

Содержание

Short alias commands	3
findip <IP> – IP address search	3
findmac <MAC> – MAC address search	3
findall – all subscribers output	4
findsub <IP> – view speed limit rules	4
findfilter <IP> – view rules in the service 16	6
dropip <IP> – delete authorization status, L2 properties; reset authorization for DHCP session	6
leasesdhcpstatus – View pool utilization statistics in KEA and SSG (in SSG with VRF partitioning)	6
subping <IPinet> <IPsubs> – Ping Internet zone from the “name” of the subscriber	8
findcgnatstatus <IP> – CGNAT status view	8
healthcheck <IP> – system status view	9
bngrestart – restart fastDPI, fastPCRF to reset all sessions and apply cold parameters	9
bngstop – stop fastDPI, fastPCRF	9
bngstart – start fastDPI, fastPCRF	9
bngstatus – status fastDPI, fastPCRF	9
bngfullrestart – restart fastDPI, fastPCRF to reset all sessions and apply cold parameters with UDR cleaning	10

Short alias commands

The alias commands allow the user to use a single word or even a single character to run any command or group of commands, including options, parameters and files.

alias — view available short commands.

findip <IP> — IP address search

Matching commands:

```
if [[ $# -ne 1 ]]; then
    echo "Use: findip 'IP_ADDRESS'"
else
    PrintDate
    echo "Authorizaion status is: "
    fdpi_cli subs auth show ${1}
    echo "L2 properties are: "
    fdpi_cli subs prop show ${1}
    echo "DHCP session info: "
    fdpi_cli dhcp show ip=${1}
    echo "Accounting session is: "
    fdpi_cli pcrf acct show ip=${1}
fi
```

findmac <MAC> — MAC address search

First finds the active (non expired) IP address for that MAC address, then similar commands for findip.

Matching commands:

```
if [[ $# -ne 1 ]]; then
    echo "Use: findmac 'mac address'"
else
    PrintDate
    declare MAC="${1}"
    declare ROWS=$(fdpi_cli subs prop show mac="${MAC}" --json --strict)
    if [[ $(echo "${ROWS}" | jq -r '[]|length') != 0 ]]; then
        declare SUB_IP=$(echo "${ROWS}" | jq -r
'.[]|records[].address.n_ip')
        echo "${SUB_IP}"
        echo "Autharizaion status is: "
        fdpi_cli subs auth show "${SUB_IP}"
        echo "L2 properties are: "
        fdpi_cli subs prop show "${SUB_IP}"
    fi
fi
```

```

    echo "DHCP session info: "
    fdpi_cli dhcp show ip="${SUB_IP}"
    echo "Accounting session is: "
    fdpi_cli pcrf acct show ip="${SUB_IP}"
    return 0
else
    echo "${MAC} is not exist"
fi
unset SUB_IP
unset ROWS
unset MAC
fi

```

findall – all subscribers output

Matching commands:

```

PrintDate
declare SUBS_ACTIVE=$(fdpi_cli subs prop show active --json --strict)
if [[ $( echo "${SUBS_ACTIVE}" | jq -ajr [].records) != 'null' ]]; then
    echo -e "$(echo "${SUBS_ACTIVE}" | jq -ajr
'.[].records[]|\n\(.address.n_ip)|\(.props.vrf)|\(.props.mac)|\(.props.l2su
bs_id)|\(.props.auth_status.ts_timestamp
.absolute_time)\n")" | grep -v -e '^"' |column -t -N
IP,VRF,MAC,L2_SUBS_ID,SESSION_STARTED -s '|'
    else
        echo "No one connected subscriber."
    fi

```

findsub <IP> – view speed limit rules

Matching commands:

```

PrintDate
echo "IP Address: ${1}"
echo "Subscriber's policing rules: "
echo -e "$(fdpi_ctrl list --policing --ip "${1}" --outformat=json2 | jq
-ajr
'.lpolicings[].policing|\n'outbound'|\(.outbound.classes[0].class)|\(.outbo
und.classes[0].ceil_value) \(.ou
tbound.classes[0].ceil_unit)|\n'outbound'|\(.outbound.classes[1].class)|\(.o
utbound.classes[1].ceil_value)
\(.outbound.classes[1].ceil_unit)|\n'outbound'|\(.outbound.classes[2].class)
|\(.ou
tbound.classes[2].ceil_value)
\(.outbound.classes[2].ceil_unit)|\n'outbound'|\(.outbound.classes[3].class)
|\(.outbound.classes[3].ceil_value)

```

```

\(.outbound.classes[3].ceil_unit)|\n'outbound' |
\(.outbound.classes[4].class)|\(.outbound.classes[4].ceil_value)
\(.outbound.classes[4].ceil_unit)|\n'outbound'|\(.outbound.classes[5].class)
|\(.outbound.classes[5].ceil_value) \(.outbound.
classes[5].ceil_unit)|\n'outbound'|\(.outbound.classes[6].class)|\(.outbound
.classes[6].ceil_value)
\(.outbound.classes[6].ceil_unit)|\n'outbound'|\(.outbound.classes[7].class)
|\(.outbound.
classes[7].ceil_value)
\(.outbound.classes[7].ceil_unit)|\n'inbound'|\(.inbound.classes[0].class)|\
(.inbound.classes[0].ceil_value)
\(.inbound.classes[0].ceil_unit)|\n'inbound'|\(.inbound.c
lasses[1].class)|\(.inbound.classes[1].ceil_value)
\(.inbound.classes[1].ceil_unit)|\n'inbound'|\(.inbound.classes[2].class)|\
(.inbound.classes[2].ceil_value) \(.inbound.classes[2].ceil_uni
t)|\n'inbound'|\(.inbound.classes[3].class)|\(.inbound.classes[3].ceil_value
)
\(.inbound.classes[3].ceil_unit)|\n'inbound'|\(.inbound.classes[4].class)|\
(.inbound.classes[4].ceil_value) \(.
inbound.classes[4].ceil_unit)|\n'inbound'|\(.inbound.classes[5].class)|\(.in
bound.classes[5].ceil_value)
\(.inbound.classes[5].ceil_unit)|\n'inbound'|\(.inbound.classes[6].class)|\
.inbound
.classes[6].ceil_value)
\(.inbound.classes[6].ceil_unit)|\n'inbound'|\(.inbound.classes[7].class)|\
(.inbound.classes[7].ceil_value) \(.inbound.classes[7].ceil_unit)\n''")|grep
-v -e '^"'|co
lumn -t -N DIRECTION,CLASS,RATE_LIMIT -s '|'

    echo 'IMPORTANT! "8 bps" is equivalent to "0 bps"'

```

An example of Policing for a particular subscriber:

```

Current time: 2024-08-21T13:54:15.917Z
IP Address: 89.179.251.130
Subscriber's policing rules:
DIRECTION  CLASS  RATE_LIMIT
outbound   0      110 mbps
outbound   1      110 mbps
outbound   2       8 bps
outbound   3       8 bps
outbound   4      110 mbps
outbound   5      110 mbps
outbound   6      110 mbps
outbound   7      110 mbps
inbound    0      110 mbps
inbound    1      110 mbps
inbound    2       8 bps
inbound    3       8 bps
inbound    4      110 mbps
inbound    5      110 mbps

```

```
inbound    6      110 mbps
inbound    7      110 mbps
IMPORTANT! "8 bps" is equivalent to "0 bps"
```

On BRAS SSG from VAS Experts, the prohibition of subscriber traffic is achieved by assigning it to a certain service (different ones are used for IP, TCP, UDP), mapping this service to a certain QoS Class, and allocating 8bit bandwidth to this Class. More details about the subscriber filters implemented in this way - in HLD.

findfilter <IP> – view rules in the service 16

Allowed TCP resources and the address for the redirect.

Matching commands:

```
PrintDate
fdpi_ctrl list --service 16 --ip ${1}
```

dropip <IP> – delete authorization status, L2 properties; reset authorization for DHCP session

The next DHCP request from the subscriber will be sent for authorization); stop Accounting session.

Matching commands:

```
PrintDate
echo "DHCP session info: "
fdpi_cli dhcp reauth ip=${1}
echo "Dropping autharizaion status..."
fdpi_cli subs auth clear ${1}
echo "Dropping L2 properties..."
fdpi_cli subs prop del ${1}
echo "Accounting session is: "
fdpi_cli pcrf acct stop ip=${1}
```

leasesdhcpstatus – View pool utilization statistics in KEA and SSG (in SSG with VRF partitioning)

Matching commands:

```
KEA_CTRL_SOCKET='/var/kea4-ctrl/kea4-ctrl-socket'
CMD_STATUS='{"command":"stat-lease4-get"}'
CMD_CONFIG='{"command":"config-get"}'
sendtodhcp(){
```

```

    echo ${1} | socat - UNIX-CONNECT:${KEA_CTRL_SOCKET}

}

jsongetvalue(){
    echo ${1} | jq -r .${2}
}

jsongetcount(){
    echo ${1} | jq -r .${2} | jq 'length'
}

getstat(){
    echo $(sendtodhcp ${CMD_STATUS})
}

}

POOL_STATUS=$(jsongetvalue "$(sendtodhcp ${CMD_STATUS})"
arguments.'''result-set''')
KEA_CONFIG=$(jsongetvalue "$(sendtodhcp ${CMD_CONFIG})" arguments.Dhcp4)
SUBS_NUMBER=$(jsongetvalue "${POOL_STATUS}" rows | jq 'length')
for (( SUBS_PROP_INDEX=0; SUBS_PROP_INDEX < ${SUBS_NUMBER};
SUBS_PROP_INDEX++ )); do
    for (( j=0; j <= ${SUBS_NUMBER}; j++ )); do
        SUBNETS_NUMBER=$(jsongetcount "${KEA_CONFIG}" '''shared-
networks'''[${SUBS_PROP_INDEX}].subnet4)
        if [[ ${SUBNETS_NUMBER} > 1 ]]; then
            for (( SUBNET_INDEX=0; SUBNET_INDEX < SUBS_NUMBER;
SUBNET_INDEX++ )); do
                if [[ $(jsongetvalue "${KEA_CONFIG}" '''shared-
networks'''[${SUBS_PROP_INDEX}].subnet4[${SUBNET_INDEX}].id) != 'null' ]];
then
                    if [[ $(jsongetvalue "${KEA_CONFIG}" '''shared-
networks'''[${SUBS_PROP_INDEX}].subnet4[${SUBNET_INDEX}].id) ==
$(jsongetvalue "${POOL_STATUS}" rows[${j}][0]) ]]; th
en
                        echo 'Pool name: '$(jsongetvalue
"${KEA_CONFIG}" '''shared-networks'''[${SUBS_PROP_INDEX}].name)' Total
addresses: '$(jsongetvalue "${POOL_STATUS}" rows[${j}][1])
'Percent usage:'$(echo "($(jsongetvalue "${POOL_STATUS}"
rows[${j}][3])/$(jsongetvalue "${POOL_STATUS}" rows[${j}][1))*100" | bc -l)
                    fi
                fi
            done
        else
            if [[ $(jsongetvalue "${KEA_CONFIG}" '''shared-
networks'''[${SUBS_PROP_INDEX}].subnet4[0].id) != 'null' ]]; then
                if [[ $(jsongetvalue "${KEA_CONFIG}" '''shared-
networks'''[${SUBS_PROP_INDEX}].subnet4[0].id) == $(jsongetvalue
"${POOL_STATUS}" rows[${j}][0]) ]]; then
                    echo 'Pool name: '$(jsongetvalue "${KEA_CONFIG}"
'''shared-networks'''[${SUBS_PROP_INDEX}].name)' Total addresses: '

```

```

$(jsongetvalue "${POOL_STATUS}" rows[${j}][1
]) 'Percent usage:'$(echo "($(jsongetvalue "${POOL_STATUS}"
rows[${j}][3])/$(jsongetvalue "${POOL_STATUS}" rows[${j}][1]))*100" | bc -
l)
        fi
    fi
done
done

fdpi_cli dhcp show stat vrf

```

subping <IPinet> <IPsubs> – Ping Internet zone from the “name” of the subscriber

Matching commands:

```

if [[ "$1" == "--help" ]] || [[ "$1" == "-h" ]] || [[ $# -eq 0 ]] || [[
$# -gt 2 ]] ; then
    echo "Usage:"
    echo "subping <IP_address_in_the_internet_side>
<IP_address_of_the_subscriber>"
else
    fdpi_cli ping inet ${1} from ${2} n=5
fi

```

findcgnatstatus <IP> – CGNAT status view

Which public address is allocated, TCP and UDP session utilization statistics.

Matching command:

```

sudo fdpi_ctrl list status --service 11 --ip <ip>

```

Check the set limit on the number of subscriber sessions in the created NAT profiles:

```

sudo fdpi_ctrl list all profile --service 11

```

Check if the white IP address the subscriber is using is not 100% populated

```

sudo fdpi_ctrl list status --service 11 --profile.name CGNAT-INET | grep
puwhb=100.00

```

где CGNAT-INET - имя 11 сервиса NAT

healthcheck <IP> – system status view

Matching command:

```
echo ----PROCESS STATE----&&systemctl status fastdpi&&systemctl status fastpcrf&&echo ----LINK STATE----&&fdpi_cli dev link state show|grep -A 1 Device&& echo ----MGMT ROUTER STATE----&&birdc show status|tail -n +2&&birdc show protocols|tail -n +2&& echo ----MAIN ROUTER STATE----&&birdc show status|tail -n +2&& birdc show protocols|tail -n +2;echo ----AUTHORIZED SUBSCRIBERS----&&fdpi_cli subs auth show|tail -n 6&& echo ----DHCP SUBSCRIBERS----&&fdpi_cli dhcp show all|tail -n 1
```

bngrestart – restart fastDPI, fastPCRF to reset all sessions and apply cold parameters

Matching command:

```
systemctl stop fastdpi && systemctl restart fastpcrf && systemctl start fastdpi
```

bngstop – stop fastDPI, fastPCRF

Matching command:

```
systemctl stop fastdpi && systemctl stop fastpcrf
```

bngstart – start fastDPI, fastPCRF

Matching command:

```
systemctl start fastdpi && systemctl start fastpcrf
```

bngstatus – status fastDPI, fastPCRF

Matching command:

```
systemctl status fastdpi && systemctl status fastpcrf
```

bngfullrestart — restart fastDPI, fastPCRF to reset all sessions and apply cold parameters with UDR cleaning

```
declare -a TABLES_LIST=("policing" "profile_names" "profiles"
"vchannel_policing" "vchannel_services")
declare -a DB_BAK_DIR="/var/db/dpi/recovered"
declare -a UDR_BAK="/tmp/data.mdb"
declare -a UDR="/var/db/dpi/data.mdb"
if [ ! -d "${DB_BAK_DIR}" ]; then
    mkdir -p "${DB_BAK_DIR}"
else
    for TABLE in "${TABLES_LIST[@]"; do
        if [ -f "${DB_BAK_DIR}/dump.${TABLE}.load ]; then
            rm -f "${DB_BAK_DIR}/dump.${TABLE}.load
        fi
    done
fi

/bin/cp -f "${UDR}" "${UDR_BAK}"

for TABLE in "${TABLES_LIST[@]"; do
    mdb_dump -f "${DB_BAK_DIR}/dump.${TABLE}.load -s "${TABLE}"
/var/db/dpi
declare -a RETVAL=$?
if [ "${RETVAL}" -ne 0 ]; then
    echo "ERROR: Can not make dump of table ${TBALE} from UDR to
${DB_BAK_DIR}/dump.${TABLE}.load"
    return 1
fi
done

systemctl stop fastdpi
systemctl stop fastpcrf

/bin/rm -f /var/db/dpi/lock.mdb
/bin/rm -f "${UDR}"

for TABLE in "${TABLES_LIST[@]"; do
    mdb_load -f "${DB_BAK_DIR}/dump.${TABLE}.load /var/db/dpi
declare -a RETVAL=$?
if [ "${RETVAL}" -ne 0 ]; then
    echo "ERROR: Can not load table ${DB_BAK_DIR}/dump.${TABLE}.load
to UDR"

    /bin/rm -f /var/db/dpi/lock.mdb
    /bin/cp -f "${UDR_BAK}" "${UDR}"
    systemctl start fastpcrf
    systemctl start fastdpi
    return 1
fi
```

done

```
systemctl start fastpcrf  
systemctl start fastdpi
```