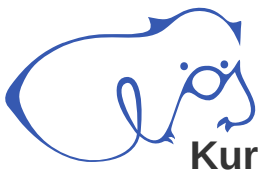


Froscon – 17-18.8.2024

## **Elos** An Event Management and Logging System

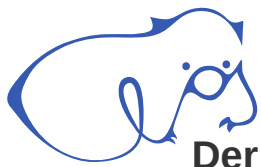
2024-07-11 | © emlix GmbH



## Kurze Vorstellung

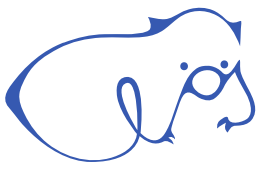
- Noch junges Projekt ca. 2022 gestartet
- Elos ist seit Juli 2023 als Open Source Projekt auf github <https://github.com/Elektrobit/elos>
- Ein gemeinsames Projekt von Elektrobit Automotive GmbH und emlix GmbH
- Laut github aktuell 7 Contributors
- Kurz zu meiner Person:
  - Wolfgang Gehrhardt
  - 2013 M.Sc. Informatik in Mittweida
  - seit 2013 als System Engineer bei emlix
  - Mail: [Wolfgang.gehrhardt@emlix.com](mailto:Wolfgang.gehrhardt@emlix.com)
  - Github: <https://github.com/gehwolf>



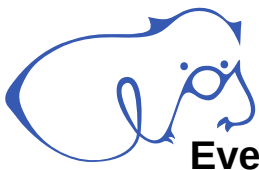


## Der Fahrplan

- Einführung / Vorstellung
- Events unter Linux : Wie Wo Was?
- Elos – Event Management und Logging System
- Elos – Was bringt es mit
- Elos – Ausblick auf die Roadmap
- Elos – Aktuelle Ökosystem

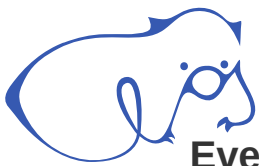


**Events unter Linux : Wie Wo Was?**



## Events unter Linux : Wie Wo Was?

- Quellen sind typischer weise :
  - Logfiles: Syslog, kmsg – /dev/kmsg, Application Logs – /var/log
  - Text basiert : /procfs && /sysfs
  - Binär-Formate: Netlink
- Die meisten Quellen haben kein maschinenlesbares Format
- Die verschieden Event-Quellen haben unterschiedliche Formate
- Die Aussage von Logfiles muss interpretiert werden
- Log-Ausgaben können sich unangekündigt zwischen Versionen von Applikationen ändern



- Coredump – Linux Kernel stellt Interface bereit

- /proc/sys/kernel/core\_pattern

```
ci@6fe651054fd1:/$ cat /proc/sys/kernel/core_pattern  
|/usr/lib/systemd/systemd-coredump %P %u %g %s %t %c %h
```

- Linux Kernel ruft Programm auf und liefert den Core-Dump via STDIN
  - Ist wieder ein anderes Format oder Syslog als Fallback

```
systemd-coredump[19198]: Process 19150 (sleep) of user 1000 terminated abnormally with signal 6/ABRT, processing...  
systemd[1]: Created slice Slice /system/systemd-coredump.  
systemd[1]: Started Process Core Dump (PID 19198/UID 0).  
systemd-coredump[19206]: [19198] Process 19150 (sleep) of user 1000 dumped core.
```

- OOM-Killer, Wann ist er gelaufen und welchen Prozess hat es getroffen?

- /proc/vmstat → „oom\_kill 0“

```
ci@91a7918ba3d8:/$ cat /proc/vmstat|grep oom  
oom_kill 0
```

- /dev/kmsg liefert

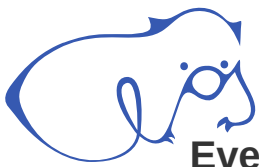
```
kernel: oom_reaper: reaped process 15860 (tail), now anon-rss:328kB, file-rss:476kB, shmem-rss:0kB
```



[https://elixir.bootlin.com/linux/v6.10.5/source/mm/oom\\_kill.c#L916](https://elixir.bootlin.com/linux/v6.10.5/source/mm/oom_kill.c#L916)

```
916 static void __oom_kill_process(struct task_struct *victim, const char *message)
917 {
918     struct task_struct *p;
919     struct mm_struct *mm;
920     bool can_oom_reap = true;
921
922     p = find_lock_task_mm(victim);
923     if (!p) {
924         pr_info("%s: OOM victim %d (%s) is already exiting. Skip killing the task\n",
925             message, task_pid_nr(victim), victim->comm);
926         put_task_struct(victim);
927         return;
928     } else if (victim != p) {
929         get_task_struct(p);
930         put_task_struct(victim);
931         victim = p;
932     }
933
934     /* Get a reference to safely compare mm after task_unlock(victim) */
935     mm = victim->mm;
936     mmgrab(mm);
937
938     /* Raise event before sending signal: task reaper must see this */
939     count_vm_event(OOM_KILL);
940     memcg_memory_event_mm(mm, MEMCG_OOM_KILL);
941
942     /*
943      * We should send SIGKILL before granting access to memory reserves
944      * in order to prevent the OOM victim from depleting the memory
945      * reserves from the user space under its control.
946      */
947     do_send_sig_info(SIGKILL, SEND_SIG_PRIV, victim, PIDTYPE_TGID);
948     mark_oom_victim(victim);
949     pr_err("%s: Killed process %d (%s) total-vm:%lukB, anon-rss:%lukB, file-rss:%lukB, shmem-rss:%lukB, UID:%u pgtables:%lukB oom_score_adj:%hd\r",
950         message, task_pid_nr(victim), victim->comm, K(mm->total_vm),
951         K(get_mm_counter(mm, MM_ANONPAGES)),
952         K(get_mm_counter(mm, MM_FILEPAGES)),
953         K(get_mm_counter(mm, MM_SHMEMPAGES)),
954         from_kuid(&init_user_ns, task_uid(victim)),
955         mm_pgtables_bytes(mm) >> 10, victim->signal->oom_score_adj);
```





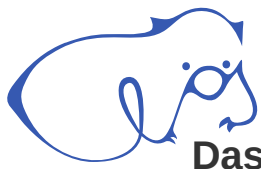
- Sensor Werte wie z.B Temperaturen aus dem sysfs:

```
ci@3d90a6eb9d3d:/$ cat /sys/class/thermal/thermal_zone0/temp  
20000
```

- SSH-, Logind-Logs

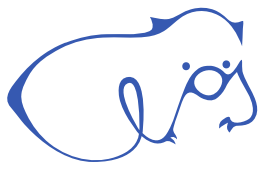
```
2 sshd-session[814224]: Accepted publickey for [REDACTED] from [REDACTED] port 33206 ssh2: RSA SHA256:[REDACTED]  
2 sshd-session[814224]: pam_unix(sshd:session): session opened for user [REDACTED](uid=1000) by [REDACTED](uid=0)  
2 sshd-session[814224]: pam_systemd(sshd:session): New sd-bus connection (system-bus-pam-systemd-814224) opened.  
2 systemd-logind[1188]: New session 38 of user [REDACTED].  
2 systemd[1]: Started Session 38 of User [REDACTED].  
2 sshd-session[814227]: Received disconnect from [REDACTED] port 33206:11: disconnected by user  
2 sshd-session[814227]: Disconnected from user [REDACTED] [REDACTED] port 33206  
2 systemd[1]: session-38.scope: Deactivated successfully.  
2 sshd-session[814224]: pam_unix(sshd:session): session closed for user [REDACTED]  
2 systemd[1]: session-38.scope: Consumed 1.290s CPU time, 120.8M memory peak.  
2 sshd-session[814224]: pam_systemd(sshd:session): New sd-bus connection (system-bus-pam-systemd-814224) opened.  
2 systemd-logind[1188]: Session 38 logged out. Waiting for processes to exit.  
2 systemd-logind[1188]: Removed session 38.
```



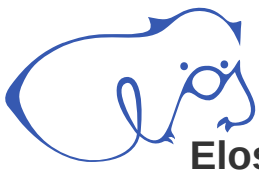


## Das Problem

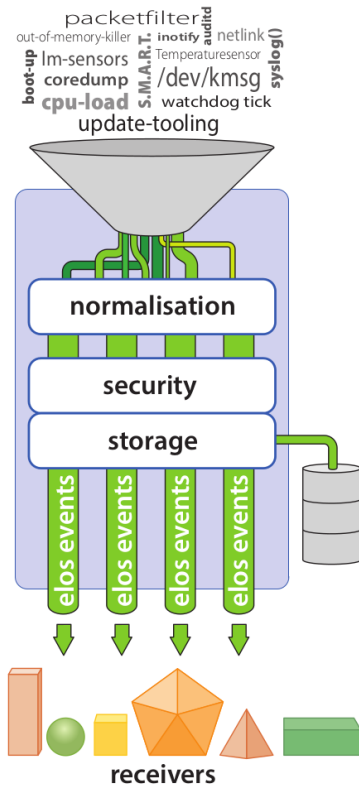
- Wo bekomme ich welche Information her?
  - Wo und wie können die Information effizient gespeichert werden?
  - Was steht in den Quellen
  - Was wird benötigt um sie auszulesen und zu interpretieren?
  - Wie halte ich den Änderungsaufwand in Grenzen?
  - Wie bekomme ich eine Überblick zum System-Status?
  - Wie kann ich auf Ereignisse reagieren?
  - Wie kann ich vergangene Ereignisse analysieren?
- **Gibt es da nicht eine Zentrale Instanz für?**



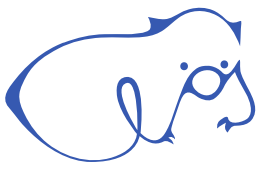
## **Elos – Event Management and Logging**



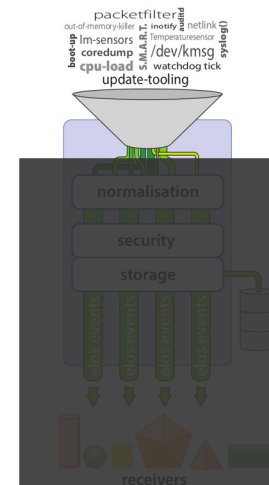
## Elos – Event Management and Logging

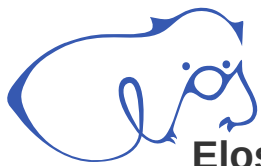


- Sammeln: Ein zentrales System zur Erhebung der Event Informationen
- Normalisieren: In das Elos Event Format übersetzen (maschinenlesbar)
- Security-Bewertung: Welche Events dürfen aus welcher Quelle kommen?
- Persistenz: Adaptives Speicher-Konzept
- Event Subscription: Reagieren in „Echtzeit“



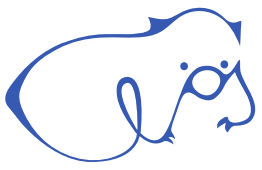
## Sammeln und Graben – Erheben von Events



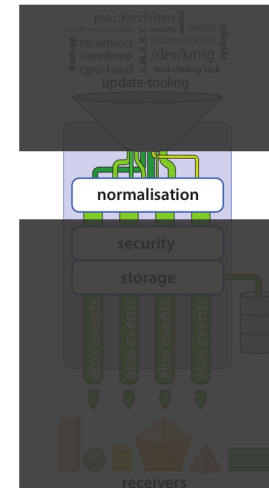


- Elos Scanner, sind Plugins die eine oder mehrerer Event-Quellen überwachen und elos Events erzeugen
- Ein Plugin kann mehre Instanzen haben.
  - Sie sollten sich dann unterscheiden in ihrer Konfiguration
  - Ermöglicht generische Scanner-Plugins, die je nach Konfiguration eine andere Quelle verwenden
- Scanner-Plugins können sich auch selbst auf Events subscriben. Das ermöglicht komplexere Events.
  - Beispielsweise die Überwachung des Systems auf Events in einer bestimmten Reihenfolge mit speziellen Schwellwerten um durch einen Event andere Prozesse zu informieren „Achtung, da passiert gleich was“ oder „System ist unter Last aber im grünen Bereich, macht weiter“

```
"Scanner": {
  "Plugins": {
    "ScannerDummy": {
      "File": "scanner_dummy.so",
      "Run": "always"
    },
    "OomKiller": {
      "File": "scanner_oomkiller.so",
      "Run": "always"
    },
    "SyslogScanner": {
      "File": "scanner_syslog.so",
      "Run": "always",
      "Config": {
        "SyslogPath": "/dev/log",
        "MappingRules": {
          "MessageCodes": {
            "4000": ".event.source.appName 'ssh' STRCMP",
            "2000": ".event.source.appName 'crinit' STRCMP",
            "1000": ".event.source.appName 'login' STRCMP"
          }
        }
      }
    },
    "KmsgScanner": {
      "File": "scanner_kmsg.so",
      "Run": "always",
      "Config": {
        "KmsgFile": "/dev/kmsg"
      }
    },
    "Shmem": {
      "File": "scanner_shmem.so",
      "Run": "always",
      "Config": {
        "ShmemFile": "scanner_shmem",
        "ShmemCreate": true,
        "ShmemLogEntries": 256,
        "ShmemOffset": 0,
        "SemFile": "scanner_shmem_sem",
        "SemCreate": true
      }
    }
  }
}
```



## Normalisieren – Das Elos Event Format



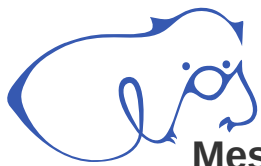


- Das einheitliche Elos-Event-Format besteht aus:
  - Date – Timestamp des Events
  - MessageCode – Ein einfacher numerischer Identifier für einen Event
  - Classification – 64 bit Feld zur Klassifizierung, Annotation von Events
  - Severity – Zur Gruppierung nach Schwere bzw. Dringlichkeit des Events
  - Hardwareid – Eindeutig System-Kennung
  - Source – Der Ursprung Auslöser des Prozesses
  - Payload – Zusätzliche Informationen für das Event
- Alle Attribute sind optional nur das `Date` wird von Elos gesetzt falls nicht vorhanden

```
{  
  "date": [1667929310,941892533],  
  "source": {  
    "appname": "openssh",  
    "filename": "/usr/bin/sshd",  
    "pid": 208,  
  },  
  "severity": 1,  
  "hardwareid": "817d6b97-75f8-4faf-ba3c-583ae1123558",  
  "classification": 6,  
  "messageCode": 8004,  
  "payload": "failed to login user xy"  
}
```



- Wallclock time stamp
- UTC wird angenommen
- Das Format ist [Sekunden,Nanos] (siehe man timepsec\_t)
- Wenn kein Timestamp gesetzt ist setzt Elos ihn zum Zeitpunkt des Dispatchens



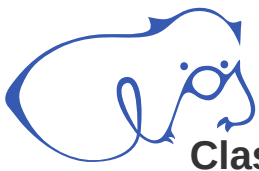
## Message Codes

- 0xxx – Elos Events
- 1xxx – Informational
- 2xxx – Program / Service Status
- 3xxx – Program / Service Resource Error
- 4xxx – Program / Programm IPC Error
- 5xxx – Program / Service Execution Error
- 6xxx – Hardware Fault
- 7xxx – Hardware Status
- 8xxx – Security Audit

Code	Description
2001	Process created
2002	Process Exited Successful
5002	Process Got SIGSEGV
5005	Process Core Dumped
6004	Critical Power
7126	Over Temperature
8004	Login Fail
8005	Login Ok
8007	Unauthorized Publishing

- Gibt Events zusätzliche Bedeutungen (Annotation von Events)
  - „Case open“ + Classification( 0x80 | 0x04 ) => Hardware + Security
- Definition <https://elektrobit.github.io/elos/src/components/event/index.html#classification>
- Ziel ist Wildwuchs und Doppeldeutigkeiten zu vermeiden
- Aktuell sind nur 11 von 32 Kategorien definiert

64-Bit Field Classification		
64-42	41-33	Bit 32-1
Reserved for future use	User Defined	Reserved & Defined by Elos



## Classification – Vordefinierte Kategorien

Value	Name	Description
0x00000000 000000000	und efined	used to indicate not available classification information
0x00000000 000000001	Kernel	all events which are related to some kernel functionality
0x00000000 000000002	Network	all network related information
0x00000000 000000004	Security	all events that are security related
0x00000000 000000008	Power	all events that is relevant for power management
0x00000000 000000010	Storage	all events which are related to filesystem and non volatile storage
0x00000000 000000020	Process	all events about the lifecycle of a process
0x00000000 000000040	IPC	all events about IPC
0x00000000 000000080	Hardware	all events about hardware
0x00000000 000000100	elos	all events about elos internals
0x00000000 000000200	Process Errors	all events about faulty or misbehaving processes

- 0x00 No logging.
- 0x01 Fatal error, not recoverable.
- 0x02 Error with impact to correct functionality.
- 0x03 Warning if correct behavior cannot be ensured.
- 0x04 Informational, providing high level understanding.
- 0x05 Detailed information for programmers.
- 0x06 Extra-verbose debug messages (highest grade of information)

```
typedef enum elosSeverityE {  
    /**/ // this comment is needed so  
    ELOS_SEVERITY_OFF = 0,  
    ELOS_SEVERITY_FATAL,  
    ELOS_SEVERITY_ERROR,  
    ELOS_SEVERITY_WARN,  
    ELOS_SEVERITY_INFO,  
    ELOS_SEVERITY_DEBUG,  
    ELOS_SEVERITY_VERBOSE,  
} elosSeverityE_t;
```

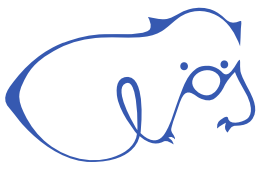
- Das Source Attribut und Sub-Attribute sind ebenfalls optional
- Es kann Prozesse oder Filesystem Ressourcen beschreiben
- Die Felder im einzelnen:
  - Filename – Dateisystem Pfad (Executable oder Resource wie /dev/log, /dev/kmsg, /proc,...)
  - PID – Der Process Identifier (Process ID)
  - Appname – Der Process Name. Er muss nicht einem Executable entsprechen und ist frei wählbar

```
"source": {  
  "appname": "openssh",  
  "filename": "/usr/bin/sshd",  
  "pid": 208,  
},
```

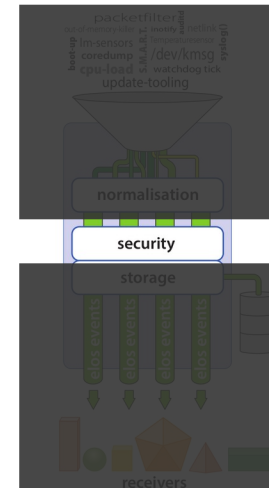
- Ist optional
- Payload kann ein beliebiges Format haben (Number, String, Json, ...)
- Der Message Code definiert was an Payload erwartet werden kann, nicht muss
- Aktuell im Json basiertem Protokoll wird nur String unterstützt

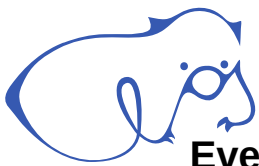


- Elos Event-Filter basieren auf Reverse Polish Notation (häufig auch RPN-Filter genannt)
- Die Event-filter Syntax
  - rule: Term|Operand Operand Operator
  - Term: <Operand> <Operand> <Operator>
  - Operand: Process | Event | Value
  - Event: .event.<field> | .e.<field> | .ev.<field>
  - Value: number|string
  - Operator: EQ|LT|LE|GT|GE|STRCMP|AND|OR|REGEX
  - <field>: Event-Attribut
- Beispiele:
  - True bei Core Dump events: .event.messageCode 5005 EQ
  - True bei Unauthorized events: .event.messageCode 8007 EQ
  - True bei Security events mit Severity höher als `warning`: .e.classification 4 EQ .e.severity 3 LE AND



## Security – Event-Authorisation und Blacklisting

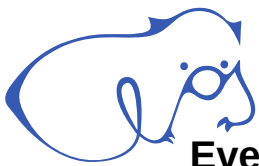




## Event-Blacklisting

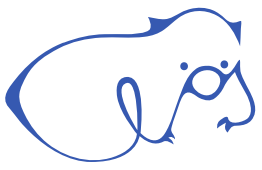
- Blacklisted Events dürfen nur von autorisierten Prozessen gepublished werden
- Versuche werden mit einem Security Event protokolliert
- Die Blacklist ist ein Event-Filter (RPN-Filter s.h. Elos-Events)

```
"EventBlacklist": ".event.messageCode 2000 EQ",
```

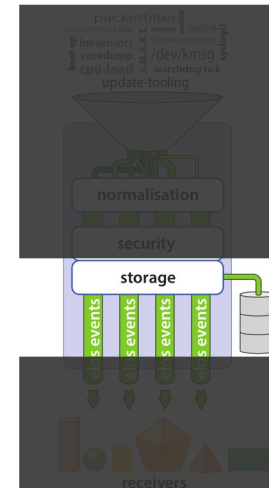


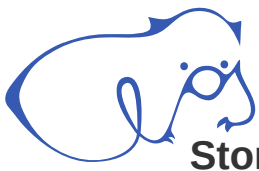
- Autorisierte Prozesse dürfen blacklistet Events publishen
- Zur Definition gibt es ProcessFilter, hier am Beispiel für ein TCP-Client-Plugin Instanz

```
"authorizedProcesses": [  
  ".process.uid 0 EQ .process.gid 0 EQ .process.exec '/bin/elosc' STRCMP AND",  
  ".process.gid 200 EQ .process.exec '/bin/elosc' STRCMP AND",  
  ".process.pid 1 EQ"  
]
```



## Storage – Adaptive Speicher System

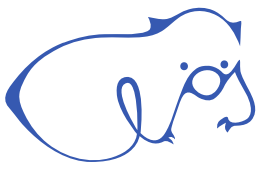




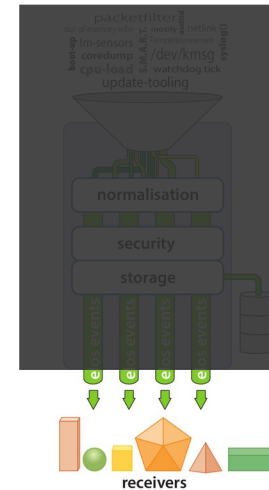
## StorageBackends

- Nutzt das Elos Plugin System „StorageBackend“-Plugins
- Ein Plugin kann mehrere Instanzen haben.
  - Eine JSON-Logfile für Coredump-Events, eines für Security-Events ...
- Eine StorageBackend-Instanz definiert über Event Filter welche Events sie verwaltet

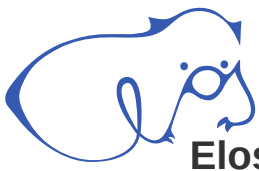
```
"EventLogging": {
  "Plugins": {
    "Dummy": {
      "File": "backend_dummy.so",
      "Run": "always",
      "Filter": [
        ".event.messageCode 1000 NE"
      ]
    },
    "DLT": {
      "File": "backend_dlt_logger.so",
      "Run": "always",
      "Filter": [
        "1 1 EQ"
      ],
      "Config": {
        "Connection": "/tmp/dlt",
        "EcuId": "ECU1",
        "AppId": "ELOS"
      }
    }
  },
  "SQLBackend": {
    "File": "backend_sql.so",
    "Run": "always",
    "Filter": [
      "1 1 EQ"
    ],
    "Config": {
      "ConnectionPath": "/tmp/elos.sqlite"
    }
  },
  "SkippedDummy": {
    "File": "backend_dummy.so",
    "Run": "never"
  },
  "JsonBackend": {
    "File": "backend_json.so",
    "Run": "always",
    "Filter": [
      "1 1 EQ"
    ],
    "Config": {
      "StoragePath": "/tmp/elosd_%host%_%date%_%count%.log",
      "Flags": [
        "O_SYNC"
      ]
    }
  }
}
```



## Receiver/Subscriber – Auf Events reagieren





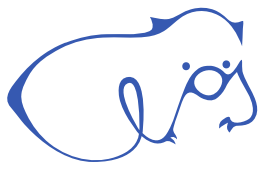


- Client-Plugins ermöglichen unterschiedliche Interfaces für Elos-Clients
- TCP-Client (libelos, python Elos Beispiel)
- Weitere sind geplant für Unix-Domain Socket und/oder Shared-Memory
- Ein Client hat prinzipiell folgende Möglichkeiten:
  - Publishen von Events
  - Subscriben auf Events
  - Fetchen von kürzlich vergangenen Events

u8	Version
u8	Command
u16	Message- Length
...	Payload

```
ELOS_MESSAGE_GET_VERSION      0x01
ELOS_MESSAGE_EVENT_PUBLISH    0x02
ELOS_MESSAGE_EVENT_SUBSCRIBE  0x03
ELOS_MESSAGE_LOG_FIND_EVENT   0x04
ELOS_MESSAGE_EVENTQUEUE_READ  0x05
ELOS_MESSAGE_EVENT_UNSUBSCRIBE 0x06

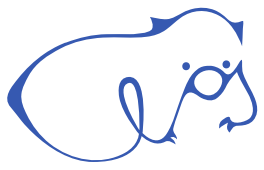
ELOS_MESSAGE_RESPONSE_BIT      0x80
ELOS_MESSAGE_RESPONSE_INVALID  ELOS_MESSAGE_RESPONSE_BIT
ELOS_MESSAGE_RESPONSE_GET_VERSION (ELOS_MESSAGE_GET_VERSION | ELOS_MESSAGE_RESPONSE_BIT)
ELOS_MESSAGE_RESPONSE_EVENT_PUBLISH (ELOS_MESSAGE_EVENT_PUBLISH | ELOS_MESSAGE_RESPONSE_BIT)
ELOS_MESSAGE_RESPONSE_EVENT_SUBSCRIBE (ELOS_MESSAGE_EVENT_SUBSCRIBE | ELOS_MESSAGE_RESPONSE_BIT)
ELOS_MESSAGE_RESPONSE_EVENT_UNSUBSCRIBE (ELOS_MESSAGE_EVENT_UNSUBSCRIBE | ELOS_MESSAGE_RESPONSE_BIT)
ELOS_MESSAGE_RESPONSE_LOG_FIND_EVENT (ELOS_MESSAGE_LOG_FIND_EVENT | ELOS_MESSAGE_RESPONSE_BIT)
ELOS_MESSAGE_RESPONSE_EVENTQUEUE_READ (ELOS_MESSAGE_EVENTQUEUE_READ | ELOS_MESSAGE_RESPONSE_BIT)
```



**Elos – Was bringt es mit**

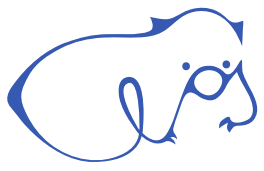
- Scanner:
  - Syslog-Scanner
  - Kmsg-Scanner
  - Shared-Memory-Scanner
  - OOM-Killer Scanner
- StorageBackends
  - SQLite-Backend
  - JSON-File Backend
  - DLT-Backend (Diagnostic Log and Trace)
  - MongoDB
  - Ringbuffer In-Memory-Backend
- Clients:
  - TCP

- Elosc – A Command Line Client
  - Publish Events
  - Subscribe to Events
  - Fetch Elos Kurzeit-Event-storage
- Elos-corcedump
  - Coredump Tool zum Verwalten von Coredumps
  - Speichert Coredumps
  - Erzeugt Coredump Events
- Elos Demos
  - ElosMon – Versenden von Emails zu ausgesuchten Events
  - elos\_log4c – Beispiel zur Integration von Elos in log4-Frameworks
  - Python Client Demo – einfaches Python-Beispiel
  - Und mehr, s.h. <https://elektrobit.github.io/elos/src/demos/index.html>



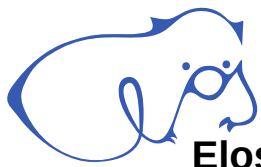
## Elos – Roadmap

- SMART (storage health) – a scanner plugin to report SMART status changes
- temperature – scanner plugin to report high and normal temperature
- panic() / kexec – a plugin to report a system crash after recovering
- in memory storage – i.e. record elos internal events
- mqtt backend – deliver events to MQTT broker
- Unix-Domain socket support – alternative client connection to TCP
- shared memory – with elos protocol
- notify about high CPU load
- notify about high disk load / disk full
- notify about network package drops
- graceful integration with other logging systems (syslog-ng, journald, kerneld)
- process monitoring – life cycle / highest load ( memory, cpu, network, storage usage)
- improve plugin development support (API, SDK, Repos) – multi language bindings (rust , python, ...)

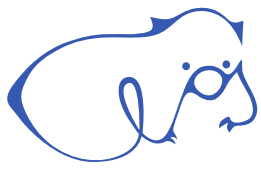


## Elos – Ökosystem





- Elos Projekt selbst auf github: <https://github.com/Elektrobit/elos>
- Elos Dokumentation: <https://elektrobit.github.io/elos>
- Meta-elos: <https://github.com/Elektrobit/meta-elos>
  - Kann benutzt werden um elos in eigene Yocto Projekte zu integrieren
  - Yocto Test Image zum spielen mit elos
- Elos-plugin: <https://github.com/Elektrobit/elos-plugins>
  - Eine Sammlung von Plugins
  - Beispiel-Plugins für Scanner/StorageBackends/Clients (Dummy-Plugins)
  - Plugins mit Abhängigkeiten, die wir nicht im elos-core haben wollen
  - Prinzipiell kann ein Plugin ein eigenes github Repo haben
- Elos-PPA : <https://launchpad.net/~elos-team/+archive/ubuntu/ppa>
- Elos-AUR Archlinux: <https://aur.archlinux.org/packages/elos>
- Elos on Docker-Hub: <https://hub.docker.com/gehwolf/elos>



## How can we support you?

emlix GmbH  
Göttingen | Berlin | Bonn

Headquarters  
Berliner Str. 12  
D-37073 Göttingen / Germany

Fon +49 (0) 551 / 306 64 - 0  
[solutions@emlix.com](mailto:solutions@emlix.com)  
[www.emlix.com](http://www.emlix.com)



- Unit Tests
- Smoketest
- Integrtaion Tests
- Benchmarks



packetfilter  
out-of-memory-killer  
up lm-sensors  
audit  
netlink  
notify  
Temperature sensor  
slog()

```
[{"date": [3533, 131501586], "source": {}, "classification": 256, "payload": "12 Events published\n"}]  
new data [1723808217, 595492452]:  
[{"date": [1723808215, 889662000], "source": {"fileName": "\dev\kmsg"}, "severity": 4, "classification": 1, "messageCode": 1111, "payload": "6,1258,3533629662,-;usb 3-9: new full-speed USB  
device number 7 using xhci_hcd\n SUBSYSTEM=usb\n DEVICE=+usb:3-9"}, {"date": [1723808215, 1032438000], "source": {"fileName": "\dev\kmsg"}, "severity": 4, "classification": 1, "messageCode": 1111, "payload": "6,1259,3533772438,-;usb 3-9: New USB device found, idVendor=1050, idProduct=0407, bcdDevice= 5.43\n SUBSYSTEM=usb\n DEVICE=c189:262"}, {"date": [3534, 253474685], "source": {}, "classification": 256, "payload": "14 Events published\n"}, {"date": [1723808215, 1032469000], "source": {"fileName": "\dev\kmsg"}, "severity": 4, "classification": 1, "messageCode": 1111, "payload": "6,1260,3533772469,-;usb 3-9: New USB device strings: Mfr=1, Product=2, SerialNumber=0\n SUBSYSTEM=usb\n DEVICE=c189:262"}, {"date": [1723808215, 1032480000], "source": {"fileName": "\dev\kmsg"}, "severity": 4, "classification": 1, "messageCode": 1111, "payload": "6,1261,3533772480,-;usb 3-9: Product: YubiKey OTP+FIDO+CCID\n SUBSYSTEM=usb\n DEVICE=c189:262"}, {"date": [1723808215, 1032489000], "source": {"fileName": "\dev\kmsg"}, "severity": 4, "classification": 1, "messageCode": 1111, "payload": "6,1262,3533772489,-;usb 3-9: Manufacturer: Yubico\n SUBSYSTEM=usb\n DEVICE=c189:262"}, {"date": [1723808215, 1038357000], "source": {"fileName": "\dev\kmsg"}, "severity": 4, "classification": 1, "messageCode": 1111, "payload": "6,1263,3533778357,-;input: Yubico YubiKey OTP+FIDO+CCID as \devices\pci0000:00\0000:00:14.0\usb3\3-9\3-9:1.0\0003:1050:0407.0005\input\input46"}, {"date": [1723808215, 1094177000], "source": {"fileName": "\dev\kmsg"}, "severity": 4, "classification": 1, "messageCode": 1111, "payload": "6,1264,3533834177,-;hid-generic 0003:1050:0407.0005: input,hidraw0: USB HID v1.10 Keyboard [Yubico YubiKey OTP+FIDO+CCID] on usb-0000:00:14.0-9\input0\n SUBSYSTEM=hid\n DEVICE=+hid:0003:1050:0407.0005"}, {"date": [1723808215, 1095998000], "source": {"fileName": "\dev\kmsg"}, "severity": 4, "classification": 1, "messageCode": 1111, "payload": "6,1265,3533835998,-;hid-generic 0003:1050:0407.0006: hiddev96,hidraw1: USB HID v1.10 Device [Yubico YubiKey OTP+FIDO+CCID] on usb-0000:00:14.0-9\input1\n SUBSYSTEM=hid\n DEVICE=+hid:0003:1050:0407.0006"}, {"date": [1723808215, 1240207000], "source": {"fileName": "\dev\kmsg"}, "severity": 4, "classification": 1, "messageCode": 1111, "payload": "6,1266,3533980207,-;input: Yubico YubiKey OTP+FIDO+CCID as \devices\virtual\input\input47"}]  
new data [1723808218, 596523742]:
```

security

storage

os events

os events

os events

os events

