interface under the entry `actor state` (preferred state is *ASAIEE*): LACP Mode (Active/Passive), LACP Speed mode (Slow [default]/Fast), Synced or Out of Sync, minimal physical interfaces to be up for the whole aggregate to be up, Aggregator ID (has to be identical on both sides), own and peer's MAC addresses, link failure count.

|*diagnose sniffer packet any "ether proto 0x8809" 6 0 a* |Sniffer to see all LACP traffic on this Fortigate: `0x8809` LACP Ethernet protocol designation, `6` - maximum verbosity, `0` - do not limit number of captured packets, `a` - show time in UTC format, rather than delta from the 1st packet seen. LACP packets should arrive from the peer's MAC address on the aggregate logical interface name, and should leave from the physical interface(s) destined to the peer's MAC address. This capture will also show LACP actor state in arriving/leaving packets - for working LACP aggregate it should be `ASAIEE` in both directions.

|*diagnose netlink port <aggregate int name> src-ip <IP> dst-ip <IP>* |Show what physical port a packet given by the filter will exit. Available filter keywords:

`src-ip` - Source IP address.

`dst-ip` - Destination IP address.

`src-mac` - Source MAC address.

`dst-mac` - Destination MAC.

`proto` - Protocol number.

`src-port/dst-port` - Source/Destination port.

`vlan-id` - VLAN number.

|===

== DHCP server, relay, client

.DHCP server, relay, client [cols=2, options="header"] |=== |Command |Description

|*show system dhcp/dhcp6 server* |Show DHCP server configuration, including DHCP address pools.

|*execute dhcp/dhcp6 lease-list [interface name]* |Show real-time list of allocated by Fortigate addresses via DHCP. It will show IP address of each client, its MAC address, device type/name (Android, iOS, Windows, etc.), the lease time and expiration.

|*execute dhcp/dhcp6 lease-clear all/start-end-IP-address-range* |Clear DHCP allocations on the Fortigate. This will NOT cause clients that already have IP addresses to release them, but will just clear Fortigate DHCP database and will start over allocating again. You can either clear *all* IP addresses in the database, or only specific IPs.

|*diagnose debug application dhcps/dhcp6s -1* |Enable real-time debug of DHCP server activity. This will show DHCP messages sent/received, DHCP options sent in each reply, details of requesting hosts.

|*diagnose debug application dhcprelay/dhcp6r -1* |Enable real-time debug of the DHCP relay agent, `dhcp6r` is for DHCPv6.

|*diagnose debug application dhcpc/dhcp6c -1* |Enable real-time debug when Fortigate is itself a DHCP Client.

|*dia sni pa any 'port 67 or port 68' 6* and for DHCPv6 *dia sni pa any 'port 546 or port 547' 6* |Run packet sniffer for DHCP or DHCPv6 packets reaching the Fortigate.

|===

== NTP debug

.NTP daemon diagnostics and debug [cols=2,options="header"] |=== |Command |Description

|*diag sys ntp status* |Current status of NTP time synchronization. Shows all NTP peers and their detailed info: reachability, stratum, clock offset, delay, NTP version.

|*execute date* | Show current date as seen by Fortigate.

|*exec time* | Show current time as seen by Fortigate.

|===

== SNMP daemon debug

.SNMP daemon debug [cols=2, options="header"] |=== |Command |Description

|*diagnose debug application snmpd -1* |ENable SNMP daemon messages debug.

|*show system snmp community* |Show SNMP community and allowed hosts configuration

|===

== BGP

.BGP debug [cols=2*,options="header"] |=== |Command |Description

|*diagnose ip router bgp level info*

diagnose ip router bgp all enable

| Set BGP debug level to INFO (the default is ERROR which gives very little info) and enable the BGP debug.

|*exec router clear bgp all* | Disconnect all BGP peering sessions and clear BGP routes in BGP table and RIB. Use with care, involves downtime.

|*get router info bgp summary* | State of BGP peering sessions with peers, one per line.

|*get router info bgp network <prefix>* | Detailed info about <prefix> from the BGP process table. Output includes all learned via BGP routes, even those not currently installed in RIB. E.g. `get router info bgp network 0.0.0.0/0`. The <prefix> is optional, if absent shows the whole BGP table.

|*get router info routing-table bgp* | Show BGP routes actually installed in the RIB.

|*get router info bgp neighbors* | Detailed info on BGP peers: BGP version, state, supported capabilities, how many hops away, reason for the last reset.

|*get router info bgp neighbors <IP of the neighbor> advertised-routes* | Show all routes advertised by us to the specific neighbor.

|*get router info bgp neighbors <IP of the neighbor> routes* | Show all routes learned from this BGP peer. It shows routes AFTER filtering on local peer, if any.

|*get router info bgp neighbors <IP of the neighbor> received-routes* | Show all received routes from the neighbor BEFORE any local filtering is being applied. It only works if `set soft-reconfiguration enable` is set for this peer under `router bgp` configuration.

|*diagnose sys tcpsock | grep 179* | List all incoming/outgoing TCP port 179 sessions for BGP.

|===

== OSPF

[cols=2,options="header"] |=== |Command |Description

|*get router info ospf status* |Info about OSPF for the whole Fortigate: Router ID, Hello timer, stats of LSA originated/received, OSPF Areas configured, number of neighbors for each Area, whether authentication is enabled per Area.

|*get router ospf* |Show all general OSPF process settings, default and not.

|*get router info ospf neighbor* |Get info on all neighbors of this Fortigate - their IPs, state, Dead Interval timers.

|*get router info ospf interface inerface-name* |Show OSPF info for a given interface - Area, Router ID, timers, DR/BDR for broadcast nets, MTU as seen by OSPF.

|*get router info ospf route* |Show OSPF routes installed into RIB.

|*diagnose ip router ospf level info* |Set OSPF debug level to the highest - info. A must before running OSPF debug to see relevant information.

|*diagnose ip router ospf option* |Enable specific debug option. If in doubt just use `dia ip router ospf all -1` to enable all OSPF debug options. Some options expect integer as info level as the last parammeter, some options need `enable` as the last parameter. Options:

`all` *n* - Enable all OSPF debug, *n* is the info level, for all set to `-1`.

`lsa` - OSPF Link State Advertisement, sending/receiving LSAs.

`nfsm` - OSPF Neighbor State Machine, not very useful.

`packet` - OSPF Packets. Show OSPF packets traffic.

`events` - OSPF events.

`ifsm` - OSPF Interface State Machine. Not very useful, use `get router info ospf interface` instead.

`nfsm` - OSPF Neighbor State Machine.

`nsm` - OSPF NSM interface.

`route` - OSPF route information.

`show` - Show status of OSPF debugging.

|===

== Admin sessions .Admin sessions management [cols=2,options="header"] |=== |Command |Description

|*get sys info admin status* |List logged in administrators showing `INDEX` value for each session

|*execute disconnect-admin-session <INDEX>* |Disconnect logged in administrator by the session INDEX.

|===

== Authentication .Authentication in all kinds LDAP, Radius, FSSO [cols=2, options="header"] |=== |Command |Description

|*diagnose firewall auth list* |List all authenticated and known by firewall usernames. It does not matter what the source is - LDAP/SSO/etc. Also shows client's IP, idle time, duration.

|*diagnose debug app fnbamd -1* |Enable debug for authentication daemon, valid for ANY remote authentication - RADIUS, LDAP, TACACS+.

|*diagnose test authserver ldap <LDAP server name in FG> <username> <password>* | Test user authenticaiton on Fortigate CLI against Active Directory via LDAP. E.g. test user `Tara Addison` against LDAP server configured in Fortigate as `LDAP-full-tree` having password `secret`: `diagnose test authserver ldap LDAP-full-tree "Tara Addison" secret`.

|*diagnose test authserver radius <RADIUS server object> <chap/pap/mschap2> <username> <password>* | Test user authentication against the configured RADIUS server. E.g. to test user `adminad1` with password `secr3t` against RADIUS server RAD1: `diagnose test authserver radius RAD1 mschap2 adminad1 secr3t` Output: `authenticate 'adminad1' against 'mschap2' succeeded, server=primary assigned_rad_session_id=9839905755137 session_timeout=0 secs idle_timeout=0 secs!`

|*diagnose debug authd fsso list* |List logged in users the Fortigate learned via FSSO

|*diagnose debug authd fsso server-status* | Show status of connections with FSSO servers. Note: it shows both, local and remote FSSO Agent(s). The local Agent is only relevant when using Direct DC Polling, without installing FSSO Agent on AD DC, so it is ok for it to be `waiting for retry ... 127.0.0.1` if you don't use it. The working state should be `connected`.

|*diagnose debug authd fsso refresh-logon* |Refresh user login information.

|*diagnose debug authd fsso refresh-group* |Refresh groups and groups memberships info.

|===

== Web and URL Filtering debug

.Web Filtering debug [cols=2, options="header"] |=== |Command |Description

|*diagnose debug rating* |Shows if the Fortiguard-based Web filtering is enabled, the contract /subscription status, whether the Anycast of Unicast used, IPs of Fortigaurd servers and the communication status.

|*diag autoupdate versions* |Show current databases versions and the last time they were updated.

|*execute update-now* |Force download and update of all the databases (AV, IPS, etc.)

|*diagnose debug application update -1*

dia debug enable

execute update-now |Enable updates debug and then force the updates.

|execute ping *service.fortiguard.net*

execute ping *update.fortiguard.net*

execute ping *guard.fortinet.net* |Check DNS resolving of the Fortiguard servers names, and connectivity to them (note: those servers usually do answer pings, but it may change any time).

|*diag webfilter fortiguard cache dump* |Show cache of web sites ratings responses from the Fortiguard.

|*dia test app urlfilter 2* |Clear the cache, no downtime.

|*dia test app urlfilter 99* |Restart the URL filtering daemon, causes short downtime.

|*dia deb app urlfilter -1* |Enable URL filering daemon debug, showing all processing steps, a lot of output as shows everything.

|===

== Fortianalyzer logging debug .Verify and debug sending logs from Fortigate to Fortianalyzer [cols=2, options="header"] |=== |Command |Description

|*get log fortianalyzer setting* |Show active Fortianalyzer-related settings on Fortigate.

|*config log fortianalyzer* |Complete Fortianalyzer configuration on CLI, as GUI configuring is usually not enough for it to work.

|*get log fortianalyzer filter* |Verify if any log sending filtering is being done, look for values of `filter` and `filter-type`. If there are any filters, it means not all logs are sent to FAZ.

|*exec log fortianalyzer test-connectivity* |Verify that Fortigate communicates with Fortianalyzer. Look at the statistics in `Log: Tx & Rx` line - it should report increasing numbers, and make sure the status is `Registration: registered`.

|*exec telnet <IP of Fortianalyzer> 514* |Test connectivity to port 514 on the Fortianalyzer. If pings are allowed between them, you can also try pinging.

|*diagnose sniffer packet any 'port 514' 4* |Run sniffer on Fortigate to see if devices exchange packets on port 514. Click in GUI on `Test Connectivity` to initiate connection.

|===

== SD-WAN verification and debug .SD-WAN verification and debug [cols=2, options="header"] |=== |Command |Description

|*diagnose sys sdwan health-check* (6.4 and newer)

diagnose sys virtual-link health-check (5.6 up to 6.4)

| Show state of all the health checks/probes. Successful probes are marked `alive`, failed probes are marked dead. Also displays `packet-loss, latency, jitter` for each probe.

|*diagnose sys sdwan member*

diagnose sys virtual-wan-link member (5.6 up to 6.4)

|Show list of SD-WAN zone/interface members. Also gives each interface gateway IP (if was set, 0.0.0.0 if not), `priority`, and `weight` both by default equal `0`, used with some SLA Types.

|*diagnose sys sdwan service*

diagnose sys virtual-wan-link service (5.6 up to 6.4)

|List configured SD-WAN rules (aka `services`), except the Implied one which is always present and cannot be disabled, but is editable for the default load balancing method used. Shows member interfaces and their status `alive` or `dead` for this rule.

|*diag sys sdwan intf-sla-log <interface name>*

diag sys virtual-wan-link intf-sla-log <interface name> (5.6 up to 6.4)

|Print log of <interface name> usage for the last 10 minutes. The statistics shown in bps: `inbandwidth`, `outbandwidth`, `bibandwidth`, `tx bytes`, `rx bytes`.

|*diag netlink interface clear <interface name>*

|Clear traffic statistics on the interface, this resets statistics of the SD-WAN traffic passing over this interface. Needed, if, for example, you changed SD-WAN rules, but not sure if it's already active. E.g. `diag netlink interface clear port1`.

|*diagnose firewall proute list* |List ALL Policy Based Routes (PBR). SD-WAN in Fortigate, after all, is implemented as a variation of PBR. This command lists manual (classic) PBR rules, along with SD-WAN created via SD-WAN rules. *Important*: Manually created PBR rules (via `Network -> Policy Routes` or on CLI `config route policy` always have preference over the SD-WAN rules, and this command will show them higher up.

|===

== Virtual Fortigate License Status .Verify status of VM Fortigate License [cols=2, options='header"] |=== |Command |Description

|*get sys status | grep -i lic* |Get status of the license (for VM only). The corect status is `Valid`.

|*diagnose debug vm-print-license* | Show detailed info on VM Fortigate license status: allowed CPUs and memory, date of license activation, license expiration date (if set), serial number.

|*diagnose hardware sysinfo vm full* |Show license data as seen by FortiGuard: status (should be `valid=1`), last time it was checked (`recv`), answer code, should be `code: 200`, `code: 401` is for duplicate license found, `code: 502` is for VM cannot connect to FortiGuard, and `code: 400` is for invalid license.

|===

== SIP ALG and helper .SIP proxy or helper debug [cols=2, options="header"] |=== |Command |Description

|*config sys settings*

get | grep alg

|Show the current SIP inspection mode. If the output is `default-voip-alg-mode: proxy-based` then the full Layer 7 proxy SIP inspection is on (*ALG* inspection). If the output is `default-voip-alg-mode: kernel-helper-based` then the Layer 4 *helper* inspection is on. In both modes Fortigate does IP address translation inside SIP packets (if needed), and opens dynamically high ports for incoming media/voice streams ports. In *ALG* mode, the Fortigate additionally does RFC compliance verification and more. So, the *ALG* mode is more prone to cause issues but also provides more security.

|*show system session-helper | grep sip -f* |If using SIP *helper* and not *ALG*, make sure there is an entry for SIP in the helpers list, usually on port 5060, but may be custom as well.

|*diagnose debug application sip -1* |Display SIP debug in real-time (lots of output). It shows IP replacement inside SIP packets if NAT involved, all SIP communication requests (`REGISTER`,`INVITE` etc.), and reply codes.

|===

== DNS server and proxy debug [cols=2, options="header"] |=== |Command |Description

|*get system dns* |Show configured DNS servers, DNS cache limit and TTL, source IP used, timeout and retry, whther NDS over TLS is enabled.

|*diagnose test app dnsproxy* |Will present all debug options for dnsproxy. Belowi are some of more useful of them.

|*diagnose test app dnsproxy 2* |Show the following statatistics: number of DNS process workers (if multiple), DNS latency against each server used, Secure DNS IP and latency - DNS server used for DNS filtering and Botnet detections, DNS cache usage, UDP vs TCP requests statistics, name of DNS Filter applied if any.

|*diagnose test app dnsproxy 1* |Clear DNS responses cache

|*diagnose test app dnsproxy 3* |Display detailed statistics for each DNS/SDNS server used and those that could be used.

|*diagnose test app dnsproxy 7* |Show the responses cached entries.

|*diagnose test app dnsproxy 6|4|5* | Work with FQDN resolved objects:

`6` - Display currently resolved FQDN addresses

`4,5` - Reload/Requery all FQDN addresses

|*diagnose test app dnsproxy 8* |Show DNS database of domain(s) configured on the Fortigate itself.

|*diagnose test app dnsproxy 9* |Reload DNS database of domain(s) configured on the Fortigate itself.

|*diagnose test app dnsproxy 10* |Show active SDNS, i.e. DNS Filter Policy used. Shows Categories as numbers, so not easily readable.

|*diagnose test app dnsproxy 12* |Reload configuration of DNS Filter, in case the changes made do not take effect immediately.

|*diagnose test app dnsproxy 15* |Show cached responses and their rating of the DNS Filter for each URL /domain scanned.

|*diagnose test app dnsproxy 16* |Clear the DNS Filter responses and ratings cache.

|*diagnose test app dnsproxy 99* |Restart the dns proxy service.

|*diagnose test app dnsproxy -1* |Enable all possible debug, a lot of output.

|===

== Administrator GUI, SSH access and API automation requests debug

[cols=2, options="header"] |=== |Command |Descritption

| *diagnose debug application httpsd -1*

|Enable diagnostics for administrator and remote REST API access via `api-user`. When debugging API automation, refrain from working in admin GUI as it will produce a lot of unrelated output.

|*diagnose debug application sshd -1* |Debug SSH administrator session.

|*dia debug cli 8* |Nice trick: this will print CLI commands the Fortigate runs when you do something in the GUI. This way we can find CLI commands without long search in Google or documentation.

|===

== Wireless Controller and managed Access Points debug

[cols=2, options="header"] |=== |Command |Description

|*diagnose wireless-controller wlac -c ap-status* |Show list of all Access Points (APs) this Fortigate is aware of with their BSSID (MAC), SSID, and Status (`accepted`, `rogue`, `suppressed`)

|*diagnose wireless-controller wlac -c vap* |Show list of APs with their BSSIDs, broadcasted SSIDs, IDs, and unlike `wlac -c ap-status` above, also shows management IP and port which can be later used for real-time debug.

|*show wireless-controller wtp-profile* |Show available Wireless Termination Points (i.e. APs) profiles with their settings. Profiles are applied to individual APs, i.e. a single profile can be applied to multiple APs.

|*show wireless-controller wtp* |Show APs known to this Fortigate individually. We can enter any given AP configuration and change settings for this AP only, i.e. `set admin disable`.

|===

== FortiTokens

[cols=2, options="header"] |=== |Command |Description

|*diagnose fortitoken info* |Show all existing on the Fortigate Fortitokens, including their status:

`new` - new token, available to be assigned to a user.

`active` - normal state, assigned to a user, hardware Fortitoken.

`provisioning` - Fortitoken Mobile (FTM), assigned to a user, waits for end user to activate it on his /her mobile phone.

`provisioned` - FTM, assigned to a user and activated by him/her as well.

`provision timeout` - user hasn't activated the assigned token in the given time window (3 days default), the token needs to be re-provisioned to a user again.

`locked` - token was locked either manually by administrator, or because Fortigate was not able to reach Fortiguard servers.

|*exec ping fds1.fortinet.com*

exec ping directregistration.fortinet.com

exec ping globalftm.fortinet.net

|Verify that Fortigate can resolve and ping the FortiGuard servers responsible for FortiToken activation /license validation.

|*show user fortitoken* |Display all Fortitokens info on license number, activation expiration (in epoch format).

|===

== Automation stitches debug

[cols=2, options="header"] |=== |Command |Description

|*diag test app autod 1* |Enable automation stitches logging.

|*diag debug cli 7* |Show stitches' running log on the CLI.

|*diag debug enable* |Enable debug.

|*diagnose automation test stitch-name log-if-needed* |Run the specified *stitch name*, optionally adding log when using Log based events.

|===

== Alerts Sending debug

[cols=2, options="header"] |=== |Command |Description

|*dia debug app alertmail -1* |Enable sessions debug for sending alerts by mail. This will show the configured settings, like from/to email address, as well as SMTP session log of connecting to the remote mail server and received/sent SMTP session codes.

|===

== fnsysctl all possible options

[cols=2, options="header"] |=== |Command |Description

|*fnsysctl ifconfig [int-name]* |Show detailed information on all/specific interface - errors, MTU, and more.

|*fnsysctl ls [path]* |List files/folders in the filesystem. Useful for post-incident investigation of Fortigate compromises, looking for a given CVE indicators of compromise (IOCs).

|*fnsysctl cat <filename>* | Show contents of a file, not all files in the filesystem are accessible. Some examples:

- Show Linux kernel version of the Fortigate:

```
fnsysctl cat /proc/version
```

- Show open TCP connections to/from Fortigate itself (use `diagnose sys tcpsock \| grep 0.0.0.0` instead):

```
fnsysctl cat /proc/net/tcp
```

- Show CPU info: `fnsysctl cat /proc/cpuinfo`

- Get memory information: `fnsysctl cat /proc/meminfo`

|*fnsysctl date* | Show date in the Linux format, ignores any options.

|*fnsysctl df -h* |Show filesystem usage, useful when you have harddisk(s) attached to the Fortigate.

|*fnsysctl du* |Show directories usage, accepts following options:

`-d n` - Limit depth to n levels deep.

`-a` - Show/count files as well, not only directories.

`-s` - Show only the summary usage of all directories/files.

`-L` - Follow all symlinks

|*fnsysctl pwd* |Show current working directory. Not very useful as we don't have access to `cd` and thus cannot change directory anyway.

|*fnsysctl ps* |List running processes. Useful together with the next command `kill`` for restarting some stuck process on Fortigate. Most of the processes in Fortigate are run via Watch Dog which means killing them will shut the running process and will restart it immediately later.

|*fnsysctl kill <process-id>* |Kill a process by its ID (PID). The only option accepted is `-s N` where N is the signal number to send as per Linux.

|*fnsysctl killall <process-name>* |Kill/restart a process by name. When using `killall` it is not recorded in the crash log file (which you read with `diagnose debug crashlog read`). Not all processes can be killed with it, e.g. hasync.

|*fnsysctl mv <src> <dst>* |Move file in the filesystem. Most of the directories on the Fortigate are read-only, but some, like tmp are not. This command will ask for the adminstrator username/password explicitly.

|*fnsysctl printenv* |Print environment variables. The only environment variable I was able to catch with this was type of Terminal used.

|*fnsysctl grep <regex> <file>* |Search contents of a file/files. The usual grep options are available:

-i	Ignore case distinctions
-l	List names of files that match
-H	Prefix output lines with filename where match was found
-h	Suppress the prefixing filename on output
-n	Print line number with output lines
-q	Quiet
-v	Select non-matching lines
-s	Suppress file open/read error messages
-c	Only print count of matching lines
-A	Print NUM lines of trailing context
-B	Print NUM lines of leading context
-C	Print NUM lines of output context

|===