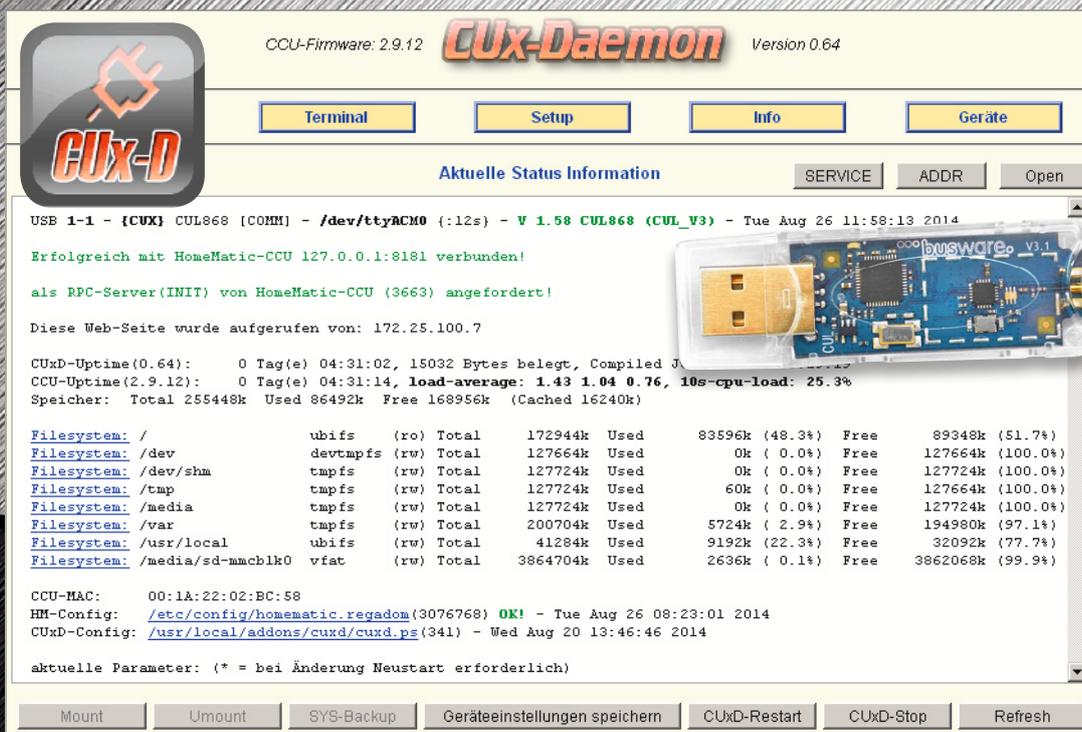




CUxD – das Leatherman für die HomeMatic®-CCU

Teil 3



CCU-Firmware: 2.9.12 **CUxDaemon** Version 0.64

Terminal Setup Info Geräte

Aktuelle Status Information SERVICE ADDR Open

USB 1-1 - {CUxD} CUL868 [COMM] - /dev/ttyACM0 (::12s) - V 1.58 CUL868 (CUL_V3) - Tue Aug 26 11:58:13 2014

Erfolgreich mit HomeMatic-CCU 127.0.0.1:8181 verbunden!

als RPC-Server(INIT) von HomeMatic-CCU (3663) angefordert!

Diese Web-Seite wurde aufgerufen von: 172.25.100.7

CUxD-Uptime(0.64): 0 Tag(e) 04:31:02, 15032 Bytes belegt, Compiled J...
CCU-Uptime(2.9.12): 0 Tag(e) 04:31:14, **load-average: 1.43 1.04 0.76, 10s-cpu-load: 25.3%**
Speicher: Total 255448k Used 86492k Free 168956k (Cached 16240k)

Filesystem:	/	ubifs	(ro)	Total	172944k	Used	83596k (48.3%)	Free	89348k (51.7%)
Filesystem:	/dev	devtmpfs	(rw)	Total	127664k	Used	0k (0.0%)	Free	127664k (100.0%)
Filesystem:	/dev/shm	tmpfs	(rw)	Total	127724k	Used	0k (0.0%)	Free	127724k (100.0%)
Filesystem:	/tmp	tmpfs	(rw)	Total	127724k	Used	60k (0.0%)	Free	127664k (100.0%)
Filesystem:	/media	tmpfs	(rw)	Total	127724k	Used	0k (0.0%)	Free	127724k (100.0%)
Filesystem:	/var	tmpfs	(rw)	Total	200704k	Used	5724k (2.9%)	Free	194980k (97.1%)
Filesystem:	/usr/local	ubifs	(rw)	Total	41284k	Used	9192k (22.3%)	Free	32092k (77.7%)
Filesystem:	/media/sd-mmcbk0	vfat	(rw)	Total	3864704k	Used	2636k (0.1%)	Free	3862068k (99.9%)

CCU-MAC: 00:1A:22:02:BC:58
HM-Config: [/etc/config/homematic_regadom\(3076768\)](#) OK! - Tue Aug 26 08:23:01 2014
CUxD-Config: [/usr/local/addons/cuxd/cuxd.ps\(341\)](#) - Wed Aug 20 13:46:46 2014

aktuelle Parameter: (* = bei Änderung Neustart erforderlich)

Mount Umount SYS-Backup Geräteeinstellungen speichern CUxD-Restart CUxD-Stop Refresh

Im dritten Teil unserer CUxD-Artikel-Reihe zeigen wir, welche hilfreichen Softwarefunktionen auch ohne zusätzliche Empfänger-Hardware durch das CUxD-Add-on zur Verfügung gestellt werden. Da viele dieser Funktionen sehr umfangreich und komplex sind, ist es empfehlenswert, die detaillierte Beschreibung der einzelnen Funktionen nochmals im CUxD-Handbuch [1] nachzulesen.



Die CUxD-Verwaltungsfunktionen

Die allgemeinen Notfall-Verwaltungsfunktionen sind über die URL „http://CCU-IP-Adresse/addons/cuxd/maintenance.html“ erreichbar (Bild 1). Hier können zum Beispiel alle aktuell auf der CCU gestarteten Prozesse angezeigt werden. Ein großer Vorteil ist, dass ein Abruf dieser Seite wesentlich schneller als ein Aufruf der WebUI erfolgt und auch nach einem Absturz des ReGaHss-Services, der WebUI oder vom CUxD noch auf diese Webseite mit allen angebotenen Funktionen per Webbrowser zugegriffen werden kann. Auch ein CCU-Neustart (Restart) aus der Ferne ist so bei Problemen einfach möglich. Über den letzten Punkt „Shell command“ können beliebige CCU-Shell-Befehle gestartet werden, die Ausgabe erfolgt in einem separaten Browser-Fenster.



Hinweis:

Sofern der Java-HM-Server deaktiviert wird, sind die Gruppen-, Diagramm- und allgemeinen Einstellungs-Funktionen der CCU2 nicht mehr verfügbar!

Weiterhin besteht die Möglichkeit, auf das gesamte Filesystem der CCU per Webbrowser lesend zuzugreifen und einzelne Dateien herunterzuladen (Bild 2). So können z. B. vor einem Neustart noch schnell die entsprechenden Logfiles (z. B. aus den Verzeichnissen /tmp/ oder /var/log/) für eine spätere Analyse gesichert werden. Der Filebrowser bietet weiterhin die Möglichkeit, Bilder als Thumbnails darzustellen und Zeichenkodierungen auf UTF-8 anzupassen. Die Sortierung der Verzeichnisansicht kann über eine Auswahlliste bestimmt werden.

Die CUxD-Statusseite

Ist der CUxD gestartet, dann können auf der CUxD-Statusseite „http://CCU-IP-Adresse/addons/cuxd/“ alle wichtigen Systeminformationen (Bild 3) abgelesen werden. Dazu gehören die angeschlossenen und verbundenen USB-Interfaces, die CCU-Uptime und aktuelle Prozessorauslastung, die Hauptspeicherauslastung und die Auslastung aller gemounteten Volumes (Bild 3 oben).

Auch alle gemounteten Filesysteme sind auf der Statusseite einsehbar, und durch einen Klick auf den Link „Filesystem:“ (Bild 3 Mitte) wird automatisch der integrierte Filebrowser gestartet (siehe Bild 2).

In der HM-Config-Zeile (Bild 3 unten) wird hinter dem „homematic.regadam“-File angezeigt, ob diese Systemdatei beim letzten Mal vollständig (OK!) oder unvollständig (ERROR!) gespeichert wurde. Bei unvollständiger Speicherung gibt es beim nächsten CCU-Neustart sehr wahrscheinlich Probleme. Hier sollte die Sicherung vor dem Neustart wiederholt werden. Das kann ganz einfach durch einen Aufruf von „DOM-Save“ auf der CUxD-Notfallseite (siehe Bild 1) erfolgen.

CUxD-System-Geräte

Die CUxD-System-Geräte „(28)“ stellen eine Reihe von nützlichen Zu-

satzfunktionen über virtuelle Geräte auf der CCU zur Verfügung (Bild 4). Diese Funktionen werden über Datenpunkte von virtuellen CCU-Geräten aktiviert oder abgefragt. Dabei handelt es sich um einen 16-Kanal-Timer, ein 16-Kanal-System.Exec mit variablen WebUI-Controls, ein speziell für RGB(W)-Dimmer gedachten 16-Kanal Multi-DIM-Exec und ein 16-Kanal-Netzwerk-Ping.

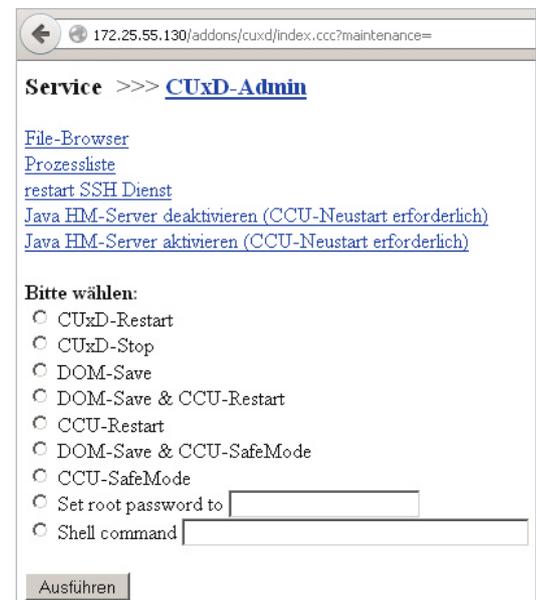


Bild 1: CUxD-Service-Funktionen

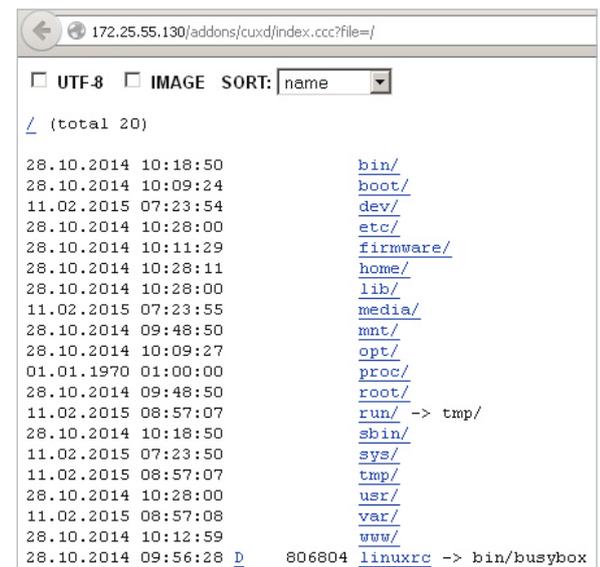


Bild 2: Der Filebrowser



The screenshot shows the CUX-Daemon web interface. At the top, it displays 'CCU-Firmware: 2.9.12' and 'CUX-Daemon Version 0.69'. Below this are navigation tabs: 'Status', 'terminal', 'Setup', 'Info', and 'Geräte'. The 'Status' tab is active, showing 'Aktuelle Status Information'. The main content area displays system information, including CPU usage, memory usage, and a list of filesystems. A red box highlights the 'CUXD-Uptime' and 'CCU-Uptime' information, showing that the system has been running for 11 days and 19 hours. Another red box highlights the 'Filesystem:' list, which shows the usage of various filesystems such as /, /dev, /dev/shm, /tmp, /media, /var, /usr/local, and /media/sd-mmcblk0. A third red box highlights the 'CCU-MAC' and 'Device-Log' information, showing the MAC address and the location of the device log file.

Bild 3: Die CUXD-Startseite mit allen wichtigen Systeminformationen im Überblick

In Verbindung mit eigenen bzw. den bereits mit CUXD ausgelieferten Scripts können so z. B. ganz einfach CCU-Backups automatisiert, eine Anwesenheit simuliert oder die Erreichbarkeit von Netzwerkgeräten überprüft werden. Die CUXD-Scripts sind in der Anleitung [1] unter Kapitel 6 „Zusatzprogramme“ beschrieben. Eine detaillierte Erläuterung der einzelnen Funktionen der System-Devices kann in der CUXD-Anleitung [1] unter Kapitel 5.8 (28 System-Devices) eingesehen werden.

Timer

Dieses Gerät ermöglicht das programmgesteuerte Auslösen von variablen und zufälligen Ereignissen. Es gibt verschiedene Möglichkeiten, den Timer zu setzen. Entweder in Sekunden relativ zur aktuellen Uhrzeit oder absolut zur Stunde bzw. zum Tag. Zusätzlich kann jedem Wert noch eine zufällige Zeit hinzugefügt werden. Auch ein automatischer Timer-Neustart nach Ablauf des Intervalls ist möglich. Neben der Möglichkeit des Abbruchs und des Retriggers läuft ein einmal gestarteter Timer auch nach einem CCU- bzw. CUXD-Neustart weiter. Mit dieser Funktion lässt sich z. B. eine Anwesenheitssimulation wie unter [2] beschreiben realisieren.

Exec

Das CUXD-Gerät Exec dient als Ersatz der undokumentierten und nicht sehr stabilen „system.exec()“-Funktion der CCU, mit welcher sich beliebige Prozesse auf der HomeMatic-Zentrale starten lassen. Weiterhin können hiermit bis zu 9 Geräteparameter, 5 Kanalparameter und 99 Parameter-Datenpunkte definiert und als Platzhalter in die Befehlszeile eingebaut werden. Für den direkten Aufruf von URLs ist auch ein automatisches URL-Encoding aller Parameter möglich. Auf der CCU kann dieses Gerät als Taster, Schalter, Jalousieaktor oder Dimmer dargestellt werden. Folgend ein Beispiel zum Ersatz des Original-CCU-„system.Exec()“ durch das „CUXD-Exec“ beim Aufruf durch ein Skript.

Vorher:

```
string stderr;
string stdout;
string url="http://192.168.0.1/web/message?text=Hello_World&type=3&tmout=10";
system.Exec("wget -q -O - „#url, &stdout, &stderr);
```

Nachher:

Für dieses Beispiel muss zuvor im CUXD ein „(28) System-Exec“-Gerät mit der Seriennummer 1 angelegt werden.

```
string url="http://192.168.0.1/web/message?text=Hello_World&type=3&tmout=10";
dom.GetObject("CUXD.CUX2801001:1.CMD_EXEC").State("wget -q -O - „#url);
```

HM-Skript mit Rückgabe der Standardausgabe:

In diesem Beispiel wird das Programm bei der Abfrage von CMD_RETS ausgeführt und die Standardausgabe danach in die Variable v geschrieben.

The screenshot shows the 'CUXD Geräte' configuration page. At the top, there is a dropdown menu for 'CUXD Gerätetyp:' with '(28) System' selected. Below this are several input fields and dropdown menus: 'Funktion:' with a dropdown menu showing 'Timer', 'Exec', 'Multi-DIM-Exec', and 'Ping/Alive'; 'Seriennummer:' with a text input field and a note 'max. 3 Stellen'; 'Name:' with a text input field and a note '(leer = wird autom. generiert)'; 'Geräte-Icon:' with a dropdown menu showing 'Fernbedienung 19 Tasten' and a small icon of a remote control. At the bottom, there is a button labeled 'Gerät auf CCU erzeugen!'.

Bild 4: Zeigt die verschiedenen Funktionen des CUXD-Gerätetyp 28 (System-Devices).



```
dom.GetObject(„CUXD.CUX2801001:1.CMD_SETS“).State(„ping -c 5 192.168.1.1“);
dom.GetObject(„CUXD.CUX2801001:1.CMD_QUERY_RET“).State(1);
var v = dom.GetObject(„CUXD.CUX2801001:1.CMD_RETS“).State();
WriteLine(v);
```

Hierbei ist zu beachten, dass die Abarbeitung des HM-Skripts erst fortgesetzt wird, nachdem das aufgerufene Programm beendet wurde. Während dieser Zeit werden auch keine anderen HM-Skripte ausgeführt!

HM-Skript ohne Rückgabe der Standardausgabe (Beispiel):

Dieser Aufruf ist der Abfrage von CMD_RETS immer dann vorzuziehen, wenn die Ausgabe des aufgerufenen Programms nicht weiterverarbeitet werden muss.

```
dom.GetObject(„CUXD.CUX2801001:1.CMD_SETS“).State(„wget -q -O /dev/null ,http://192.168.0.99:50000/track=neue_email.mp3“);
dom.GetObject(„CUXD.CUX2801001:1.CMD_RUNS“).State(1);
```

Hierbei wird die Abarbeitung des HM-Skripts sofort und unabhängig von der Laufzeit des aufgerufenen Programms fortgesetzt!

Das Beispiel in [Bild 5](#) zeigt den Aufruf eines Systembefehls ohne HM-Skript direkt aus einem Zentralenprogramm über den Datenpunkt **CMD_EXEC**, hiermit kann z. B. sehr einfach ein wöchentliches CCU-Backup auf eine SD-Karte realisiert werden.

Es wird vorausgesetzt, dass die SD-Karte unter dem Verzeichnis „/media/sd-mmcbk0“ erreichbar ist und auf der SD-Karte das Verzeichnis „backup“ angelegt wurde. Das Verzeichnis „backup“ kann mit folgendem Befehl über die CUXD-Service-Seite -> Shell Command erstellt werden.

```
mkdir -p /media/sd-mmcbk0/backup
```

Soll das Backup in einem anderen Verzeichnis abgelegt werden, dann ist der Parameter entsprechend anzupassen. Der Parameter in unserem Beispiel sieht so aus:

```
/usr/local/addons/cuxd/extra/ccu_backup.tcl /media/sd-mmcbk0/backup
```

Meldungen in das System-Logfile schreiben

Über den Datenpunkt **SYSLOG** kann aus einem HM-Skript oder über eine Programmverknüpfung direkt eine Meldung in das Syslog (z. B. /var/log/messages) der CCU geschrieben werden:

```
dom.GetObject(„CUXD.CUX2801001:1.SYSLOG“).State(„eine Statusmeldung“);
```

Ein sinnvoller Anwendungsfall für die Ausgabe von Meldungen im Syslog ist das Debuggen (zur Fehleranalyse) von aufgerufenen Programmverknüpfungen bzw. HM-Skripten im laufenden Betrieb.

Zufallszahl erzeugen:

Da es mittels HM-Skript keine Funktion zur Erzeugung von Zufallszahlen gibt, existieren dafür bereits verschiedene Lösungsvorschläge [5]. Eine noch einfachere Möglichkeit bietet der Datenpunkt **RAND**. Er dient zum Erzeugen von Integer-Zufallszahlen zwischen 0 und MAX. Der MAX-Wert kann durch einen Schreibzugriff auf diesen Datenpunkt gesetzt und jederzeit geändert werden. Außerdem kann für jeden Kanal des Gerätes ein unterschiedlicher MAX-Wert gesetzt werden. Er wird dauerhaft in der CUXD-Gerätekonfiguration gespeichert.

So setzen wir den MAX-Wert auf 255 ...

```
dom.GetObject(„CUXD.CUX2801001:1.RAND“).State(255);
```

... und so erzeugen wir eine Integer-Zufallszahl zwischen 0 und MAX:

```
integer rand = dom.GetObject(„CUXD.CUX2801001:1.RAND“).State().ToInteger();
```

Ping

Dieses Gerät dient zum Prüfen der Erreichbarkeit von Hosts anhand von ICMP-Paketen (Ping) oder per Verbindungsversuch auf konfigurierte TCP-Ports. Anhand der TCP-Ports können beliebige Server-Dienste überwacht werden. Neben dem aktiven Aussen-

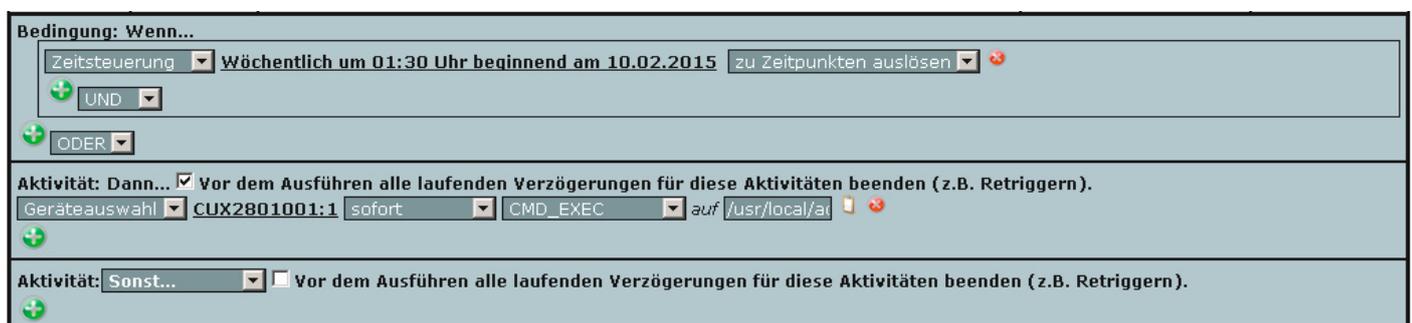


Bild 5: Zentralenprogramm für ein wöchentliches CCU-Backup

den von ICMP-Pings werden auch ankommende ICMP-Pings von den konfigurierten Adressen verarbeitet und starten den Timer für das Sende-Intervall neu. Ist z. B. der Intervall auf 90 s gesetzt und wird alle 60 s vom konfigurierten Host ein Ping an die CCU gesendet, dann erkennt die CCU den Host als ALIVE (erreichbar), ohne dass jemals ein Ping von der CCU an diesen Host gesendet wurde.

Um beim Anpingen eines Smartphones mit aktiviertem WLAN Energie zu sparen, können die Ping-Sende-Intervalle nach Erreichbarkeit und nach Nichterreichbarkeit getrennt voneinander konfiguriert werden. Mit dieser Funktion lässt sich z. B. eine Anwesenheitserkennung von Smartphones per WLAN realisieren, zwei Beispiele hierzu finden sich unter [3] und [4].

Daten-Logging

Das Logging von beliebigen CCU-Geräten mittels CUXD stellt eine performante und ressourcenschonende Möglichkeit zum Aufzeichnen von Daten zur späteren Weiterverarbeitung dar. Da hier mit der Zeit große Datenmengen entstehen können, empfiehlt es sich, das Logfile täglich auf einen Massenspeicher auszulagern (USB-Stick, SD-Karte, NFS-Volume).

SD-Karte mounten:

Die SD-Karte sollte auf der CCU2 über die WebUI (Einstellungen -> Systemsteuerung -> Allgemeine Einstellungen) initialisiert werden. Danach ist sie nach jedem CCU-Neustart automatisch unter dem Verzeichnis /media/sd-mmblk0/ gemountet.

USB-Stick mounten:

Ein USB-Stick muss auf einer CCU immer manuell gemountet werden. Diese Aufgabe kann der CUXD beim Starten erledigen. Dazu sind im CUXD-Setup die folgenden Parameter erforderlich:

```
MOUNTCMD=mount -t vfat /dev/sda1 /mnt
```

```
UMOUNTCMD=umount /mnt
```

Danach ist auf der CUXD-Statusseite die Taste „Mount“ und anschließend die Taste „Geräteeinstellungen speichern“ zu drücken. Die „Mount“-Taste sollte im gedrückten Zustand verbleiben.

Beim Logging werden die Daten vom CUXD in eine Datei geschrieben, diese Datei wird automatisch angelegt und muss vorher nur im CUXD-Setup mittels DEVLOGFILE-Parameter definiert werden. Um die Systemstabilität nicht zu beeinträchtigen, empfiehlt es sich, die aktuelle Logdatei in der RAM-Disk der CCU zu erzeugen (/tmp-Verzeichnis) und dann jeweils 1x täglich mittels DEVLOGMOVE-Parameter auf einen angeschlossenen USB-Stick bzw. die SD-Karte zu verschieben. Das konfigurierte Zielverzeichnis muss existieren.

CUXD-Setup Beispiel-Konfiguration:

```
DEVLOGFILE=/tmp/devlog.txt
```

```
DEVLOGSIZE=
```

```
DEVLOGMOVE=/media/sd-mmblk0/cuxd/devlog
```

Die Log-Datei kann dann über die CUXD-Adminoberfläche unter „Info → Device-Log“ ausgelesen oder mit dem CUXD-Highcharts Add-on [6] grafisch (Bild 8) dargestellt werden.

Welche Datenpunkte von welchem Gerät geloggt werden, kann im CUXD-Setup mit dem Parameter **LOGIT=** festgelegt werden, zudem ist sicherzustellen, dass die CUXD-Parameter **SUBSCRIBE_WR=1** und **SUBSCRIBE_RF=1** im CUXD-Setup gesetzt sind (Bild 6).

In der CUXD-Dokumentation sind alle relevanten Parameter für das Logging ausführlich und mit Beispielen beschrieben.

Zum Loggen von Datenpunkten, die nur bei Wertänderungen und relativ selten übertragen werden, wie z. B. Thermostat-Soll-Temperaturen (SETPPOINT), eignet sich ein periodischer Aufruf (z. B. jede Stunde) der folgenden HM-Befehle:

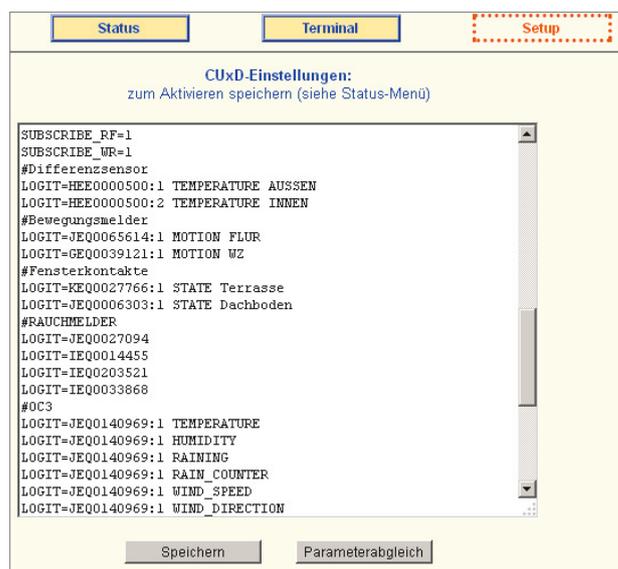


Bild 6: Beispiel-Konfiguration für das CUXD-Logging

```
string s = „HEQxxxxxxx:2.SETPOINT“;
var v = dom.GetObject(„BidCos-RF.“#s).Value();
dom.GetObject(„CUXD.CUX2801001:1.LOGIT“).State(s#“;“#v);
```

Im Beispiel wird ein CUXD-System.Exec()-Gerät mit der Seriennummer 1 genutzt. Die Seriennummer des zu loggenden Gerätes ist entsprechend anzupassen.

Neben dem Logging von Komponenten können über eine Programmverknüpfung mit CUXD auch Systemvariablen geloggt werden (Bild 7). Hierzu sind je nach Art des Ausführens folgende Skripte zu verwenden:

Skript zum Logging bei Aktualisierung:

```
object o = dom.GetObject(„$src“);
if (o) {
    dom.GetObject(„CUXD.CUX2801001:1.LOGIT“).State(o.Name()#“;
    “#o.Value());
}
```

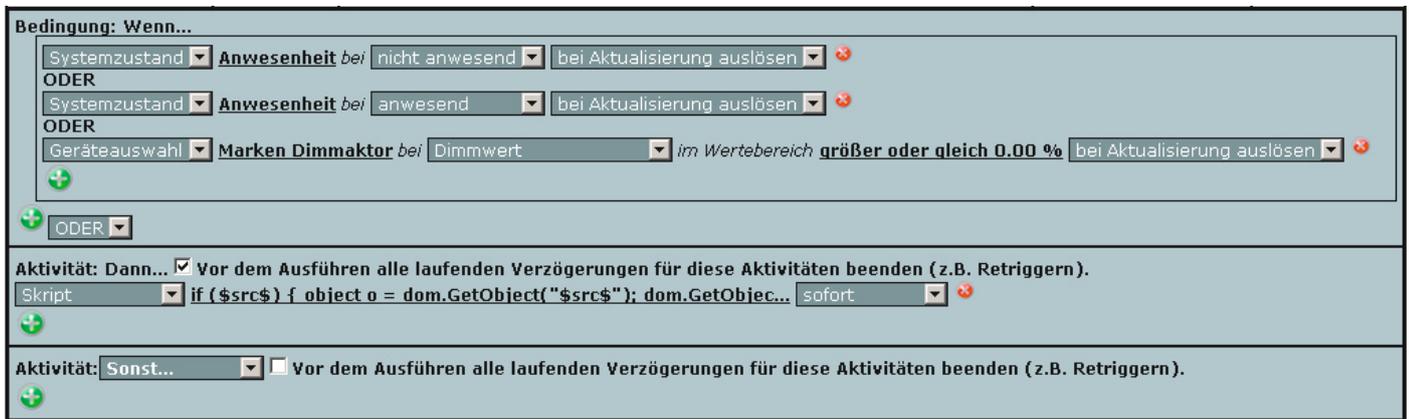


Bild 7: Zentralenprogramm zum Loggen von Systemvariablen

Skript zum Logging bei Änderung:

```
object o = dom.GetObject(„$src$“);
if (o) {
    if (o.Value() <> o.LastValue()) {
        dom.GetObject(„CUXD.CUX2801001:1.LOGIT“).State(o.Name()#“;“#o.Value());
    }
}
```

CUXD-Highcharts

Zur grafischen Darstellung der mittels CUXD geloggtten Daten kann die CCU-Zusatzsoftware CUXD-Highcharts [6] genutzt werden (Bild 8).

Beispiel-Konfiguration mit Ablage der Logfiles auf der CCU2-SD-Karte:

- CUXD- und Highcharts-Add-on installieren
- SD-Karte per WebUI initialisieren
- auf CUXD-Service-Seite
 - > Shell Command: mkdir -p /media/sd-mmcbk0/cuxd/devlog
 - > Ausführen
- im CUXD-Setup die folgenden Parameter setzen (dürfen nur 1x vorhanden sein!):
 - DEVLOGFILE=/tmp/devlog.txt
 - DEVLOGSIZE=
 - DEVLOGMOVE=/media/sd-mmcbk0/cuxd/devlog
- die gesetzten Parameter auf der CUXD-Statusseite überprüfen
- Aufruf der Diagramme über CUXD -> Info -> Device-Log -> Chart bzw. über <http://CCU-IP-Adresse/addons/cuxchart/menu.html>

Aufgrund des großen Funktionsumfangs und der Komplexität kann ELV zur Zusatzsoftware leider keinerlei Support übernehmen. Bei alle Fragen zu CUXD steht Ihnen allerdings das HomeMatic-Forum [1] zur Verfügung, welches durch viele erfahrene User und auch den Entwickler selbst betreut wird und somit als Support-Plattform dient. **ELV**

**Weitere Infos:**

- [1] www.cuxd.de
- [2] homematic-forum.de/forum/viewtopic.php?f=18&t=14227&p=110910#p111460
- [3] homematic-forum.de/forum/viewtopic.php?f=37&t=19891
- [4] homematic-forum.de/forum/viewtopic.php?f=31&t=14058
- [5] www.homematic-inside.de/tecbase/homematic/scriptlibrary/item/random
- [6] www.homematic-inside.de/software/cuxd-highcharts

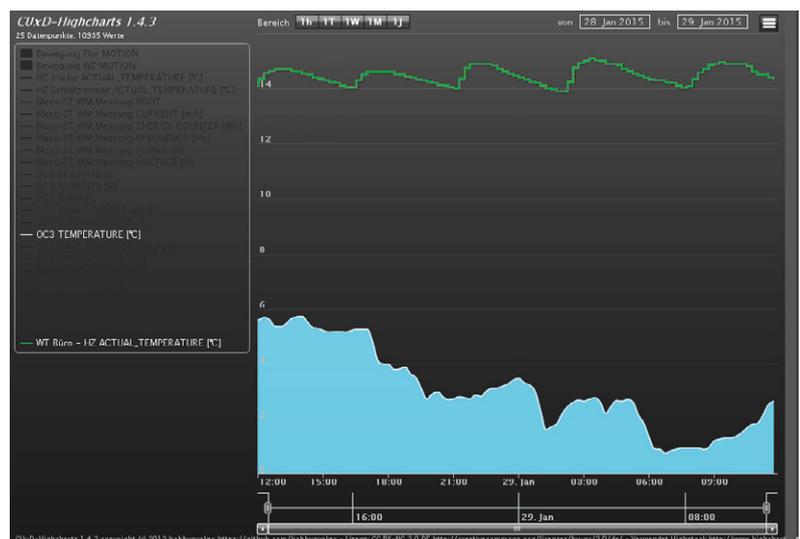


Bild 8: Zusatzsoftware CUXD-HighCharts