EdgeView Commands

This page is for EdgeView client side command descriptions. For information on EdgeView design and architecture, please see Design and Architecture, Graphical Workflow and FAQ.

To run EdgeView client script, the docker is required on user's laptop. The EdgeView client program is started in a script with 'docker run'. For an example of the client script, see EdgeView Client Script. The EdgeView commands are checked against EdgeView Policies to be allowed to run on devices.

Command Help

EdgeView script (e.g. edgeview.sh, your script name may vary) with the '-h' or '-help', it will display the output like:

edgeview.sh -h

eve-edgeview [-token <session-token>] [-inst <instance-id>] <query command>
query options:

[acl app arp connectivity flow if mdns nslookup ping route socket speed tcp tcpdump trace url wireless]
[app configitem cat cp datastore dmesg download du hw lastreboot ls model newlog pci ps cipher usb techsupport top volume]
log/search-pattern [-time <start_time>-<end_time> -json -type <app|dev> -line <num>]
pub/ [baseosmgr domainmgr downloader edgeview global loguploader newlogd nim nodeagent tpmmgr vaultmgr volumemgr watcher zedagent zedclient zedmanager zedrouter zfsmanager]

For more detail help on EdgeView commands, see https://wiki.lfedge.org/display/EVE/EdgeView+Commands

The URL above points to this document.

For help on a specific command, for example on 'flow' (assume it's a multiple instance, need to specify '-inst 1'):

edgeview.sh -inst 1 flow -h

flow[/<some pattern>] - display ip flow information in the kernel search pattern
e.g. flow/sport=53 -- display all the ip flow matches source port of 53
flow/10.1.0.2 -- display all the ip flow matches ip address of 10.1.0.2

The rest of this document will have more detailed information on each of the EdgeView commands.

Multi-Instances

When the EdgeView is started based on the configuration on the controller, it can be a single instance or multiple instance session. The multi-instance case is used when there is a need for multiple users to access the device or applications at the same time. For the multi-instance session, the user needs to supply the 'instance ID' when issuing EdgeView command, for example with the above 'edgeview.sh' script, an instance number is needed:

```
edgeview.sh -inst 2 route
```

The above command uses the 'instance 2' for EdgeView session to the device to query for the routes on the system.

In this document, the instance-ID will not be used in the command descriptions. Users just need to insert the '-inst <num>' if the EdgeView session is a multi-instance type.

Network Commands

The EdgeView networking related commands are the items printed from the '-h' output:

[acl app arp connectivity flow if mdns nslookup ping route socket speed tcp tcpdump trace url wireless]

Acl

```
acl[/<filter>] - to display all filters of running and configured ACL

  e.g. acl -- display all filters of ACL

       acl/nat -- display in table nat of ACL
```

It will display the access list for running ACL and also configured ACL of the EVE device. To see a subset on iptables, use the filter string. The filter string can be 'raw', 'filter', 'nat' or 'mangle'.

Addhost

```
addhost/host-name/host-IP - to add a host entry in EdgeView container's /etc/hosts file
```

The 'addhost' command adds one entry of specified hostname to IP address mapping into the EdgeView container's '/etc/hosts' for. This can be useful if there is no DNS entry for the hostname and the user knows the static mapping. An example:

```
edgeview.sh addhost/zedcontrol.local.zededa.net/32.165.49.119

...
127.0.0.1 localhost
::1 localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
32.165.49.119 zedcontrol.local.zededa.net
```

This '/etc/hosts' is in the EdgeView container, and not in EVE device's host and not in other EVE containers.

App

```
app[/app-string] - to display all the app or one specific app

  e.g. app -- display all apps in brief

       app/iot -- display a specific app, which app name has substring of iot in more detail
```

If the command is issued without the '/<app-string>', it will display all the user (or DomU) applications running on the device. For each application, it displays the App number, UUID, network bridge information, status in terms of boot time, CPU and memory, it also displays the VNC and App log settings.

if the '/<app-string>' is supplied in the command, it will display the specific user application with more detail information. The 'app-string>' is a subset of the App name string regardless of cases. For example, if the app name is 'cluster-k3s-2-agent1', one can use 'app/agent' to query. Besides the above information, it gets more networking information from the application. It will try to ping the App IP address to see if it is 'Up', it will get the DHCP info of the application, and iptable rules related to the application.

Connectivity

```
connectivity - display the port config list with index
```

This display can be used for troubleshooting network port issues when there is a configuration change. It displays the list of port configurations of the interfaces, and the current index in the list. When the EVE device can not connect to the controller, it will try to connect to the controller with a different port configuration on this list periodically.

Flow

```
flow[/<some pattern>] - display ip flow information in the kernel search pattern

  e.g. flow/sport=53 -- display all the ip flow matches source port of 53

       flow/10.1.0.2 -- display all the ip flow matches ip address of 10.1.0.2
```

The 'flow' command displays the content of the Linux Conntrack table. It contains the detailed network 5-tuple information for EVE device and user applications current network endpoints information. One can use filters to search for specific IP addresses or port numbers for display.

If

```
if[/intf-name] - display interface related information briefly

e.g. if/eth0 -- display interface eth0 related information
```

This command prints a brief device interface information. An interface name string can be entered as a filter. It also prints the proxy configuration if it exists.

Mdns

```
mdns[/intf-name][/service] - display zeroconf related information

  e.g. mdns/eth0 -- display mDNS for default service 'workstation' on interface 'eth0'

       mdns/bn1/https -- display mDNS for service 'https' on bridge 'bn1'

       mdns -- display mDNS for default service 'workstation' on all 'UP' interfaces
```

The mDNS is a multicast protocol for auto discovery of services. It will send queries over all the 'UP' interfaces on the device to discover the service being advertised. By default it sends out for 'workstation._tcp' service which is a well-known mDNS service supported.

The EVE 'local datastore' implementation for App is based on the mDNS to bind the domain name with '.local'. This 'mdns' command can be used to query on the bridge where the App is located and to see if it replies with the service for datastore image download.

Nslookup

```
nslookup[/<ip or name>] - display domain name and dns server information
  e.g. nslookup/www.amazon.com -- display DNS information on www.amazon.com
       nslookup/8.8.8.8 -- display DNS information on address 8.8.8.8
```

The 'nslookup' command is simply getting the DNS resolution result from the EVE device on site.

Ping

```
ping[/<ip or name>] - ping to 8.8.8.8 from all the UP interfaces or ping a specific address

  e.g. ping -- ping to 8.8.8.8 from each source IP address of the interfaces
```

```
ping/192.168.1.1 -- ping the address of 192.168.1.1
```

The 'ping' command if without any option, it will try to ping '8.8.8.8' and the controller of the EVE device from each of the interfaces. It can take a domain name or IP address option to send out the ICMP packets. It can be used on internal IP addresses such as App interface IP address, or on external servers to see if the device can reach it.

Route

```
route - display all the ip rule and their ip table entries
```

The 'route' command displays the IP Rule tables in the Linux kernel and their IP routes. It also walks through all the 'UP' interfaces and displays their associated routes.

Showcerts

```
showcerts[/<url>][/proxy-addr:proxy-port] - display TLS connection certificates of server side

    e.g. showcerts/zedcloud.local.zededa.net -- display TLS certificates from the controller

         showcerts/zedcloud.local.zededa.net/10.10.1.128:3128 -- display controller TLS certificates
         through a proxy server
```

The 'showcerts' command displays remote server or peer TLS certificates. It can optionally take the server side URL and also the proxy setting for proxy IP address and port. If no option is given, the showcerts finds the controller URL in '/config/server' file for the URL part. If the management port has proxy configuration, the 'showcerts' uses the proxy IP and port when querying the controller site.

The output of the certificates only gets a few human readable items, such as 'Version', 'Serial Number', 'Signature Algorithem', 'Validity' and 'Subject'.

The bellow example displays the 'showcerts' without option, the server url is '`zedcloud.local.zededa.net`', and the proxy is '`31.198.61.228:3128`'. The certificates from the peer are belong to the server or the controller, not to the proxy server. This may help the toubleshooting to determine if the proxy is a passthrough or a MiTM type.

```
edgeview.sh showcerts

=== Network: <peercerts> ===


url: zedcloud.local.zededa.net/31.198.61.228:3128
(0) Certificate:

    Data:

        Version: 3

    Serial Number:

        503025477018159975346019544684339737623192390922

    Signature Algorithem: SHA256-RSA

        Issuer:CN=Zededa Inc. Intermediat CA1,O=Zededa Inc.,ST=California,C=US
        Validity:

            Not Before: 2022-04-11 18:19:37 +0000 UTC
            Not After: 2023-04-21 18:19:37 +0000 UTC

        Subject: CN=zedcloud.local.zededa.net,O=Zededa Inc.,L=San Jose,ST=California,C=US

(1) Certificate:

    Data:

        Version: 3
        Serial Number:

            4098

    Signature Algorithem: SHA256-RSA

        Issuer:CN=Zededa Inc. Root CA,O=Zededa Inc.,L=San Jose,ST=California,C=US,1.2.840.113549.1.9.1
        =#0c0f63657274407a65646564612e6e6574
        Validity:

            Not Before: 2017-03-20 19:19:54 +0000 UTC
            Not After: 2027-03-18 19:19:54 +0000 UTC
```

```
Subject: CN=Zededa Inc. Intermediat CA1,O=Zededa Inc.,ST=California,C=US
```

Socket

```
socket - display all the ipv4 listening socket ports and established ports
```

The 'socket' command displays all the listening TCP and UDP sockets in the Linux kernel and the server information. It also displays the currently established sockets 5-tuple information.

Speed

```
speed[/intf-name] - run speed test and report the download and upload speed

  e.g. speed/wlan0 -- run speed test on interface wlan0
```

The 'speed' command uses the 'speedtest' utility to measure the download and upload speed of the device. The outbound interface name can be supplied to run the speed test over that port. An example of the output:

```
Hosted by XenSpec (Fremont, CA) [45.56 km]: 5.258 ms
Testing download speed.....................................................................
Download: 700.19 Mbit/s
Testing upload speed.....................................................................................
Upload: 773.80 Mbit/s
```

Tcp

See detail of the command in the section TCP Channel Commands.

Tcpdump

```
tcpdump/intf-name/[options] - tcpdump on the interface, can specify duration with -time, default is 60 sec

e.g. tcpdump/eth0/ -- run tcpdump on eth0 with default 60 seconds or maximum of 100 entries

tcpdump/eth0/'port 443' -time 10 -- run tcpdump on eth0 and port 443 with 10 seconds
```

The 'tcpdump' command is to capture the IP packets using the 'tcpdump' utility of Linux. The outbound interface needs to be specified.

The user can also supply more 'tcpdump' options such as port number or host IP address using e.g. 'port 53' or 'host 10.10.10.10' to capture the IP packets with those filters. The command will return the results either it times out or it has captured the maximum of 100 packets. The default timeout is 60 seconds. The user can specify the timeout in the range of (1, 120) seconds by '-time <value>'.

Trace

```
trace[/<ip or name>] - traceroute to www.google.com and zedcloud server, or to specified ip or name, 10 hops
limit
e.g. trace -- traceroute to www.google.com and to zedcloud server
trace/www.microsoft.com -- run traceroute to www.microsoft.com
```

The 'trace' command uses the 'traceroute' utility of Linux and returns the hop-by-hop result if available. It uses two-queries per hop (useful in ECMP) and it is limited to a maximum of 10 hops. The option can be an IP address or domain name.

Url

```
url - display url metrics for zedclient, zedagent, downloader and loguploader
```

The 'url' command is for the statistics of different services to the controller and data-stores. The services include 'zedagent', 'zedrouter', 'loguploader', etc. This stats is kept from the device reboot time. The command also displays the management interface traffic send and receive stats. An example of a subset of output:

```
- zedagent stats
interface: eth0
Success: 1853 Last Success: 2022-08-12 18:45:00.027419055 +0000 UTC
  https://zedcloud.local.zededa.net/api/v2/edgedevice/id/37df4d43-6d3e-4369-a455-9a189b1426bb/config
    Recv (KBytes): 7, Sent 307120, SentMsg: 880, TLS resume: 880, Total Time(sec): 108

 https://zedcloud.local.zededa.net/api/v2/edgedevice/id/37df4d43-6d3e-4369-a455-9a189b1426bb/info
   Recv (KBytes): 0, Sent 219684, SentMsg: 82, TLS resume: 82, Total Time(sec): 9
```

This example here is the part for EVE 'zedagent' service send/receive from different URLs for downloading device configure and upload device info messages. The stats contains the send and receive payload bytes, and number of messages (in 'SendMsg'). The 'TLS resume' indicates among all the TLS message exchanges, how many have the 'TLS resumption' (defined in RFC-5247). The stats has also the total round-trip time it spends on all those messages with that URL.

Wireless

```
wireless - display the iwconfig wlan0 info and wpa_supplicant.conf content
```

The 'wireless' command displays the 'wlan' and 'wwan' interfaces (if they exist on the EVE device) relation information, such as 'wpa_supplicant.conf' file content, the port configuration for the wireless interfaces and status.

System Commands

The EdgeView system related commands (similar to network commands, but less related to TCP/IP) are the items printed from the '-h' output:

```
[configitem cat cp datastore dmesg download du hw lastreboot ls model newlog pci ps cipher usb techsupport
top volume]
```

Configitem

```
configitem - display the device configitem settings, highlight the non-default values
```

The 'configitem' is user supplied configuration for the EVE device, for example to change the default logging on device from 'info' into 'debug'. This 'configitem' command will display all the available items and also highlight the changed values in color (if the terminal supports that). It has the global items and EVE agent specific items if changed. An example of changed 'debug.enable.usb' item in part of the output:

debug.default.loglevel: info

debug.enable.usb: false; default true

Cat

```
cat/<path to filename> - to display the content of a file

  e.g. cat//config/device.cert.pem -- display the /config/device.cert.pem file content

       cat/<path> -line <num> -- display only <num> of lines, like 'head' if <num> is positive, like 'tail'
if the <num> is negative
```

This is similar to the Linux 'cat' utility to display a file's content. the file starts from the full path from the EVE Linux root directory.

The option can take a '-line <num>', and this is similar to Linux 'head' and 'tail' utility to only display the number of lines from the file depending on whether the <num> is a positive or negative value. For example to tail the last 5 lines of the current device log:

eddgeview.sh cat/persist/newlog/collect/current.device.log -line -5

content type: text/plain; charset=utf-8

{"severity":"info","source":"edgeview","iid":"22761","content":"{\"level\":\"info\",\"msg\":\"recv[ep-inst-1:147.92.91.124] cmd: cat/persist/newlog/collect/current.device.log\",\"obj_name\":\"edgeview-cmd\",\"obj_type\":\"newlog-gen-event\",\"pid\":22761,\"source\":\"edgeview\",\"time\":\"2022-08-12T19:51:47.844877276Z\"}\n","msgid":2944,"timestamp":{"seconds":1660333907,"nanos":844877276}}

{"severity":"info","source":"edgeview","iid":"22761","content":"{\"level\":\"info\",\"msg\":\"Sent 3 messages, total 2245 bytes to websocket\",\"pid\":22761,\"source\":\"edgeview\",\"time\":\"2022-08-12T19:51:47.852530881Z\"}\n","msgid":2945,"timestamp":{"seconds":1660333907,"nanos":852530881}}

{"severity":"info","source":"edgeview","iid":"22761","content":"{\"level\":\"info\",\"msg\":\"read: no device, continue\",\"pid\":22761,\"source\":\"edgeview\",\"time\":\"2022-08-12T19:51:47.942746969Z\"}\n","msgid":2946,"timestamp":{"seconds":1660333907,"nanos":942746969}}

{"severity":"info","source":"edgeview","iid":"22761","content":"{\"level\":\"info\",\"msg\":\"recv[ep-inst-1:147.92.91.124] cmd: cat/persist/newlog/collect/current.device.log\",\"obj_name\":\"edgeview-cmd\",\"obj_type\":\"newlog-gen-event\",\"pid\":22761,\"source\":\"edgeview\",\"time\":\"2022-08-12T19:51:53.702389706Z\"}\n","msgid":2947,"timestamp":{"seconds":1660333913,"nanos":702389706}}

Cp

For the details of 'cp' command, see section Copy File Command.

Datastore

```
datastore - display the device current datastore: EQDN, type, cipher information
```

On EVE devices, it needs to be configured with datastore(s) for image downloading. The EVE image datastore is by default configured. The application image datastore is dynamically configured on the EVE device depending on the applications.

Dmesg

```
dmesg - display the device current dmesg information
```

The 'dmesg' command is to display the system log in kernel memory. The log severity Error and above is printed in Red, and Warn is printed in Pink, and rest of them in normal text color.

E.g.:

```
edgeview.sh dmesg

[ 1.063566] sd 0:0:1:0: [sda] Attached SCSI disk

[ 1.064425] sd 0:0:1:0: Attached scsi generic sg0 type 0

[ 1.065545] Rounding down aligned max_sectors from 4294967295 to 4294967288

[ 1.066723] db_root: cannot open: /etc/target

[ 1.067668] tun: Universal TUN/TAP device driver, 1.6

[ 1.159820] VMware vmxnet3 virtual NIC driver - version 1.5.0.0-k-NAPI

[ 1.174923] i8042: Warning: Keylock active
```

Download

```
download - display the download config and status during downloading operation and url stats since reboot
```

The 'download' command displays (only if the device is currently downloading image(s)) the configuration for download, and the status of downloading or progress. It also displays the download statistics since the last reboot.

Du

```
du - display linux 'du' in disk usage of a directory

  e.g. du//persist/vault -- get the total disk usage of files under that directory
```

For example, the above 'du//persist/vault' has the output:

- Disk Usage: /persist/vault

203.24 (MBytes)

Hw

```
hw - display the hardware from lshw information in json format
```

The 'hw' command uses the utility 'lshw' and it does not take options. It displays the device hardware information in JSON format.

Lastreboot

```
lastreboot - display the last reboot reasons and stack if the information is saved
```

The 'lastreboot' command will display the content of '/persist/log/reboot-reason.log' if it exist, and '/persist/newlog/panicStacks' if saved.

Ls

```
ls/<path to filenames> - to display the file/directory information

  e.g. ls//config/device.cert.pem -- display the /config/device.cert.pem file info

       ls//config/"device*" -- display all the files with prefix 'device' in /config
```

The 'ls' command displays the files information in the directory. It can take a wildcard in the file's name string to match a subset of files in the directory. For example:

*edgeview.sh ls//run/"zedagent*touch"*

*- ls cmd: /run/zedagent*touch*

-rw-r--r--, 2022-08-12 20:58:43.587358564 +0000 UTC, 0, zedagent-localappinfo.touch

-rw-r--r--, 2022-08-12 20:58:58.231320286 +0000 UTC, 0, zedagent-localdevinfo.touch

-rw-r--r--, 2022-08-12 20:58:58.228712666 +0000 UTC, 0, zedagent-location.touch

-rw-r--r--, 2022-08-12 20:58:53.320588547 +0000 UTC, 0, zedagent.touch

-rw-r--r--, 2022-08-12 20:58:58.228707899 +0000 UTC, 0, zedagentattest.touch

-rw-r--r--, 2022-08-12 20:58:58.232764804 +0000 UTC, 0, zedagentccerts.touch

-rw-r--r--, 2022-08-12 20:58:58.219222416 +0000 UTC, 0, zedagentconfig.touch

-rw-r--r--, 2022-08-12 20:58:58.204337241 +0000 UTC, 0, zedagentdevinfo.touch

-rw-r--r--, 2022-08-12 20:58:58.231334827 +0000 UTC, 0, zedagentecerts.touch

-rw-r--r--, 2022-08-12 20:58:58.205523822 +0000 UTC, 0, zedagentflowlog.touch

-rw-r--r--, 2022-08-12 20:58:58.205553158 +0000 UTC, 0, zedagenthwinfo.touch

-rw-r--r--, 2022-08-12 20:58:58.199945666 +0000 UTC, 0, zedagentmetrics.touch

-rw-r--r--, 2022-08-12 20:58:58.205532665 +0000 UTC, 0, zedagentobjectinfo.touch

Model

```
model - display the hardware model information in json format
```

Not yet supported

Newlog

```
newlog - display the newlog statistics and file information in each of the newlog directory and disk usage
```

The 'newlog' command displays the device logging statistics since the last reboot, and also it displays the logging zip file directory for 'devUpload', 'appUpload' and 'keepSentQueue' for the number of files in directory and time range of the files.

Pci

```
pci - display the lspci information on device
```

The 'pci' command runs the 'lspci' utility and displays all the PCI devices information.

PProf

```
pprof/on|off - turn on/off pprof http debugging in pillar on port 6543
```

More information about pprof can be found here: https://github.com/google/pprof

The port can be forwarded to the local machine with the *tcp* command.

Ps

```
ps/<string> - display the process status information on matching string

  e.g. ps/containerd -- display the processes with name of containerd
```

The 'ps' command displays the 'pid', system times, 'vms', 'rss', CPU, memory and 'cmdline' information. It takes a string as a filter for the 'cmdline' of the process. E.g.:

edgeview.sh ps/"edge-view-init"

- ps: PID Times VMS RSS CPU% MEM% Name Cmdline

001160: {"cpu":"cpu","user":0.0,"system":0.2,"idle":0.0,"nice":0.0,"iowait":0.0,"irq":0.0,"softirq":0.0,"steal":0.0,"guest":0.0,"guestNice":0.0,"stolen":0.0}, 1646592, 1191936, 0.048, 0.015, /bin/sh /usr/bin/edge-view-init.sh

Cipher

```
cipher - display cipher information on datastore, device and controller certificates, etc.
```

```
The 'cipher' command displays the certificates in '/persist/certs' directory, the datastore configured cipher
information, the TPM edge-node certs information, and controller certificate information.
```

For example, in the TPM cert info:

```
- TPMmgr Edgenode Certs:

40d54a918e2057350b38ba916a93f3a1:
```

```
  hash Algo: 1, Cert ID: QNVKkY4gVzULOLqRapPzoQ==, Cert Type: EdchXchange, Is TPM: true

    subject: CN=Device ECDH certificate,O=The Linux Foundation,L=San Francisco,ST=CA,C=US, serial:
394883739663285504205557011368746703 76, valid until: 2042-08-10 18:24:10 +0000 UTC

    issuer: CN=EVE,O=The Linux Foundation

9746991e739889b4bd4fd204ae12d372:

  hash Algo: 1, Cert ID: l0aZHnOYibS9T9IErhLTcg==, Cert Type: Ek, Is TPM: true

    subject: CN=Device Endorsement Key certificate,O=The Linux Foundation,L=San Francisco,ST=CA,C=US,
serial: 1154022839481741200814624789405443542 13, valid until: 2042-08-10 18:24:10 +0000 UTC

    issuer: CN=EVE,O=The Linux Foundation

b788be811856e0077cdaf5825763cddf:

  hash Algo: 1, Cert ID: t4i+gRhW4Ad82vWCV2PN3w==, Cert Type: signing, Is TPM: true

    subject: CN=Device Attestation certificate,O=The Linux Foundation,L=San Francisco,ST=CA,C=US, serial:
135279607474940962312277550145450798740, valid until: 2042-08-10 18:24:10 +0000 UTC

    issuer: CN=EVE,O=The Linux Foundation
```

Usb

```
usb - display the lsusb information on device
```

The 'usb' command uses the 'lsusb' utility to display the device USB information.

Tar

```
  tar/<path to directory>  -  to generate a tarfile of the directory

    e.g. tar//persist/agentdebug  -- download the tarfile persist.agentdebug.<time>.tar of that directory
```

The 'tar' command generates a tar file from the directory on a remote device with the given path. It will deposit the tar file in the mounted directory on the user's laptop for downloading files. This command allows the directory with data up to 512 MBytes. Certain directories may have user sensitive data and can not be tarred, e.g. '/persist/vault', '/persist/clear' and '/run/domainmgr/cloudinit'. The files with file name ends with '.key.pem' and '.key' will not be included in the tar file.

Techsupport

For the details of 'techsupport' command, see section Tech-Support Command.

Top

```
top - display linux 'top' in one batch

  e.g. top -line 20 -- display the first 20 lines of linux 'top' output
```

The 'top' command uses the Linux 'top' utility and displays the process information in an ordered way. The user can use the '-line <num>' option to only display the top number of lines from the output.

Volume

```
volume - display the app volume and content tree information for each app
```

The 'volume' command displays application volume information on the device. It has volume configuration and content tree information. For example, a container on the device:

```
- App fledge-app
  volume config, ID: 3a14546a-fdc5-491e-b992-5aec38181493
    name: fledge-app_0_m_0, ID 3a14546a-fdc5-491e-b992-5aec38181493, RefCount: 1

  content tree config: d371045d-3778-4fd7-ad64-3942ebc027fa
    url: brycedianomic/fledge_osdu, format: CONTAINER, sha:
    size: 0, name: dianomic-fledge
```

Log Search Commands

The 'log' command is used to search for log entries for the device and applications. Even though the logs are normally uploaded to the controller side, the users of the enterprise may not have the capability to search in the cloud. The application logs depending on the setting, it can be configured not to upload, then the only way to view them is through this 'log search' in EdgeView if needed.

```
log/<search string> [-time <start>-<end>] [-json] [-type <app|dev>] - display log with search-string,
default is now to 30 mins ago

  e.g. log/panic -time 0.2-2.5 -- display log contains 'panic', from 0.2 to 0.5 hours ago

          log/Clock -type app -- display previous 30 minutes log contains 'Clock' in app log

          log/certificate -time 2021-08-15T23:15:29Z-2021-08-15T22:45:00Z -json -- display log during the
          specified time in RFC3339 format which contains 'certificate' in json format

          log/copy-logfiles -time 2022-02-15T22:25:00Z-2022-02-15T22:40:00Z -- 'copy-logfiles' is reserved
          usage, to download all logfiles in the specified time duration, maximum time frame is 30 minutes
```

The 'log' command can take a number of options besides the <search string>. If none is specified, then the search duration is from now back to 30 minutes ago for all types of logs (device log and app log).

Log Search with specified time range

The 'log/<string> -time <range>' can be used to define the search of log in a particular time. The '-time' format can be either a floating point numbers range, or in RFC-3339 type.

The floating point number si represented in hour units. For example, 'log/container -time 0.1-0.3' will search all the logs from 0.1 hour to 0.3 hour ago for any string that matches 'container'. The RFC-3339 format can be used for any time range before, for example: 'log/container -time 2022-08-15T01:35:56Z-2022-08-15T02:00:00Z' for log search in that UTC time range. The starting and ending time is limited to a maximum of 5 hours.

Log Search with type

The type option can be 'dev', 'app' or 'all'. The default is 'all'.

Log display in JSON format

By default, the 'log/<search string>' output only displays a subset of key items, not the whole entry. To see the complete log entry, add the '-json' option.

Log file upload to user's laptop

Another way to study the log entries from the EVE device is to upload the log files onto the user's laptop. Since the user may not know what to search and wants to save the complete log entries during the time frame for more careful examination now or later. Because there is no search string needed, EdgeView reserves the string 'copy-logfiles' for this purpose. The command 'log/copy-logfiles' is used to upload log entries onto the user's laptop. The time range format is the same as log searches described above, except that the maximum range is 30 minutes.

The files will be uploaded onto the user's EdgeView container directory '/download', and when using the 'docker run' for EdgeView, it needs to mount the volume from your local directory onto the docker's '/download'.

For example, if the user wants to upload all the log entries from now back to 30 minutes ago (default time range):

```
edgeview.sh log/copy-logfiles

file: name logfiles-20220815221401.tar, size 71680
================================================

log files saved at /download/logfiles-20220815221401

file: app.9ce27b08-47e0-4568-a56e-b6dd0e4514cd.log.1660599913706.gz, size 1707
file: app.9ce27b08-47e0-4568-a56e-b6dd0e4514cd.log.1660600218779.gz, size 2003
file: app.9ce27b08-47e0-4568-a56e-b6dd0e4514cd.log.1660600528842.gz, size 1714
file: app.9ce27b08-47e0-4568-a56e-b6dd0e4514cd.log.1660600833916.gz, size 1707
file: app.9ce27b08-47e0-4568-a56e-b6dd0e4514cd.log.1660601143977.gz, size 2148
file: app.9ce27b08-47e0-4568-a56e-b6dd0e4514cd.log.1660601449037.gz, size 1702
file: dev.log.1660600023724.gz, size 8317
file: dev.log.1660600328787.gz, size 8334
file: dev.log.1660600633854.gz, size 7886
file: dev.log.1660600938925.gz, size 8290
file: dev.log.1660601243989.gz, size 8282
file: dev.log.1660601549052.gz, size 8554

uncompressed into /download/logfiles-20220815221401/app.9ce27b08-47e0-4568-a56e-b6dd0e4514cd.log.txt

uncompressed into /download/logfiles-20220815221401/dev.log.txt
```

In this example, since the user does not use time option, by default it is now to 30 minutes before. It creates a directory under the '/download' with the timestamp of current time in 'yyyymmddhhmmss' format. there are 6 device log gzip files, and 6 app files (belong to the same app). It is then decompressed to the gzip files and merged into text files with JSON format for each log entry inside. There will only be one device log file, there can be zero or more application log files with the app UUID in the filename. The text log files keep the strict time order of the log entries from the earliest to the latest.

Since the script mount the local file '/tmp/download' to the container '/download', the user can view those files locally:

```
% ls -lt /tmp/download/logfiles-20220815221401

total 1872

-rw-r--r-- 1 <username> wheel 790799 Aug 15 15:14 dev.log.txt

-rw-r--r-- 1 <username> wheel 160141 Aug 15 15:14 app.9ce27b08-47e0-4568-a56e-b6dd0e4514cd.log.txt
```

Pub/Sub Commands

The 'pub' commands are meant for users who have extensive EVE internal knowledge. EVE-OS has many services, e.g. 'zedagent', 'zedrouter', etc. Each service publishes some configuration or status for other services to consume or to subscribe. All those publications reflect the current state of EVE operation. When troubleshooting a problem, sometimes one needs to know what a service is publishing to see if it is correct. The majority of the publications are in the directory '/run/<service-name>/'. A service can publish one or more types of data, which are located in its subdirectories.

All the services supports 'pub' command:

```
[baseosmgr domainmgr downloader edgeview global loguploader newlogd nim nodeagent tpmmgr vaultmgr volumemgr
watcher zedagent zedclient zedmanager zedrouter zfsmanager]
```

the 'pub/<service-name>' will display all the data items of the service. For instance, for 'domainmgr' service, it has the data for 'AssignableAdapters', 'Capabilities', 'CipherMetrics', 'DomainMetric', 'HostMemory' and "processMetric'. The user can supply a string for the display of only that particular data-structure, e.g. only for 'DomainMetric', that display with 3 applications on the device:

```
edgeview.sh pub/domainmgr/domain

/run/domainmgr/DomainMetric

  service: 00000000-0000-0000-0000-000000000000.json

{

     "UUIDandVersion": {

          "UUID": "00000000-0000-0000-0000-000000000000",

          "Version": ""

     },

     "CPUTotalNs": 2453062500000,

     "CPUScaled": 4,

     "AllocatedMB": 0,

     "UsedMemory": 3715,

     "MaxUsedMemory": 0,

     "AvailableMemory": 4067,

     "UsedMemoryPercent": 47,

     "LastHeard": "2022-08-13T02:41:22.014224992Z",

     "Activated": true

}

service: 85e58364-0b95-414b-911e-08df57627a04.json

{

     "UUIDandVersion": {

          "UUID": "85e58364-0b95-414b-911e-08df57627a04",
```

```
          "Version": "2"
     },
     "CPUTotalNs": 1504271850003,
     "CPUScaled": 2,
     "AllocatedMB": 2648,
     "UsedMemory": 2648,
     "MaxUsedMemory": 2648,
     "AvailableMemory": 0,
     "UsedMemoryPercent": 100,
     "LastHeard": "2022-08-13T02:41:22.014224992Z",
     "Activated": true
}
service: 9ce27b08-47e0-4568-a56e-b6dd0e4514cd.json
{
     "UUIDandVersion": {
          "UUID": "9ce27b08-47e0-4568-a56e-b6dd0e4514cd",
          "Version": "2"
     },
     "CPUTotalNs": 4723668233462,
     "CPUScaled": 1,
     "AllocatedMB": 1624,
     "UsedMemory": 699,
     "MaxUsedMemory": 700,
     "AvailableMemory": 925,
     "UsedMemoryPercent": 43.0418701171875,
     "LastHeard": "2022-08-13T02:41:22.014224992Z",
     "Activated": true
}
service: d2009ffc-8fba-42e2-8870-a3ef0d0b7c1b.json
{
     "UUIDandVersion": {
          "UUID": "d2009ffc-8fba-42e2-8870-a3ef0d0b7c1b",
          "Version": "2"
     },
     "CPUTotalNs": 424687427279,
     "CPUScaled": 2,
     "AllocatedMB": 2648,
     "UsedMemory": 1152,
     "MaxUsedMemory": 1154,
     "AvailableMemory": 1496,
```

```
        "UsedMemoryPercent": 43.50453186035156,

        "LastHeard": "2022-08-13T02:41:22.014224992Z",

        "Activated": true

    }
```

The 'pub' command also support display more than one service by using the comma, for example, if the user wants to display both 'zedclient' and 'zedrouter' publications, below can be used:

```
edgeview.sh pub/zedclient,zedrouter
```

TCP Channel Commands

The 'tcp' command is the most useful one in EdgeView. It sets up a TCP relay channel from the user's laptop through the Dispatcher into the EVE device and further relaying the TCP traffic to applications on the device or even external hosts. It allows multiple TCP channels to multiple remote endpoints at the same time.

```
tcp/ip-address:port[/ip-address:port...][/proxy[@ip-addr]] - tcp connection to the ip addresses for
services, local mapping ports 9001 and above

    e.g. tcp/192.168.1.1:8080 -- points your browser to the locally listening port and http browsing
    192.168.1.1:8080

        tcp/10.1.0.2:80/10.1.0.2:8081 -- points your browser to the locally listening ports and http
        browsing remote 10.1.0.2 both 80 and 8081 ports

        tcp/proxy/localhost:5903 -- https proxy to locally listening ports and vnc viewer to #3 port on
        device

        tcp/proxy@10.1.2.3 -- https proxy and specify the address of DNS name server for URL lookup
```

TCP channel setup requires the port mapping for docker run. For EdgeView single instance, or the first instance, the port 9001 to 9005 is mapped. Each additional instance will add 5 to that range.

The 'tcp' command may be disabled for application or external hosts by policies.

The 'tcp' command is different from other EdgeView commands, the purpose of running the command is not to get back some query results or uploading files, but to setup a TCP (relay) service on the laptop with the mapped ports, and different client application(s) on the laptop will connect to the TCP service, but virtually to relay/connect to the remote TCP endpoints. Think of the TCP channel as a virtual TCP port mapping service with the user's laptop as the frontend, and the applications at remote as the backend. This virtual TCP port mapping service works across the Internet, firewall, proxy, etc.

Access/Log-in Application

If the user wants to log into the applications on the EVE device. There are multiple ways to do that using EdgeView. If the application runs SSHd inside, the user can use SSH:

- first use 'edgeview.sh app' to find out the IP address of the applications, for example two applications and their interface IPs are 10.1.0.130 and 192.168.1.100
- setup the TCP channel for ssh into both applications: edgeview.sh tcp/10.1.0.130:22/192.168.1.100:22

```
tcp mapping locally listening 2 ports to remote:
0.0.0.0:9001 -> 10.1.0.130:22

0.0.0.0:9002 -> 192.168.1.100:22
```

- assume both applications, has username of 'testing', open two more terminals on your laptop, launch the ssh session on first app with 'ssh testing@localhost -p 9001'; and in another terminal, launch another ssh session with 'ssh testing@localhost -p 9002'

In the above example, the ssh client session targets the laptop with port number 9001 and 9002, which will be virtually connecting to the remote applications 10.1.0.130:22 and 192.168.1.100:22. EdgeView handles all the TCP packets stitching and relaying.

If the application is a VM on the EVE device, instead of using SSH, the user can use VNC to connect to the application's console using EdgeView. Assume the two applications as above:

- first use 'edgeview.sh app' to find out the VNC display ID of the applications. For example, the VNC IDs are 4 and 5 for the applications.
- setup TCP channel into applications consoles by: edgeview.sh tcp/localhost:5904/localhost:5905

```
tcp mapping locally listening 2 ports to remote:
0.0.0.0:9001 -> localhost:5904

0.0.0.0:9002 -> localhost:5905
```

- open two VNC viewers on the laptop, enter the VNC endpoint for one: 'localhost:9001' and the other 'localhost:9002'.

The reason the 'tcp' command has the option to 'localhost:590x' is that the VNC service is maintained by the EVE Dom0 side 127.0.0.1 with port numbers 590x for application console access.

The above two mappings, on the left side of '' are the user laptop endpoints, and on the right side of '' are the EVE device side endpoints, or the endpoints reachable from the EVE device.

Access TCP Services of Application

The applications may have TCP services other than SSH, for example, it may have normal HTTP service. EdgeView TCP channel can be used to access those services by specifying the related ports. Here is an example of 'fledge' IoT application. It services 3 different TCP ports, 80, 8081 and 4840. Port 80 is for initial browser connection; after connecting, there is a page to setup browser to another port 8081 usually. The port 4840 is for accessing the 'TCP binary' data for IoT applications, it requires a special OPC UA client software to access. Here is an example to access the 'fledge' application on EVE device:

- first use 'edgeview.sh app' to find out the 'fledge' interface IP address, in this case it's 10.1.0.3

```
- app: fledge-app, appNum: 1

app uuid 9ce27b08-47e0-4568-a56e-b6dd0e4514cd

== bridge: bn1, nbu1x1, 10.1.0.3, 00:16:3e:00:01:01
```

- setup the TCP channel for the 'fledge' endpoints: edgeview.sh tcp/10.1.0.3:80/10.1.0.3:8081/10.1.0.3:4840

```
tcp mapping locally listening 3 ports to remote:
0.0.0.0:9001 -> 10.1.0.3:80

0.0.0.0:9002 -> 10.1.0.3:8081

0.0.0.0:9003 -> 10.1.0.3:4840
```

- open a web browser to 'http://localhost:9001', and set up the page for browser switching to endpoint 'localhost:9002' for HTTP service. To access the OPC data, on the MacOS, the user can download the 'prosys-opc-us-client' and set the remote endpoint to 'localhost:9003'.

Access TCP Services of External Hosts

To access the applications external to the EVE device (not part of the user applications), for instance a 'local profile' server which shares the LAN with the EVE device. It is similar to the above 'Access TCP Services of Application'. The only difference is the policies are controlled separately for applications and external endpoints. Obviously the user can not use 'edgeview.sh app' to find out external hosts and applications, those remote endpoints have to be learned through some other mechanism.

Access TCP Services of EVE device (Dom0 side)

This is also similar to the above application access, but with Dom0 side of the IP addresses and ports. For example, TCP access to the 'meta data' services for internal bridges:

- first use 'edgeview.sh socket' to find out which ports Dom0 side listening on

```
tcp LISTEN 0 1 0.0.0.0:5904 0.0.0.0:* users:(("qemu-system-x86",pid=4192,fd=24))

tcp LISTEN 0 1024 10.1.0.1:80 0.0.0.0:* users:(("zedbox",pid=1903,fd=378))

tcp LISTEN 0 1024 10.3.0.1:80 0.0.0.0:* users:(("zedbox",pid=1903,fd=347))

tcp LISTEN 0 1 0.0.0.0:5905 0.0.0.0:* users:(("qemu-system-x86",pid=4112,fd=24))
```

- setup TCP channel to both bridge's 'meta data' servers: edgeview.sh tcp/10.1.0.1:80/10.3.0.1:80

```
tcp mapping locally listening 2 ports to remote:
0.0.0.0:9001 -> 10.1.0.1:80

0.0.0.0:9002 -> 10.3.0.1:80
```

- open a web browser with 'http://localhost:9001/eve/v1/network.json' to get the external ip address on the device, it returns below in this example:

```
{"caller-ip":"10.1.0.1:57152","external-ipv4":"192.168.86.36","hostname":""}
```

Access HTTPs Service of Remote Endpoints

See the section Proxy Command.

Proxy Command

As mentioned above, the TCP channel sets up a virtual port mapping across the Internet with the frontend on the user's laptop and the backend being the remote endpoints from the EVE device. For many TCP services, that work just fine. But HTTPs is different, it has the certificates which define the domain name or service IP addresses. When a local web browser points to "https://localhost:9001", that application service will have issues with this 'localhost' or any IP address it does not have. The browser will also have the problem of verifying the certificates the server passes over. See FAQ on proxy for detail.

EdgeView TCP channel has the 'proxy' option for this usage. It requires the EdgeView TCP channel as a 'Virtual Proxy' service. This BTY is a 'pass-through' proxy, not a MiTM proxy. As in any proxy server, your application or host points to the laptop's proxy service port, and the proxy action is mainly on the EdgeView of the remote EVE device.

For example, the application with interface IP address of 10.1.0.2 is listening on TCP port 6443 as a HTTPs service for kubernetes API service.

- setup TCP proxy command: edgeview.sh tcp/proxy

```
tcp mapping locally listening 1 ports to remote:

0.0.0.0:9001 -> proxy
```

- assume the user has downloaded the 'kubeConfig' file on a local laptop, a 'kubernetes' management software can be used to point to the proxy server of 'localhost:9001'. The kubeConfig remote API server address in this case is the real remote IP address: 10.1.0.2, and that is inside the certificate the API server uses.

The reason this 'proxy' is part of the TCP command is that the 'proxy' service is only for one port 9001 here, the others can still be used for regular TCP channels for SSH service and others.

HTTPs with Domain Name

The above example is for HTTPs working with the IP address in the URL, but normally the URL contains the domain name instead of IP address. If the server's domain name can be acquired through normal DNS lookup EVE uses, then this is not a problem. In the case where the server's domain name is private, only being served by internal DNS record, then the user needs to know about the internal DNS server IP address from some out-band mechanism. EdgeView supports the proxy service with explicit specifying the DNS server IP address (e.g. it is 192.168.1.100) by:

edgeview.sh tcp/proxy@192.168.1.100

The usage is the same as a normal EdgeView proxy, with the browser pointing to an URL which has the server's domain name, and directing the proxy service to 'localhost:9001' in this case.

HTTPs with Static Hostname Mapping

If there is no local/private DNS server available, or the user does not know about it, but the user knows the domain name and the IP address of the server offering the HTTPs service, a static hostname mapping entry can be added to the '/etc/hosts' of the EdgeView container. (similar to docker's '--add-host' option) See the command 'Addhost' above.

Copy Files Command

```
cp/<path> - copy file from the device to locally mounted directory by specify the path

     e.g. cp//config/device.cert.pem -- copy the /config/device.cert.pem file to local directory

          cp//persist/newlog/keepSentQueue/dev.log.1630451424116.gz -- copy file with path to local directory
```

The 'cp' command allows the user to copy a file from EVE device onto the laptop. This is for the Dom0 side of the files, not application files inside the VM. For application VM files, users can use the above TCP command with SSH mapping, and use SCP to get the files. This 'cp' assumes the SCP or SSH is not available on the Dom0 side. The 'cp' copies the remote EVE device file onto the container's '/download' directory. When issued the 'docker run', the volume needs to be mapped locally.

The syntax is simple, only need to specify the path to the filename on the device, e.g.:

edgeview.sh cp//persist/certs/server-signing-cert.pem

file: name server-signing-cert.pem, size 1715
==========

file saved at /download/server-signing-cert.pem

Exam the uploaded file:

% cat /tmp/download/server-signing-cert.pem

-----BEGIN CERTIFICATE-----

MIIExTCCAq2gAwIBAgICEOYwDQYJKoZIhvcNAQELBQAwXjELMAkGA1UEBhMCVVMx

EzARBgNVBAgMCkNhbGlmb3JuaWExFDASBgNVBAoMC1plZGVkYSBJbmMuMSQwIgYD

...

Tech-Support Command

```
techsupport - show tech-support, run various edgeview commands with output downloaded in a compressed file
```

The 'techsupport' command is to gather most of the EdgeView other command output and save into a compressed file then uploading to the user's laptop. This is similar to some router vendor's "show tech-support" command on devices. The command includes the above mentioned EdgeView commands:

Network: (route, arp, if, acl, connectivity, url, socket, app, mdns, nslookup/google.com, trace/8.8.8.8, wireless, flow)

System: (hw, model, pci, usb, lastreboot, newlog, volume, app, datastore, cipher, dmesg, configitem)

Pubsub: (nim, domainmgr, nodeagent, baseosmgr, tpmmgr, global, vaultmgr, volumemgr, zedagent, zedmanager, zedrouter, zedclient, fsmanager, edgeview, watcher)

See detail of the description of each command in this document.

Similar to 'cp' and 'log/copy-logfiles', the 'techsupport' requires the docker volume of directory '/download' be mounted in the user's local system. This command takes a while to run (about 60 seconds). Here is an example of the output:

edgeview.sh techsupport

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

file: name techsupport-20220816202445.gz, size 82317

```
==================================================
```

file saved at /download/techsupport-20220816202445.gz

User can view the file (this file uncompressed to 569349 bytes in size):

```
% ls -l /tmp/download/techsupport-20220816202445.gz

-rw-r--r-- 1 <username> wheel 82317 Aug 16 13:25 /tmp/download/techsupport-20220816202445.gz
```

View the file content (it starts with the above network 'route' command):

```
zcat < /tmp/download/techsupport-20220816202445.gz | head -50

     - Show Tech-Support -


Device IPs: [192.168.86.36]; Endpoint IP 167.12.91.124:40839
UUID: 07ab2c40-408a-4bc2-b9f2-8ca94235074f
Controller: zedcloud.local.zededa.net
EVE-OS release 0.0.0-master-ca6084b9-2022-08-16.20.07-kvm-amd64, IMGA
Edge-View Ver: 0.8.2, JWT expires at 2022-08-23T01:35:56Z
2022-08-16T20:24:45Z(UTC), uptime 474 (sec) = 0 days


- network info -

=== Network: <route> ===


- ip rule:

1000: table 510, ip rule 1000: from all to all table 510

     routes in table: 510
     {Ifindex: 10 Dst: <nil> Src: <nil> Gw: <nil> Flags: [] Table: 510 Realm: 0}

     ...
```

Show TechSupport before Device Onboarding

When a device has problems with onboarding to the controller, it sometimes requires experienced engineers to troubleshoot the issues. The EdgeView access can not be used since the device has not onboarded yet and EdgeView session can not be enabled on the device.

It will help the troubleshooting if the 'techsupport' file of the current device status can be obtained, and the compressed 'techsupport' file can be copied onto a USB disk. This will help without the engineers to be physically on the console of the device.

Assume someone has the console access to the EVE device, go to the directory of '/run/edgeview', do a 'touch run-techsupport' there. Wait for about 60 seconds, there will be a compressed 'techsupport' file generated in the '/run/edgeview' directory. Here is an example:

```
07ab2c40-408a-4bc2-b9f2-8ca94235074f:~# cd /run/edgeview/
07ab2c40-408a-4bc2-b9f2-8ca94235074f:/run/edgeview# ls
07ab2c40-408a-4bc2-b9f2-8ca94235074f:/run/edgeview# touch run-techsupport
07ab2c40-408a-4bc2-b9f2-8ca94235074f:/run/edgeview# ls -lt
total 92
-rw-r--r-- 1 root root 92917 Aug 22 22:07 techsupport-20220822220657.gz
07ab2c40-408a-4bc2-b9f2-8ca94235074f:/run/edgeview#
```

Then copy the file '/run/edgeview/techsupport-20220822220657.gz' onto a USB disk.