

CCU-Historian

V1.0.0

Handbuch

MDZ

info@ccu-historian.de

Inhalt

| | | |
|-------|---|----|
| 1 | Einleitung..... | 1 |
| 1.1 | Zielsetzungen..... | 1 |
| 1.2 | Umsetzung..... | 1 |
| 2 | Zusätzliche Funktionalität | 2 |
| 2.1 | Vorverarbeitung aller Wertaktualisierungen | 2 |
| 2.1.1 | Delta Kompression | 3 |
| 2.2 | Zwischenspeicherung von Wertänderungen im Arbeitsspeicher | 3 |
| 2.3 | Web-Server..... | 4 |
| 3 | Installation..... | 4 |
| 3.1 | Update einer vorhandenen Installation | 4 |
| 3.2 | Sicherungskopie der Datenbank erstellen | 4 |
| 4 | Konfiguration..... | 5 |
| 4.1 | Liste aller Optionen | 6 |
| 4.2 | Konfiguration der angeschlossenen Geräte | 8 |
| 4.2.1 | Optionen für eine CCU1 oder CCU2 | 9 |
| 4.2.2 | Optionen für eine HM BINRPC-Schnittstelle | 9 |
| 4.2.3 | Beispiele | 10 |
| 4.3 | Fehlersuche in der Konfigurationsdatei | 10 |
| 4.4 | Automatisierte Erstellung von Backups der Datenbank | 11 |
| 4.5 | Eigene Web-Seiten einbinden | 12 |
| 4.6 | Firewall-Einstellungen | 13 |
| 5 | Start | 13 |
| 5.1 | Startparameter | 14 |
| 6 | Datenbank | 16 |
| 6.1 | Historientabellen für Gerätedatenpunkte | 16 |
| 6.2 | Historientabellen für Systemvariablen..... | 17 |
| 6.3 | Tabelle DATA_POINTS mit Meta-Informationen..... | 17 |
| 6.4 | Beispiele für SQL-Anfragen..... | 18 |
| 6.5 | Web-Bedienschnittstelle | 19 |
| 6.6 | ODBC-Schnittstelle | 20 |
| 7 | Web-Schnittstelle | 21 |

| | | |
|-------|--|----|
| 7.1 | Standard Web-Seiten | 21 |
| 7.1.1 | Hauptseite / Übersicht Datenpunkte | 22 |
| 7.1.2 | Detail-Seite zu einem Datenpunkt | 22 |
| 7.1.3 | Mehrere Datenpunkte als Trend darstellen | 24 |
| 7.1.4 | Verwaltung Datenpunkte | 24 |
| 7.1.5 | Werkzeuge..... | 25 |
| 7.1.6 | Konfiguration | 26 |
| 7.2 | JSON-RPC Schnittstelle | 26 |
| 7.2.1 | Anfragearten | 27 |
| 7.2.2 | Rückgaben | 28 |
| 7.2.3 | Methoden | 29 |
| 7.3 | Export von Zeitreihen | 34 |
| 7.4 | Generierung von Trend-Diagrammen | 37 |
| 7.4.1 | Anpassung der Trend-Darstellung | 40 |
| 7.5 | Eigene Web-Seiten | 48 |
| 7.5.1 | Weiterführende Dokumentation..... | 49 |
| 7.5.2 | Vordefinierte Variablen für die Skripte | 49 |
| 8 | Unterstützung..... | 49 |
| 8.1 | Erstellung eines Fehlerprotokolls | 50 |
| 9 | Anhang..... | 50 |
| 9.1 | Änderungshistorie | 50 |

1 Einleitung

Der CCU-Historian stellt ein Langzeitarchiv für die HomeMatic-Zentrale (CCU) zur Verfügung. In dem Langzeitarchiv werden Kommunikationsvorgänge der CCU-Schnittstellen (BidCos-RF, BidCos-Wired und System) aufgezeichnet. Darunter befinden sich z.B. die Messwerte aller Sensoren und alle ausgeführten Schaltvorgänge. Aus der Logikschicht der CCU werden zusätzlich die Systemvariablen archiviert. Die gesammelten Daten werden in einer Datenbank abgelegt und stehen daraufhin für Visualisierungen oder Analysen zur Verfügung. Für einen ersten Überblick werden Web-Seiten mit Trend-Diagrammen durch einen eingebetteten Web-Server generiert.

1.1 Zielsetzungen

Folgende Zielsetzungen waren und sind bei der Entwicklung maßgebend:

1. Keine Software-Installation oder Konfiguration auf der CCU.
2. Verwendung von offiziellen bzw. gut dokumentierten Schnittstellen.
3. Möglichst geringe Belastung der CCU.
4. Selbstkonfiguration des Langzeitarchivs.
5. Verwendung einer Datenbank mit offenen Schnittstellen.
6. Einfacher Export der Daten über eine Web-Schnittstelle.

1.2 Umsetzung

Im Folgenden wird die Umsetzung der einzelnen Ziele näher erläutert:

1. Auf der CCU sind für den CCU-Historian keine zusätzlichen Software-Pakete nötig. Der CCU-Historian kann direkt mit einer neu ausgelieferten CCU verwendet werden. Des Weiteren müssen auch keine gesonderten Programme oder Systemvariablen auf der CCU angelegt werden.
2. Der CCU-Historian verwendet die *HomeMatic XML-RPC* bzw. *BIN-RPC API* und die *Remote HomeMatic-Script API*. Bei der Script API wird auf die Verwendung des nicht dokumentierten Befehls *system.Exec* verzichtet.
3. Die Kommunikationsvorgänge des CCU-Historians mit der CCU werden auf das Notwendige beschränkt. Datenbankabfragen sind völlig von der CCU entkoppelt, und belasten nur den Rechner auf dem der CCU-Historian läuft.
4. Nach der ersten Konfiguration des CCU-Historians für die Inbetriebnahme, ist keine weitere Konfigurationspflege mehr nötig. Insbesondere wenn Geräte an der CCU angelern oder entfernt werden, aktualisiert der CCU-Historian selbstständig die Verwaltungsstrukturen in der Datenbank. Von neu angelerten Geräten werden automatisch alle Kanäle und die zugehörigen Parameter erkundet.
5. Auf Grund der zu erwartenden Datenmengen und den vielfältigen Abfragemöglichkeiten wird eine Datenbank mit SQL-Unterstützung und den Schnittstellen *JDBC* und *ODBC* verwendet.

Dadurch können auch mächtige Report-Werkzeuge wie z.B. BIRT (<http://www.eclipse.org/birt/phoenix>) auf die Daten zugreifen.

6. In dem CCU-Historian ist ein Web-Server eingebettet, der für das einfache Exportieren von Zeitreihen und die Generierung von Trend-Diagrammen zuständig ist. Über eine URL mit zusätzlichen Parametern, die im einfachsten Fall über einen Web-Browser eingegeben wird, wird die Abfrage angestoßen.

Der CCU-Historian ist in der Programmiersprache *Groovy* (<http://www.groovy-lang.org/>) implementiert und benötigt für die Ausführung *Java* in der Version 7. Er ist damit plattformunabhängig und kann somit auch z.B. unter Linux eingesetzt werden.

Um die zusätzliche Installation einer Datenbank zu vermeiden, wurde die *H2 Database Engine* (<http://www.h2database.com>) in den CCU-Historian eingebettet. Diese stellt zusätzlich eine Web-Bedienschnittstelle zur Verfügung. Dadurch kann die Datenbank leicht über einen Web-Browser verwaltet werden.

Über eine Web-Schnittstelle kann der CCU-Historian Daten exportieren oder darstellen. Dazu ist der Web-Server Jetty (<http://www.eclipse.org/jetty/>) ebenfalls im CCU-Historian eingebettet. Zusätzlich zu den HTML-Seiten des CCU-Historians kann der Anwender eigene HTML-Seiten über den Web-Server anbieten. Dadurch kann der Anwender beliebige Übersichtsseiten mit aktuellen Messwerten und/oder Trend-Diagrammen erstellen.

Für die Generierung von Trend-Diagrammen wurde die Bibliothek JFreeChart (<http://www.jfree.org/jfreechart/>) verwendet.

2 Zusätzliche Funktionalität

Neben der Grundfunktionalität eines Langzeitarchivs bietet der CCU-Historian zusätzliche Möglichkeiten, die im Folgenden beschrieben sind.

2.1 Vorverarbeitung aller Wertaktualisierungen

Die Wertaktualisierungen aller Datenpunkte durchlaufen eine Vorverarbeitung bevor sie in der Datenbank abgelegt werden. Die Vorverarbeitung dient dazu, die anfallende Datenmenge zu reduzieren indem unnötige Wertaktualisierungen verworfen werden und/oder bestimmte Bereinigungsfunktionen auf den Werteverlauf eines Datenpunktes anzuwenden.

Die Konfiguration erfolgt über die Web-Bedienschnittstelle des CCU-Historians. Je Datenpunkt kann eine individuelle Vorverarbeitung aktiviert und konfiguriert werden (s.a. Abschnitt 7.1.4).

Zurzeit unterstützte Algorithmen zur Vorverarbeitung:

| Name | Parameter | Beschreibung |
|------------------------------|--|--|
| Delta Kompression | Erforderliche Wertedifferenz (Standardwert: 0,0) | Bei der Delta Kompression wird die Wertedifferenz von zwei aufeinanderfolgenden Wertaktualisierungen betrachtet. Nur wenn diese größer oder gleich dem angegebenen Parameter ist, wird die Wertaktualisierung in der Datenbank abgelegt. |
| Zeitliche Kompression | Zeitlicher Mindestabstand in Sekunden | Nur Wertaktualisierungen, die einen zeitlichen Mindestabstand zur vorhergehenden Aktualisierung besitzen, werden in der Datenbank gespeichert. |

2.1.1 Delta Kompression

Mit der *Delta Kompression* ist es möglich, ohne oder mit nur einem geringen Qualitätsverlust, die aufgezeichnete Datenmenge zum Teil erheblich zu reduzieren. Der Parameterwert gibt dabei die maximale Abweichung der aufgezeichneten Werte zu den tatsächlichen Werten vor.

Diese Kompression ist bei Datenpunkte vom Typ *ACTION* nicht sinnvoll, da diese Datenpunkte immer den gleichen Wert (1,0) besitzen.

Beispiel:

Ein Temperatúraußensensor liefert etwa 567 Wertaktualisierungen pro Tag. Bei aktivierter *Delta Kompression* mit einem Parameterwert von 0,01, der unterhalb der Messgenauigkeit von 0,1 °C liegt, reduziert sich die Menge auf beispielsweise 191 Wertaktualisierungen pro Tag. Das heißt, die Datenmenge wird fast um Faktor 3 ohne einen Verlust an Qualität des Temperaturverlaufs reduziert.

2.2 Zwischenspeicherung von Wertänderungen im Arbeitsspeicher

Wertänderungen können im Arbeitsspeicher zwischengespeichert werden. Die ankommenden Wertänderungen werden dann nicht mehr sofort in die Datenbank geschrieben, was einen häufigen Zugriff auf das Speichermedium (Festplatte, Speicherkarte, USB-Stick) der Datenbank vermeidet. Wenn zusätzlich das Datei-Log durch Setzen der Option *logSystem.fileLevel=Level.OFF* (Standardeinstellung) abgeschaltet wird, und der Erkundungszyklus durch Setzen der Option *historian.metaCycle* verlängert wird, so erfolgt für einen bestimmten Zeitraum keinerlei Zugriff auf das Speichermedium. Bei einem Systemabsturz gehen aber alle Wertänderungen, die noch im Arbeitsspeicher liegen, verloren.

Die Zwischenspeicherung wird durch das Setzen der Option *historian.bufferTime* aktiviert. Zusätzlich kann die maximale Anzahl der zwischengespeicherten Werte durch die Option *historian.bufferCount* gesetzt werden.

Folgende Konfigurationsoptionen müssen zum Beispiel für eine Zwischenspeicherung von einem Tag gesetzt werden:

```
logSystem.fileLevel=Level.OFF
historian.metaCycle=24*60*60*1000
historian.bufferTime=24*60*60*1000
```

2.3 Web-Server

Der CCU-Historian besitzt einen eingebetteten Web-Server. Über diesen wird die Web-Oberfläche des CCU-Historians angeboten. Die dazugehörigen Dateien liegen im Unterverzeichnis *webapp* des Installationsordners. Dort können auch beliebige benutzerdefinierte Dateien im Verzeichnis *webapp/custom* abgelegt werden. Diese können dann unter der Adresse <http://localhost/custom> abgerufen werden.

Weitere Informationen sind in den Abschnitten 4.5 und 7.5 zu finden.

3 Installation

Der CCU-Historian wird in einer ZIP-Datei (ccu-historian-X.Y.Z-bin.zip) mit folgendem Inhalt zum Download angeboten:

- ccu-historian-sample.config (Beispielkonfigurationsdatei)
- ccu-historian.exe (Startprogramm für MS Windows)
- ccu-historian.jar (Java-Archiv mit der Applikation)
- CCU-Historian_Kurzanleitung.pdf (Dokumentation)
- Lizenz.txt (Hinweise zur rechtmäßigen Verwendung der Software)
- webapp (Verzeichnis mit den Standard-Web-Seiten)

Für die Installation reicht es, die ZIP-Datei in ein leeres Verzeichnis zu entpacken. Das gesamte Verzeichnis kann später an einen beliebigen anderen Ort verschoben werden, und der CCU-Historian ohne Konfigurationsänderung wieder gestartet werden.

3.1 Update einer vorhandenen Installation

Die ZIP-Datei kann direkt in eine vorhandene Installation entpackt werden. Die Konfigurationsdatei *ccu-historian.config* und die vorhandene Datenbank werden nicht überschrieben. Normalerweise können die Konfigurationsdatei und die Datenbank ohne manuelle Anpassungen mit neueren Versionen des CCU-Historians verwendet werden.

Hinweis: Vor dem Einspielen eines Updates sollte immer eine Sicherungskopie der Datenbank und der bisherigen Installation erstellt werden (s.a. Abschnitt 3.2).

3.2 Sicherungskopie der Datenbank erstellen

Das komplette Archiv, also alle Zeitreihen und die nötigen Verwaltungsinformationen, werden in einer Datei abgelegt. Der Dateiname wird durch die Konfigurationsoptionen *database.dir* (Standardeinstellung **./data**) und *database.name* (Standardeinstellung **history**) vorgegeben. Die Dateiendung ist immer **.h2.db**. Daraus resultiert dann der vollständige Dateipfad **<Installationsverzeichnis vom CCU-Historian>/data/history.h2.db**.

Mit einer einfachen Dateikopie kann das komplette Archiv an anderer Stelle gesichert werden. Der CCU-Historian darf während des Kopierens nicht gestartet sein.

4 Konfiguration

Die Konfiguration erfolgt über die Datei *ccu-historian.config*. Diese Datei ist nicht im Installationspaket vorhanden. Sie muss durch Kopieren oder Umbenennen der Datei *ccu-historian-sample.config* erst erstellt werden.

Die Konfigurationsdatei kann mit einem beliebigen Text-Editor bearbeitet werden. In der Datei befindet sich eine Liste an Konfigurationsoptionen in der Form:

Konfigurationsoption=Wert

Alle Konfigurationsoptionen besitzen einen Standardwert. Nur vom Standard abweichende Konfigurationsoptionen müssen deshalb gesetzt werden.

Kommentarzeilen werden mit zwei Schrägstrichen (//) eingeleitet. Ein ganzer Block kann durch die Verwendung von /* am Anfang und */ am Ende auskommentiert werden. Zeichenkettenwerte (z.B. die Adresse der CCU) müssen in einfachen Hochkommata (') eingeschlossen werden. Eine Liste wird mit einer eckigen Klammer ([]) eingeleitet und mit einer eckigen Klammer (]) abgeschlossen. Die einzelnen Elemente werden durch Kommata (,) getrennt.

Bei Konfigurationsoptionen, mit denen ein Datei- oder Verzeichnispfad gesetzt wird, muss als Pfadtrennzeichen ein plattformunabhängiger Schrägstrich (/) verwendet. Der unter MS Windows übliche rückwärtige Schrägstrich (\) darf nicht verwendet werden. Relative Pfade beginnen mit Punkt und Schrägstrich (./) und werden vom Verzeichnis der Datei ccu-Historian.exe aus ausgewertet.

Im einfachsten Fall reicht das Setzen der Konfigurationsoptionen *devices.device1.type* und *devices.device1.address* für den Betrieb des CCU-Historians aus. Falls z.B. eine CCU1 mit der IP-Adresse 192.168.0.2 verbunden werden soll, müssen die beiden Konfigurationsoptionen wie folgt gesetzt werden:

```
devices.device1.type=CCU1
devices.device1.address='192.168.0.2'
```

Bei einigen Java VMs funktioniert die automatische Erkennung der eigenen IP-Adresse des CCU-Historian-Rechners nicht. Dann müssen folgende zwei Optionen manuell auf die IP-Adresse des CCU-Historian-Rechners gesetzt werden, z.B. für die IP-Adresse 192.168.0.5:

```
devices.historianAddress='192.168.0.5'
webServer.historianAddress='192.168.0.5'
```

Wenn ein anderes Passwort für die Datenbank verwendet werden soll, so sollte dies vor dem ersten Start gesetzt werden:

```
database.password='NeuesPasswort'
```


Für die Fehlersuche ist das Erstellen von Log-Dateien mit weiteren Informationen hilfreich:

logSystem.fileLevel=Level.FINER

Die Konfigurationsdatei kann im laufenden Betrieb des CCU-Historians angepasst werden. Zum Zeitpunkt des Speicherns sollte sie natürlich in einem gültigen Zustand vorliegen. Die geänderte Konfiguration wird spätestens nach 15 Sekunden neu geladen. Etwaige Fehler werden im Konsolenfenster angezeigt.

4.1 Liste aller Optionen

| Optionsname | Standardwert | Beschreibung |
|--------------------------------|--------------------------|---|
| logSystem.consoleLevel | Level.INFO | Meldungsfilter für die Konsole Mögliche Werte: Level.OFF, Level.SEVERE, Level.WARNING, Level.INFO, Level.CONFIG, Level.FINE, Level.FINER, Level.FINEST, Level.ALL |
| logSystem.fileLevel | Level.OFF | Meldungsfilter für die Log-Datei Mögliche Werte: Level.OFF, Level.SEVERE, Level.WARNING, Level.INFO, Level.CONFIG, Level.FINE, Level.FINER, Level.FINEST, Level.ALL |
| logSystem.fileName | './ccu-historian-%g.log' | Dateiname für die Log-Datei(en) %g wird durch eine aufsteigende Nummer ersetzt. |
| logSystem.fileLimit | 1000000 | Dateigrößenbegrenzung für Log-Dateien in Bytes |
| logSystem.fileCount | 5 | Max. Anzahl an Log-Dateien |
| logSystem.binRpcLevel | Level.WARNING | Zusätzlicher Filter für Log-Meldungen der BINRPC-Schnittstelle |
| database.dir | './data' | Verzeichnis für die Ablage der Datenbank |
| database.name | 'history' | Datenbankname |
| database.user | 'sa' | Datenbankbenutzer |
| database.password | 'ccu-historian' | Datenbankpasswort |
| database.webEnable | true | Aktivierung der Web-Bedienschnittstelle: false=aus; true=ein |
| database.webPort | 8082 | Netzwerk-Port für die Web-Bedienschnittstelle |
| database.webAllowOthers | false | Zugriff über das Netzwerk auf die Web-Bedienschnittstelle erlauben: false=aus; true=ein |
| database.tcpEnable | false | Aktivierung der TCP-Schnittstelle: false=aus; true=ein |
| database.tcpPort | 9092 | Netzwerk-Port für die TCP-Schnittstelle |
| database.tcpAllowOthers | false | Zugriff über das Netzwerk auf die TCP - Schnittstelle erlauben: false=aus; true=ein |
| database.pgEnable | false | Aktivierung der PostgreSQL-Schnittstelle: false=aus; true=ein |
| database.pgPort | 5435 | Netzwerk-Port für die PostgreSQL-Schnittstelle |
| database.pgAllowOthers | false | Zugriff über das Netzwerk auf die PostgreSQL - Schnittstelle erlauben: false=aus; true=ein |

| | | |
|-----------------------------------|---------------|---|
| database.backup | " | Dateipfad mit Platzhaltern zur Ablage von Backups der Datenbank. Mit dem Standardwert ist diese Funktionalität abgeschaltet. Für eine genaue Beschreibung siehe Abschnitt 4.3. |
| historian.metaCycle | 3600000 | Zykluszeit für die Abfrage von Meta-Informationen [Millisekunden]; 0=Abfrage deaktiviert |
| historian.bufferCount | 5000 | Max. Anzahl an Wertänderungen, die im Arbeitsspeicher gepuffert werden sollen. Diese Option ist nur wirksam, wenn <i>historian.bufferTime</i> größer als 0 gesetzt wurde. |
| historian.bufferTime | 0 | Max. Zeitspanne, in der Wertänderungen im Arbeitsspeicher gepuffert werden sollen. [Millisekunden]; 0=Pufferung ist deaktiviert |
| webServer.port | 80 | Port für den eingebetteten Web-Server |
| webServer.dir | './webapp' | Wurzelverzeichnis für die Dateien und Skripte, die über den Web-Server abrufbar sein sollen. |
| webServer.logLevel | Level.WARNING | Zusätzlicher Meldungsfilter für Log-Meldungen des eingebetteten Web-Servers. Jede Meldung wird wie immer auch durch <i>logSystem.consoleLevel</i> und <i>logSystem.fileLevel</i> gefiltert. |
| webServer.historianAddress | " | Unter der angegebenen Adresse ist der Web-Server von außerhalb erreichbar. Das Setzen dieser Option ist nur sinnvoll, wenn der CCU-Historian-Rechner über mehrere Netzwerkschnittstellen (z.B. auch VPN-Verbindungen) verfügt oder die automatische Erkennung nicht funktioniert . Diese Option sollte immer zusammen mit devices.historianAddress gesetzt werden. Leer: Die Adresse wird automatisch ermittelt. |
| webserver.trendDesigns | [:] | Definition verschiedener Trend-Darstellungsformen. Ein Satz an Einstellungen (Farben, Strichstärken, usw.) wird unter einem frei gewählten Bezeichner abgespeichert. Die Einstellungen können nachher bei der Trend-Generierung abgerufen werden. Eine detaillierte Beschreibung ist im Abschnitt 7.4.1 zu finden. |
| webServer.apiKeys | [] | Der Export von Zeitreihen (s.a. Abschnitt 7.3), die Generierung von Trend-Diagrammen (s.a. Abschnitt 7.4) und die JSON-RPC Schnittstelle (s.a. Abschnitt 7.2) können durch Schlüsselwörter geschützt werden. Der Anfragende muss bei Benutzung einer der Schnittstellen eines der Schlüsselwörter mit übertragen. Beispiel mit einem Schlüsselwort <i>abc</i> : ['abc'] Beispiel mit zwei zulässigen Schlüsselwörtern: ['abc', 'def'] |
| webServer.menuLinks | [:] | Einbindung von eigenen Web-Verweisen in das Menü der Web-Applikation des CCU-Historians (s.a. Abschnitte 2.3, 4.5 und 7.5). |

4.2 Konfiguration der angeschlossenen Geräte

Im folgenden einleitenden Beispiel wird eine CCU1 mit einem zusätzlich installierten CUxD konfiguriert:

```
devices.device1.type=CCU1
devices.device1.address='192.168.0.5'
devices.device1.plugin1.type=CUXD
```

Die Optionen für die Konfiguration der angeschlossenen Geräte haben als ersten Namensbestandteil (bis zum ersten Punkt) das Schlüsselwort *devices*.

Folgende Konfigurationsoptionen gelten für alle angeschlossenen Geräte:

| Optionsname | Standardwert | Beschreibung |
|---------------------------------|--------------|---|
| devices.historianAddress | " | Die angegebene Adresse wird bei der CCU als Zieladresse für die Benachrichtigungen registriert. Das Setzen dieser Option ist nur sinnvoll, wenn der CCU-Historian-Rechner über mehrere Netzwerkschnittstellen (z.B. auch VPN-Verbindungen) verfügt oder die automatische Erkennung nicht funktioniert . Diese Option sollte immer zusammen mit <code>webServer.historianAddress</code> gesetzt werden. Leer: Die Adresse wird automatisch ermittelt. |
| devices.historianRpcPort | 2010 | Netzwerk-Port für den XMLRPC-Server des CCU-Historians |

Der zweite Namensbestandteil (zwischen dem ersten und dem zweiten Punkt) fasst die Optionen für ein Gerät zusammen. Für diesen Namensbestandteil sind die Schlüsselwörter *device1* bis *device10* erlaubt. Der CCU-Historian kann sich also mit bis zu zehn Geräten verbinden. Im Folgenden wird für die Gerätenummer 1 bis 10 der Platzhalter <G-Nr> verwendet. Die Gerätenummern können in beliebiger Reihenfolge sein und müssen nicht fortlaufend vergeben werden.

Für alle Geräte sind folgende Optionen zu setzen:

| Optionsname | Standardwert | Beschreibung |
|---|----------------------------|--|
| devices.device<G-Nr>.address | Option muss gesetzt werden | Die Adresse des Gerätes. Meistens handelt es sich um die IP-Adresse der CCU. |
| devices.device<G-Nr>.type | Option muss gesetzt werden | Zulässige Gerätetypen sind z.B.: CCU1, CCU2, BINRPC |

Ein Gerät kann eventuell durch zusätzliche Hardware und/oder Software erweitert werden. Zum Beispiel kann eine CCU1 mit der Software *CUxD* und der zugehörigen Hardware *FS20*-Komponenten unterstützen, oder eine CCU2 wird mit dem *HomeMatic Wired RS485 LAN Gateway* betrieben. Die Konfiguration dieser Plug-Ins erfolgt über einen dritten Namensbestandteil. Bis zu zehn Plug-Ins können über die Schlüsselwörter *plugin1* bis *plugin10* konfiguriert werden. Im Folgenden wird für die

Plug-In-Nummer 1 bis 10 der Platzhalter <P-Nr> verwendet. Die Plug-In-Nummern können in beliebiger Reihenfolge sein und müssen nicht fortlaufend vergeben werden.

Für alle Plug-Ins sind folgende Optionen zu setzen:

| Optionsname | Standardwert | Beschreibung |
|---|----------------------------|---|
| devices.device<G-Nr>.plugin<P-Nr>.type | Option muss gesetzt werden | Zulässige Plug-In-Typen sind z.B.: CUXD, HMWLGW |

4.2.1 Optionen für eine CCU1 oder CCU2

In diesem Abschnitt sind die zusätzlichen Konfigurationsoptionen für ein Gerät vom Typ CCU1 oder CCU2 beschrieben.

| Optionsname | Standardwert | Beschreibung |
|---|--------------|--|
| devices.device<G-Nr>.timeout | 10000 | Auf BIN-RPC-Anfragen muss die CCU innerhalb der angegebenen Zeitspanne antworten, ansonsten wird ein Fehler ausgelöst. [Millisekunden] |
| devices.device<G-Nr>.reinitTimeout | 300000 | Falls längere Zeit keine Nachrichten von der CCU empfangen worden sind, wird die BIN-RPC API neu initialisiert. Die Zeit wird in Millisekunden angegeben. |
| devices.device<G-Nr>.writeAccess | false | Schreibzugriff auf der CCU erlauben. Dadurch kann z.B. der aktuelle Zustand eines Datenpunktes geändert werden. |
| devices.device<G-Nr>.sysVarDataCycle | 30000 | Zykluszeit für das Lesen von Systemvariablen [Millisekunden] |
| devices.device<G-Nr>.prefix | " | Falls mehrere Geräte vom gleichen Typ verwendet werden (auch CCU1 mit CCU2), müssen die Geräte durch ein Prefix unterschieden werden. Dieses Prefix wird z.B. in den Tabellennamen eingefügt. Es sind keine Sonderzeichen erlaubt. |

Für eine CCU2 sind die Plug-Ins *CUXD* und *HMLGW* zulässig, für eine CCU1 nur das Plug-In *CUXD*.

4.2.2 Optionen für eine HM BINRPC-Schnittstelle

Der CCU-Historian kann so konfiguriert werden, dass er ein beliebiges Gerät, das eine *HomeMatic BINRPC-Schnittstelle* besitzt, abfragt und archiviert.

Es wird zurzeit nur die binäre Version (BINRPC) der *HomeMatic XML-RPC-Schnittstelle v1.502* (zu finden auf <http://www.eq-3.de/software.html>) unterstützt.

| Optionsname | Standardwert | Beschreibung |
|--|----------------------------|---|
| devices.device<G-Nr>.name | Option muss gesetzt werden | Name der Schnittstelle; Er darf keine Leerzeichen oder Sonderzeichen enthalten. |

| | | |
|---|----------------------------|--|
| devices.device<G-Nr>.port | Option muss gesetzt werden | Falls mehrere Geräte vom gleichen Typ verwendet werden (auch CCU1 mit CCU2), müssen die Geräte durch ein Prefix unterschieden werden. Dieses Prefix wird z.B. in den Tabellennamen eingefügt. Es sind keine Sonderzeichen erlaubt. |
| devices.device<G-Nr>.timeout | 10000 | Auf BIN-RPC-Anfragen muss das Gerät innerhalb der angegebenen Zeitspanne antworten, ansonsten wird ein Fehler ausgelöst. [Millisekunden] |
| devices.device<G-Nr>.reinitTimeout | 300000 | Falls längere Zeit keine Nachrichten von dem Gerät empfangen worden sind, wird die BIN-RPC API neu initialisiert. Die Zeit wird in Millisekunden angegeben. |
| devices.device<G-Nr>.writeAccess | false | Schreibzugriff auf das Gerät erlauben. Dadurch kann z.B. der aktuelle Zustand eines Datenpunktes geändert werden. |

Ein BINRPC-Gerät besitzt niemals Plug-Ins.

4.2.3 Beispiele

Eine CCU1

```
devices.device1.type=CCU1
devices.device1.address='192.168.0.5'
```

CCU2 mit HMWLGW und CUxD

```
devices.device1.type=CCU2
devices.device1.address='192.168.0.5'
devices.device1.plugin1.type=HMWLGW
devices.device1.plugin2.type=CUXD
```

Eine CCU1 mit CUxD und eine CCU2 mit HMWLGW

```
devices.device1.type=CCU1
devices.device1.address='192.168.0.5'
devices.device1.prefix='HOME_'
devices.device1.plugin1.type=CUXD
devices.device2.type=CCU2
devices.device2.address='192.168.0.6'
devices.device2.prefix='WORK_'
devices.device2.plugin1.type=HMWLGW
```

4.3 Fehlersuche in der Konfigurationsdatei

Bei einer fehlerhaften Konfigurationsdatei erscheint normalerweise nur folgende Meldung:

```
Exception: Configuration file ccu-historian.config is invalid
```

Weitere Informationen zu Fehlern können auch über die Kommandozeile aktiviert werden.
(Normalerweise geschieht dies in der Konfigurationsdatei, diese ist aber defekt.)

```
ccu-historian.exe -loglevel finest
```

bzw.

```
java -jar ccu-historian.jar -loglevel finest
```

Detaillierte Fehlerinformationen (insbesondere für den Entwickler) sind wie folgt zu finden:

```
2016-10-22 22:00:40!SEVERE !Exception: Configuration file ccu-historian.config is invalid
2016-10-22 22:00:40!FINE !java.lang.Exception: Configuration file ccu-historian.config is invalid
    at mdz.ccuhistorian.Configuration.readFile(Configuration.groovy:122)
    at mdz.ccuhistorian.Main.start(Main.groovy:64)
    at mdz.ccuhistorian.Main.access$0(Main.groovy)
    at mdz.ccuhistorian.Main$_run_closure4.doCall(Main.groovy:55)
    at mdz.ccuhistorian.Main$_run_closure4.call(Main.groovy)
    at mdz.ccuhistorian.LogSystem.catchIoLog(LogSystem.groovy:78)
    at mdz.ccuhistorian.Main.run(Main.groovy:55)
    at mdz.ccuhistorian.Main.main(Main.groovy:23)
Caused by: groovy.lang.MissingPropertyException: No such property: unbekannt for class: Config
    at Config.run(Config:4)
    at mdz.ccuhistorian.Configuration.readFile(Configuration.groovy:120)
    ... 7 more
```

Zeile des Fehlers

Fehlermeldung

4.4 Automatisierte Erstellung von Backups der Datenbank

Es können automatisiert Backups der Datenbank erstellt werden. Diese Funktionalität wird mit der Option *database.backup* konfiguriert. Durch diese Option wird ein Dateipfad mit bestimmten Platzhaltern zur Ablage der erstellten Backup-Dateien angegeben. Relative Dateipfade beziehen sich auf das Arbeitsverzeichnis vom CCU-Historian. Dies ist normalerweise das Installationsverzeichnis.

Hinweis: Bei Dateipfaden auf einem Windows-Betriebssystem sind die rückwärtigen Schrägstriche (\) durch normale Schrägstriche zu ersetzen (/).

Die verwendbaren Platzhalter haben folgende Bedeutungen:

| Platzhalter | Bedeutung |
|-------------|---------------------------|
| %Y | Jahr (4-stellig) |
| %M | Monat (2-stellig) |
| %D | Tag im Monat (2-stellig) |
| %W | Woche im Jahr (2-stellig) |
| %h | Stunde (2-stellig) |
| %% | Prozentzeichen |

Die Zykluszeit für die Backup-Erstellung ergibt sich aus dem verwendeten Platzhalter, der die kleinste Intervalllänge besitzt. Falls eine Zykluszeit abgelaufen ist, und der CCU-Historian zu diesem Zeitpunkt nicht gestartet war, so wird nachträglich kein Backup erstellt.

Falls der angegebene Dateiname die Dateiendung **.zip** besitzt, so wird die Backup-Datei ZIP-komprimiert. Bei der Dateiendung **.gz** wird die Backup-Datei GZIP-komprimiert.

Die erstellte Backup-Datei ist textuell und beinhaltet alle SQL-Kommandos, um die komplette Datenbank wieder herzustellen. Die Wiederherstellung erfolgt über die Web-Bedienschnittstelle der Datenbank (s.a. Abschnitt 6.5) durch Ausführen des folgenden SQL-Kommandos:

```
RUNSCRIPT FROM 'Dateiname der Backup-Datei'
```

Falls die Backup-Datei ZIP- oder GZIP-komprimiert ist, so ist an das SQL-Kommando noch **COMPRESSION ZIP** bzw. **COMPRESSION GZIP** anzuhängen.

Beispiele

```
database.backup='db_%Y-%M-%D.zip'
```

Es wird jeden Tag um Mitternacht eine ZIP-komprimierte Backup-Datei erstellt. Die erstellten Dateien sind z.B.: db_2014-04-30.zip, db_2014-05-01.zip, db_2014-05-02.zip, usw.

```
database.backup='L:/Backups/%Y-w%W.gz'
```

Es wird am Beginn jeder Woche (jeden Montag um 00:00 Uhr) eine GZIP-komprimierte Backup-Datei erstellt. Die erstellten Dateien sind z.B.: L:/Backups/2013-w52.gz, L:/Backups/2014-w01.gz, L:/Backups/2014-w02.gz, usw.

4.5 Eigene Web-Seiten einbinden

Mit der Konfigurationsoption *webServer.menuLinks* können im linken Menü der Web-Seiten benutzerdefinierte Einträge hinzugefügt werden. Dazu müssen folgende Unteroptionen gesetzt werden:

| Optionsname | Standardwert | Beschreibung |
|---|----------------------------|---|
| webServer.menuLinks.link<Nr>.text | Option muss gesetzt werden | Text des Verweises |
| webServer.menuLinks.link<Nr>.address | Option muss gesetzt werden | Adresse des Verweises (Es können relative oder absolute Adressen, vom eigenen Web-Server oder von fremden Web-Servern angegeben werden.) |

Die Option dient primär dazu auf eigene Web-Seiten im Verzeichnis *webapp/custom* zu verweisen (s.a. Abschnitt 7.5).

Beispiel

```
webServer.menuLinks.link1.text='Bsp. Vorjahresvergleich'  
webServer.menuLinks.link1.address='/custom/example1.html'
```

Ergebnis



4.6 Firewall-Einstellungen

Für eine vollständige Funktion des CCU-Historians müssen einige Netzwerkverbindungen zwischen dem CCU-Historian-Rechner und der CCU freigeschaltet werden.

Auf dem CCU-Historian-Rechner müssen folgende Netzwerk-Ports erreichbar sein:

| Port-Nr. | Funktion | Zugehörige Konfigurationsoption |
|-------------|---|---------------------------------|
| 80 | Port für den eingebetteten Web-Server | webServer.port |
| 2011 | Netzwerk-Port für den XMLRPC-Server des CCU-Historians | devices.historianRpcPort |
| 8082 | Netzwerk-Port für die Web-Bedienschnittstelle der Datenbank | database.webPort |
| 9092 | (Optional) Netzwerk-Port für die TCP-Schnittstelle der Datenbank | database.tcpPort |
| 5435 | (Optional) Netzwerk-Port für die PostgreSQL-Schnittstelle der Datenbank | database.pgPort |

Auf der CCU müssen folgende Netzwerk-Ports erreichbar sein:

| Port-Nr. | Funktion |
|-------------|--|
| 2000 | Schnittstellenprozess BidCos-Wired (CCU1 oder CCU2 mit HMWLGW) |
| 2001 | Schnittstellenprozess BidCos-RF |
| 2002 | Schnittstellenprozess System (nur CCU1) |
| 2010 | Schnittstellenprozess HM-IP (nur CCU2) |
| 8181 | HM Skript Schnittstelle |
| 8701 | CUxD (wenn angeschlossen) |

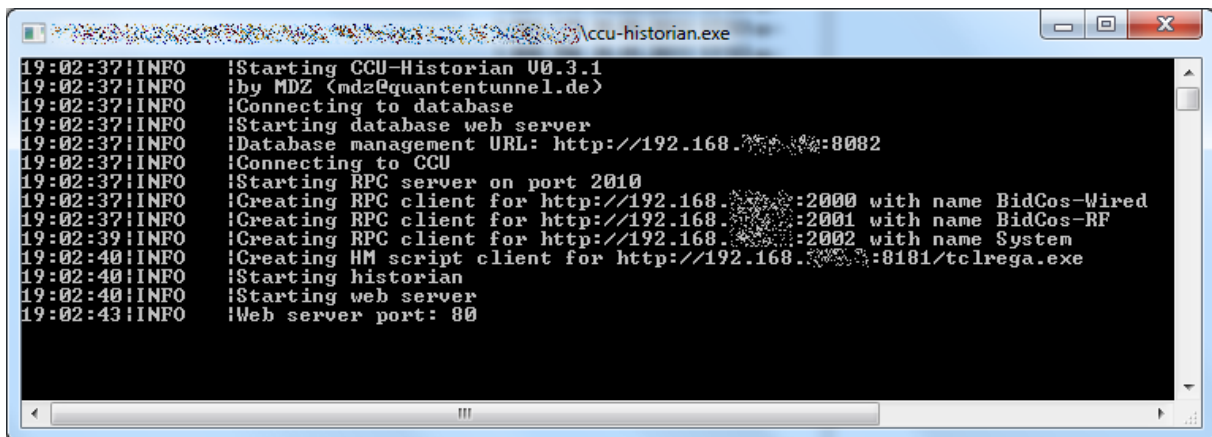
Die Firewall der CCU muss entsprechend konfiguriert sein, dass der CCU-Historian auch die CCU erreichen kann. Hinweis: Eine geänderte Firewall-Einstellung wird unter Umständen erst nach einem Neustart der CCU aktiv!

5 Start

Der Start des CCU-Historians erfolgt unter MS Windows durch Ausführen der Datei *ccu-historian.exe* (z.B. mit einem Doppelklick). Unter anderen Betriebssystemen muss folgender Befehl auf der Konsole eingegeben werden:

```
java -jar ccu-historian.jar
```

Es öffnet sich dann ein Konsolenfenster, das bei einer Standardkonfiguration folgenden Inhalt zeigt:



```
19:02:37!INFO      !Starting CCU-Historian V0.3.1
19:02:37!INFO      !by MDZ (mdz@quantentunnel.de)
19:02:37!INFO      !Connecting to database
19:02:37!INFO      !Starting database web server
19:02:37!INFO      !Database management URL: http://192.168.1.100:8082
19:02:37!INFO      !Connecting to CCU
19:02:37!INFO      !Starting RPC server on port 2010
19:02:37!INFO      !Creating RPC client for http://192.168.1.100:2000 with name BidCos-Wired
19:02:37!INFO      !Creating RPC client for http://192.168.1.100:2001 with name BidCos-RF
19:02:39!INFO      !Creating RPC client for http://192.168.1.100:2002 with name System
19:02:40!INFO      !Creating HM script client for http://192.168.1.100:8181/tclrega.exe
19:02:40!INFO      !Starting historian
19:02:40!INFO      !Starting web server
19:02:43!INFO      !Web server port: 80
```

Beim ersten Start wird automatisch eine Datenbank angelegt. Der Name der Datenbank wird durch die Konfigurationsoption *database.name* gesetzt, das Verzeichnis für die Datenbankdateien durch *database.dir* angegeben. Das beim Erstellen verwendete Passwort (Konfigurationsoption *database.password*) darf für den späteren Zugriff nicht verändert werden.

Je nach Konfiguration werden auch Log-Dateien erstellt, die bei Verwendung der Standardwerte für die Optionen *logSystem.fileName*, *logSystem.fileLimit* und *logSystem.fileCount* im Verzeichnis der Datei *ccu-historian.exe* liegen und folgende Namen erhalten: *ccu-historian-0.log*, *ccu-historian-1.log*, usw.

Für den Betrieb des CCU-Historians muss das Konsolenfenster geöffnet bleiben. Um den CCU-Historian zu beenden, das Konsolenfenster aktivieren, und dann die Tastenkombination Strg-C betätigen. Nur mit dieser Vorgehensweise meldet sich der CCU-Historian auch ordnungsgemäß bei der CCU ab.

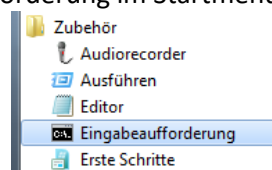
Für den automatischen Start des CCU-Historians beim Rechnerstart kann unter *MS Windows* die *Aufgabenplanung* vom Betriebssystem verwendet werden.

5.1 Startparameter

Der CCU-Historian unterstützt verschiedene Kommandozeilenparameter um bereits Einstellungen vor dem Lesen der Konfigurationsdatei zu setzen (z.B. der Pfad zur Konfigurationsdatei) und verschiedene Wartungsarbeiten durchzuführen (z.B. Sicherung/Wiederherstellung der Datenbank).

Die Liste der unterstützten Kommandozeilenparameter inklusive Beschreibung wird durch Aufruf des CCU-Historians in der Eingabeaufforderung mit dem Parameter **-help** ausgegeben.

Unter MS Windows ist die Eingabeaufforderung im Startmenü an folgender Stelle zu finden:



Mit dem Kommando **cd Installationsverzeichnis** muss als erstes in das Installationsverzeichnis des CCU-Historians gewechselt werden.

Danach ist unter MS Windows Folgendes einzugeben:

```
ccu-historian -help
```

Auf anderen Betriebssystemen muss der CCU-Historian mit **java** gestartet werden:

```
java -jar ccu-historian.jar -help
```

Die Ausgabe sieht wie folgt aus:

```
usage: ccu-historian [options]
options:
  -clean <date>          deletes all time series entries before the
                        specified date (format is DD.MM.YYYY or
                        YYYY-MM-DD) and then shuts down
  -compact               compacts the database and then shuts down
  -config <file>         changes the path to the configuration file
  -createscript <file>   dumps the database to a file as SQL script and then
                        shuts down
  -help                 prints this help message
  -loglevel <level>     sets the console log level (off, severe, warning,
                        info, fine, finer, finest) before the
                        configuration file is read
  -recalc               recalculates compressed data points and then shuts
                        down
  -runscript <file>     runs an SQL script from file and then shuts down
```

Liste aller Startparameter:

| Startparameter | Argument | Standardwert | Beschreibung |
|---------------------|----------------------|----------------------|---|
| clean | Datum | - | Löscht alle Zeitreiheneinträge vor dem angegebenen Datum (Datumsformat Tag.Monat.Jahr oder Jahr-Monat-Tag) und beendet sich dann. |
| compact | - | - | Kompaktiert die Datenbank. Dadurch wird ungenutzter Speicher in der Datenbank freigegeben. Dies sollte immer nach Benutzung der Startparameter <i>clean</i> oder <i>recalc</i> durchgeführt werden. |
| config | Konfigurations-datei | ccu-historian.config | Setzt den Pfad zur Konfigurationsdatei. |
| createscript | Skriptdatei | - | Der Datenbankinhalt wird als SQL-Skript in der angegebenen Datei gespeichert. Danach beendet sich der CCU-Historian. |
| help | - | - | Gibt eine Hilfe zu den Kommandozeilenoptionen aus. |
| loglevel | | info | Setzt den Meldungsfilter für Log-Meldungen auf der Konsole. Dieser Meldungsfilter ist bereits vor dem Lesen der Konfigurationsdatei aktiv. Dadurch können Fehler in der Konfigurationsdatei besser analysiert werden. |
| recalc | - | - | Komprimiert die Zeitreihen erneut (s.a. Abschnitt 2.1) und beendet sich dann. |

| | | | |
|------------------|-------------|---|--|
| runscript | Skriptdatei | - | Führt die angegebenen SQL-Kommandos in der Skript-Datei aus und beendet sich dann. |
|------------------|-------------|---|--|

6 Datenbank

6.1 Historientabellen für Gerätedatenpunkte

Das interne Datenmodell der HomeMatic CCU sieht in groben Zügen wie folgt aus: Jedes an der CCU angeschlossene Gerät besitzt eine eindeutige Serien-Nummer. Jedes Gerät stellt eine Anzahl an Kanälen zur Verfügung, die über eine Kanal-Nummer identifiziert werden. Jeder Kanal besitzt eine Anzahl an Datenpunkten, die über ihren Namen identifiziert werden. Um einen Datenpunkt eindeutig zu identifizieren wird also die Serien-Nummer des Geräts, die Kanal-Nummer und der Name des Datenpunktes benötigt.

Mit diesen Informationen legt der CCU-Historian eine Tabelle für jeden Datenpunkt an. In dieser wird dann die gesamte Historie des Datenpunktes gespeichert. Der Tabellename wird nach folgender Regel für numerische (Logikwert, Werteliste, Zahl, Alarm) Datenpunkte generiert:

D_*Schnittstellename_Kanalname_Datenpunkt*

Für Datenpunkte vom Typ Zeichenkette gilt folgende Regel:

C_*Schnittstellename_Kanalname_Datenpunkt*

Der Schnittstellename ist z.B. **BidCos_RF**, **BidCos_Wired** oder **System**. Alle nicht alphanumerischen Zeichen werden durch einen Unterstrich (**_**) ersetzt.

Eine Historientabelle besitzt folgenden Aufbau:

| Spalte | Name | Datentyp | Beschreibung |
|----------|-------|--|--|
| 1 | TS | TIMESTAMP | Zeitstempel |
| 2 | VALUE | DOUBLE (numerischer Datenpunkte) VARCHAR (Zeichenkettendatenpunkte) | Wert |
| 3 | STATE | INTEGER | Status: 1=Der Wert wurde durch ein Event von der CCU übermittelt. |

Die Tabelle zu einem Datenpunkt wird erst dann angelegt, wenn der erste Datensatz beim CCU-Historian eintrifft.

Alle Datenpunkte mit den Namen RAIN_COUNTER oder SUNSHINEDURATION werden als Zähler behandelt, die nur bis zu einem Maximalwert zählen und dann überlaufen. Der CCU-Historian bemerkt diese Überläufe und korrigiert automatisch alle nachfolgenden Werte.

6.2 Historientabellen für Systemvariablen

Jede Systemvariable wird von der Logikschicht der CCU über eine eindeutige Objekt-Identifikation (eine Zahl) referenziert. Diese ID ändert sich auch bei Umbenennung der Variable nicht. Der CCU-Historian nutzt diesen Umstand, um einen eindeutigen Tabellennamen im folgenden Format zu generieren: **D_Schnittstellename_SystemvariablenID_VALUE**

Für Systemvariablen vom Datentyp Zeichenkette gilt folgende Regel:

C_Schnittstellename_SystemvariablenID_VALUE

Der Schnittstellename ist z.B. **SysVar**.

Die angelegten Tabellen haben denselben Aufbau wie die Historientabellen der Gerätedatenpunkte (s.a. Abschnitt 6.1). Falls der Datentyp einer bestehenden Systemvariable zwischen numerisch (Logikwert, Werteliste, Zahl, Alarm) und Zeichenkette wechselt, so wird für diese Systemvariable eine zweite Tabelle angelegt, um den geänderten Datentyp speichern zu können. Das Feld **TABLE_NAME** in der Tabelle **DATA_POINTS** gibt immer den jeweils gültigen Tabellennamen an.

Die Werte der Systemvariablen werden zyklisch vom CCU-Historian abgefragt. Die Zykluszeit wird mit der Konfigurationsoption *historian.sysVarDataCycle* gesetzt. Wenn sich der Wert einer Systemvariablen zwischen zwei Abfragen ändert, so wird der tatsächliche Zeitpunkt der Änderung abgespeichert. Wenn sich der Wert mehrmals zwischen zwei Abfragen ändert, so kann nur die letzte Änderung ermittelt und abgespeichert werden. Eine Zykluszeit kleiner als 5000 Millisekunden sollte nicht eingestellt werden.

6.3 Tabelle DATA_POINTS mit Meta-Informationen

Für jeden Datenpunkt pflegt der CCU-Historian einen Eintrag in der Tabelle **DATA_POINTS**. Die Tabelle hat folgenden Aufbau:

| Spalte | Name | Datentyp | Beschreibung |
|--------|--------------|----------|---|
| 1 | DP_ID | INT | Eindeutige ID des Datenpunktes im CCU-Historian |
| 2 | TABLE_NAME | VARCHAR | Name der Historientabelle |
| 3 | STATE | INT | Reserviert |
| 4 | INTERFACE | VARCHAR | CCU-Schnittstelle des Gerätes (BidCos-RF, BidCos-Wired, System oder SysVar) |
| 5 | ADDRESS | VARCHAR | Seriennummer:Kanalnummer oder Systemvariablen-ID |
| 6 | IDENTIFIER | VARCHAR | Datenpunktbezeichner |
| 7 | DISPLAY_NAME | VARCHAR | Anzeigenname des Kanals/der Systemvariable in der Logikschicht der CCU |
| 8 | COMMENT | VARCHAR | Kommentar (wird noch nicht gesetzt) |
| 9 | PARAM_SET | VARCHAR | Parameter-Satz des Datenpunktes (z.B. VALUES für dynamische Werte) |
| 10 | TAB_ORDER | INT | Für die Beschreibung der Spalten 9 bis 17 wird auf die Dokumentation der XML-RPC-Schnittstelle verwiesen (Abschnitt 3.2). |
| 11 | MAXIMUM | DOUBLE | |
| 12 | UNIT | VARCHAR | Bei Systemvariablen werden die Felder 11, 12, 13, 15, 17 mit Informationen aus der Logikschicht der CCU gefüllt. |
| 13 | MINIMUM | DOUBLE | |
| 14 | CONTROL | VARCHAR | |
| 15 | OPERATIONS | INT | |

| | | |
|-----------|---------------|---------|
| 16 | FLAGS | INT |
| 17 | TYPE | VARCHAR |
| 18 | DEFAULT_VALUE | DOUBLE |

Wenn vom CCU-Historian Werte von einem unbekannten Gerätedatenpunkt empfangen werden, werden als erstes die Spalten 1 bis 6 und 9 in der Tabelle *DATA_POINTS* gefüllt und eine neue Historientabelle angelegt. Dadurch können die empfangenen Werte sofort in die Datenbank weggeschrieben werden. In einem separaten Arbeitszyklus (s.a. Konfigurationsoption *historian.metaCycle*) werden die Meta-Informationen in den restlichen Spalten nachgepflegt.

Die Meta-Informationen von Systemvariablen werden kontinuierlich abgeglichen. Wenn z.B. der Name einer Systemvariablen geändert wird, wird der neue Name beim nächsten Zyklus automatisch nachgepflegt. Bei Systemvariablen wird in der Spalte *INTERFACE* der Wert *SysVar* eingetragen.

Eventuelle weitere, hier nicht dokumentierte Tabellen werden vom CCU-Historian nur intern verwendet und können sich in neueren Versionen des CCU-Historians ändern oder nicht mehr vorhanden sein.

6.4 Beispiele für SQL-Anfragen

Alle am 22.1.2011 gesendeten Helligkeitswerte des Bewegungsmelders mit der Serien-Nummer GEQ0126000 sortiert nach dem Zeitstempel abfragen:

```
SELECT * FROM V_GEQ0126000_1_BRIGHTNESS WHERE TS>='2011-01-22' AND
TS<'2011-01-23' ORDER BY TS
```

Die minimale und maximale Helligkeit an diesem Tag abfragen:

```
SELECT MIN(VALUE), MAX(VALUE) FROM V_GEQ0126000_1_BRIGHTNESS WHERE
TS>='2011-01-22' AND TS<'2011-01-23'
```

Anzahl der Kommunikationsstörungen an diesem Tag mit diesem Gerät:

```
SELECT COUNT(*) FROM V_GEQ0126000_0_UNREACH WHERE VALUE<>0.0 AND
TS>='2011-01-22' AND TS<'2011-01-23'
```

Die zuletzt vom Gerät übermittelte (also aktuelle) Helligkeit:

```
SELECT * FROM V_GEQ0126000_1_BRIGHTNESS WHERE TS=(SELECT MAX(TS)
FROM V_GEQ0126000_1_BRIGHTNESS)
```

Den Helligkeitsverlauf vom 22.1.2011 als CSV-Datei exportieren. Die Datei wird im Verzeichnis von der Datei ccu-historian.exe erstellt. Verwendete Hochkommas im SQL-Ausdruck müssen verdoppelt werden:

```
CALL CSVWRITE ('Helligkeitsverlauf.csv', 'SELECT * FROM
V_GEQ0126000_1_BRIGHTNESS WHERE TS>=''2011-01-22'' AND TS<=''2011-01-
23'' ORDER BY TS')
```

Anmerkung: Über die Web-Schnittstelle des CCU-Historians (s.a. Abschnitt 7) können auch CSV-Daten exportiert werden.

Eine Auflistung aller vorhandenen Historien-Tabellen liefert folgender SQL-Ausdruck:

```
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME  
LIKE 'V\_%'
```

Die Existenz einer Tabelle kann mit folgender Abfrage festgestellt werden. Sie liefert einen Wert ungleich 0, falls die angegebene Tabelle `V_GEQ0126000_1_BRIGHTNESS` existiert:

```
SELECT COUNT(*) FROM INFORMATION_SCHEMA.TABLES WHERE  
TABLE_NAME='V_GEQ0126000_1_BRIGHTNESS'
```

6.5 Web-Bedienschnittstelle

Auf die Web-Bedienschnittstelle der Datenbank kann lokal mit der Web-Adresse <http://localhost:8082> zugegriffen werden. Dazu muss die Konfigurationsoption `database.webEnable=true` gesetzt sein. Falls der Zugriff auch von anderen Rechnern aus erlaubt sein soll, so muss die Option `database.webAllowOthers=true` zusätzlich gesetzt werden.

Für den Zugriff auf die Datenbank muss als erstes eine Anmeldung erfolgen. Die Angaben für *JDBC URL*, *Benutzername* und *Passwort* hängen von den Konfigurationsoptionen `database.dir`, `database.name`, `database.user` und `database.password` ab. Wenn die Standardwerte für diese Optionen verwendet werden, ist der Anmelde-Dialog wie folgt auszufüllen:

The screenshot shows the 'Login' dialog box of the CCU-Historian web interface. At the top, there is a language dropdown menu set to 'Deutsch' and three links: 'Optionen', 'Tools', and 'Hilfe'. The dialog is divided into two sections. The top section, titled 'Gespeicherte Einstellung:', shows a dropdown menu with 'Generic H2 (Embedded)' selected. Below it, the 'Einstellungs-Name:' field also contains 'Generic H2 (Embedded)', with 'Speichern' and 'Entfernen' buttons to its right. The bottom section contains fields for 'Datenbank-Treiber Klasse:' (filled with 'org.h2.Driver'), 'JDBC URL:' (filled with 'jdbc:h2:./data/history'), 'Benutzername:' (filled with 'sa'), and 'Passwort:' (filled with a masked password). At the bottom of this section are 'Verbinden' and 'Verbindung testen' buttons.

Die Web-Oberfläche der Datenbank sieht wie folgt aus: Links wird eine Liste der vorhandenen Tabellen angezeigt, oben rechts werden SQL-Ausdrücke eingegeben und unten rechts werden die Ergebnisse von ausgeführten SQL-Ausdrücken angezeigt.

The screenshot shows a web-based database interface. On the left, there is a tree view of tables under the path 'jdbc:h2:data/history'. The tables listed include various V_GEQ0127 and V_GEQ023 tables with different states like _0_STICKY_UNREACH, _0_UNREACH, _1_LEVEL, _1_WORKING, _1_BRIGHTNESS, _1_ERROR, _1_INSTALL_TEST, and _1_MOTION. At the top right, there is a text input field for the SQL command, containing 'SELECT * FROM V_GEQ0127_1_BRIGHTNESS'. Below this, there is a button 'Ausführen (Strg+Enter)' and a 'Leeren' button. The bottom right section displays the results of the query in a table format.

| TS | VALUE | STATE |
|-------------------------|-------|-------|
| 2011-01-22 08:44:47.628 | 33.0 | 1 |
| 2011-01-22 08:50:49.445 | 33.0 | 1 |
| 2011-01-22 08:51:05.213 | 33.0 | 1 |
| 2011-01-22 08:56:08.881 | 33.0 | 1 |
| 2011-01-22 08:56:34.355 | 33.0 | 1 |
| 2011-01-22 09:00:40.383 | 33.0 | 1 |
| 2011-01-22 09:06:25.938 | 33.0 | 1 |
| 2011-01-22 09:11:42.898 | 33.0 | 1 |
| 2011-01-22 09:16:30.88 | 33.0 | 1 |
| 2011-01-22 09:20:49.879 | 33.0 | 1 |

6.6 ODBC-Schnittstelle

Die eingebettete Datenbank unterstützt das PostgreSQL-Netzwerkprotokoll. Dadurch kann der PostgreSQL-ODBC-Treiber auch für den Zugriff auf die CCU-Historian-Datenbank verwendet werden. In der Konfiguration muss die Option `database.pgEnable=true` gesetzt sein. Falls der Zugriff auch von anderen Rechnern aus erlaubt sein soll, so muss die Option `database.pgAllowOthers=true` zusätzlich gesetzt werden.

Der PostgreSQL-ODBC-Treiber ist unter folgender URL zu finden:

<http://www.postgresql.org/ftp/odbc/versions>

Fertige Installationspakete sind im Ordner *MSI* zu finden.

Hinweis: Unter einem 64-Bit Windows gibt es sowohl 64-Bit als auch 32-Bit-Treiber. Deswegen gibt es auch einen 64-Bit *ODBC-Datenquellen-Administrator*, der über das normale Startmenü erreichbar ist, und zusätzlich eine 32-Bit-Version, die unter `c:\Windows\SysWOW64\odbcad32.exe` zu finden ist. Die benötigte Version von Treiber und Administrator hängt von der Applikation ab, die den Treiber verwenden will.

Nach Installation des Treibers muss im *ODBC-Datenquellen-Administrator* eine neue Datenquelle unter Verwendung des PostgreSQL-Treibers angelegt werden. Die Einstellungen sind wie folgt zu setzen:

Das Feld *Database* setzt den Dateinamen (ohne Endung) der Datenbank. Dieser muss eventuell als absoluter Pfad angegeben werden. Zum Beispiel muss er für eine CCU-Historian-Installation unter Linux im Verzeichnis `/opt/ccuhistorian` wie folgt lauten: `/opt/ccu-historian/data/history`

7 Web-Schnittstelle

Über eine Web-Schnittstelle kann der CCU-Historian Daten exportieren oder darstellen. Dazu ist ein Web-Server im CCU-Historian eingebettet. Zusätzlich zu den HTML-Seiten des CCU-Historians kann der Anwender eigene HTML-Seiten über den Web-Server anbieten. Dadurch kann der Anwender beliebige Übersichtsseiten mit aktuellen Messwerten und/oder Trend-Diagrammen erstellen.

7.1 Standard Web-Seiten

Der CCU-Historian bietet einen Standardsatz an Web-Seiten. Auf diese kann mit der Adresse <http://localhost> zugegriffen werden. Wenn die Konfigurationsoption `webServer.port` nicht den Standardwert 80 besitzt, muss an die Adresse noch die konfigurierte Portnummer angehängt werden (z.B. <http://localhost:81>).

Falls für die Web-Seiten ein Passwort gesetzt worden ist, muss beim ersten Zugriff das Passwort eingegeben werden:

(Das Passwort wird auf der Web-Seite *Konfiguration* gesetzt oder gelöscht.)

7.1.1 Hauptseite / Übersicht Datenpunkte

Auf der Hauptseite des CCU-Historians werden eine Liste weiterer Web-Seiten und Anwendungen (z.B. CCU, Datenbank, Konfiguration) und eine Liste aller Datenpunkte angezeigt:

| CCU-Historian | | Übersicht Datenpunkte | | | | | | | |
|--|---------------|------------------------|----------------|--------------|----------------|---------|---------------------|-------------------------|--------------------------|
| Menü | Schnittstelle | Kanal | Parameter | Historian-ID | Aktueller Wert | Einheit | Zeitstempel | Details | Ausw. |
| Übersicht Datenpunkte | BidCos-RF | Alarmgeber außen:0 | STICKY_UNREACH | 1 | 1,00 | | 23.07.2011 23:13:10 | Details | <input type="checkbox"/> |
| CCU | BidCos-RF | Alarmgeber außen:0 | UNREACH | 2 | 1,00 | | 23.07.2011 23:13:10 | Details | <input type="checkbox"/> |
| Datenbank | BidCos-RF | Außenbeleuchtung Gr. 1 | STATE | 39 | 0,00 | | 24.07.2011 00:00:04 | Details | <input type="checkbox"/> |
| Konfiguration | BidCos-RF | Außenbeleuchtung Gr. 1 | WORKING | 40 | 0,00 | | 24.07.2011 00:00:04 | Details | <input type="checkbox"/> |
| Benutzer <input type="button" value="Abmelden"/> | BidCos-RF | Außenbeleuchtung Gr. 2 | STATE | 42 | 0,00 | | 24.07.2011 00:00:04 | Details | <input type="checkbox"/> |
| | BidCos-RF | Außenbeleuchtung Gr. 2 | WORKING | 43 | 0,00 | | 24.07.2011 00:00:04 | Details | <input type="checkbox"/> |
| | BidCos-RF | Bewegung Flur EG | BRIGHTNESS | 10 | 43,00 | | 24.07.2011 10:48:15 | Details | <input type="checkbox"/> |
| | BidCos-RF | Bewegung Flur EG | ERROR | 11 | 0,00 | | 24.07.2011 10:48:15 | Details | <input type="checkbox"/> |
| | BidCos-RF | Bewegung Flur EG | INSTALL_TEST | 18 | 1,00 | | 24.07.2011 10:32:58 | Details | <input type="checkbox"/> |
| | BidCos-RF | Bewegung Flur EG | MOTION | 16 | 0,00 | | 24.07.2011 10:37:39 | Details | <input type="checkbox"/> |

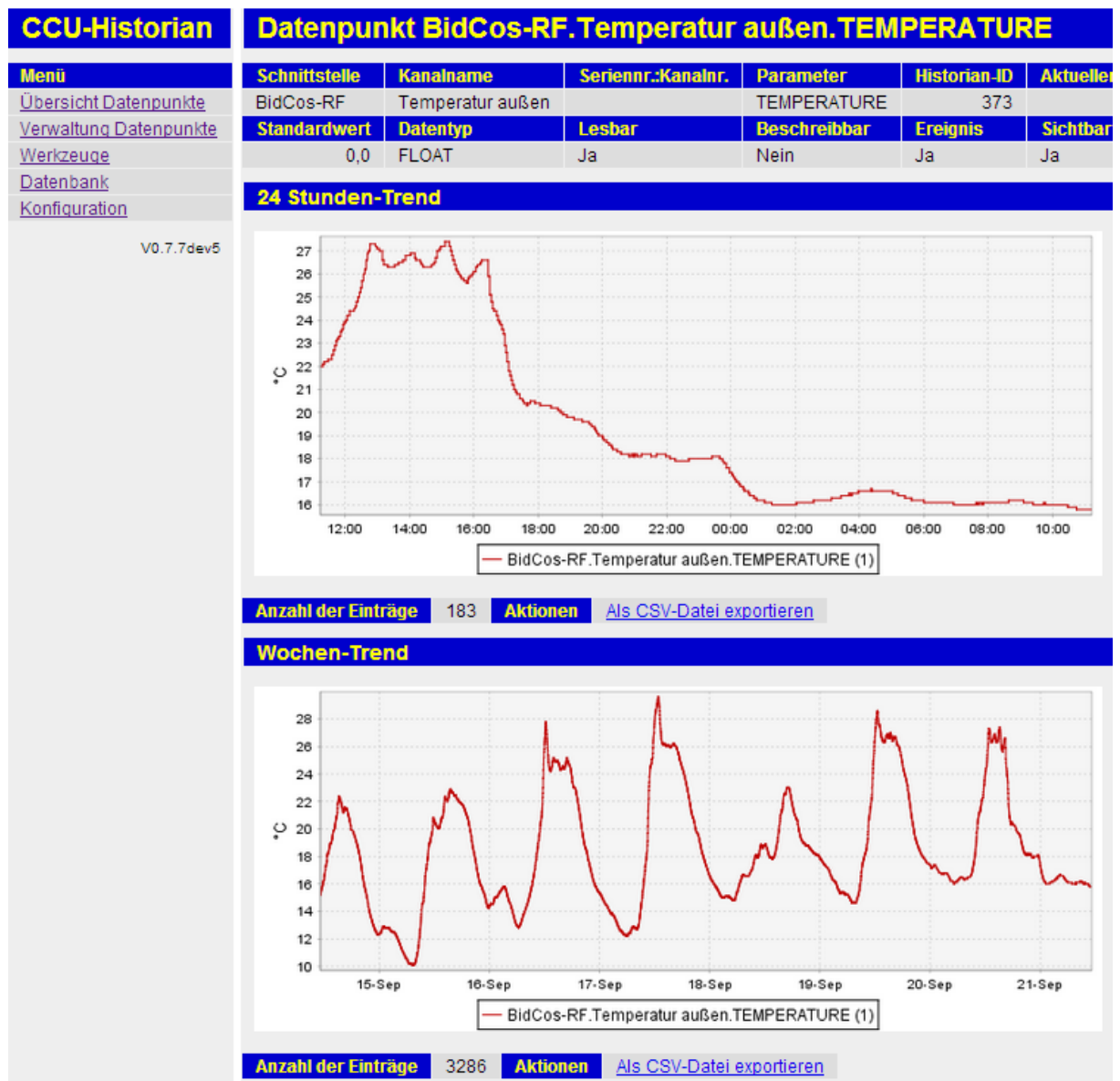
Alle bekannten Datenpunkte werden mit der Historian-ID, dem Namen und dem aktuellen Wert aufgelistet. Die Historian-ID identifiziert einen Datenpunkt eindeutig (Tabelle *DATA_POINTS*, Feld *DP_ID*, s.a. Abschnitt 6.3), und wird für den Export von Zeitreihen und der Trend-Generierung benötigt. Die letzte Übermittlung des Wertes von der CCU an den Historian ist der Spalte Zeitstempel zu entnehmen.

7.1.2 Detail-Seite zu einem Datenpunkt

Zu jedem Datenpunkt können weitere Details angezeigt werden. Dazu existiert der Verweis *Details* in der letzten Spalte. Auf der Detail-Seite zu einem Datenpunkt werden weitere Meta-Informationen (wie Seriennummer, Kanalnummer, Minimum, Maximum, usw.) und zwei Trends für numerische Datenpunkte oder eine Werteliste für Zeichenkettendatenpunkte angezeigt. Ein Trend zeigt die Entwicklung des Wertes innerhalb der letzten 24 Stunden an, ein zweiter Trend innerhalb der letzten Woche. Die Werteliste für Zeichenkettendatenpunkte enthält alle gespeicherten Werte der letzten Woche.

Unterhalb jeder Trend-Darstellung und der Werteliste wird ein Verweis *Als CSV-Datei exportieren* angezeigt. Mit diesem Verweis kann die im Trend oder in der Werteliste dargestellte Zeitreihe im CSV-Format auf dem lokalen Rechner abgespeichert werden.

Die Detail-Seite zu einem numerischen Datenpunkt sieht zum Beispiel wie folgt aus:



Für Datenpunkte vom Typ Zeichenkette wird eine tabellarische Darstellung verwendet. Es werden die Einträge der letzten 7 Tage angezeigt.

Datenpunkt SysVar.Testsystemvariable Str.VALUE

| Schnittstelle | Kanalname | Seriennr.:Kanalnr. | Parameter | Historian-ID | Minimum | Ma |
|---------------|------------------------|--------------------|-----------|--------------|---------|----|
| SysVar | Testsystemvariable Str | 8353 | VALUE | 12 | | |

Werte der letzten Woche

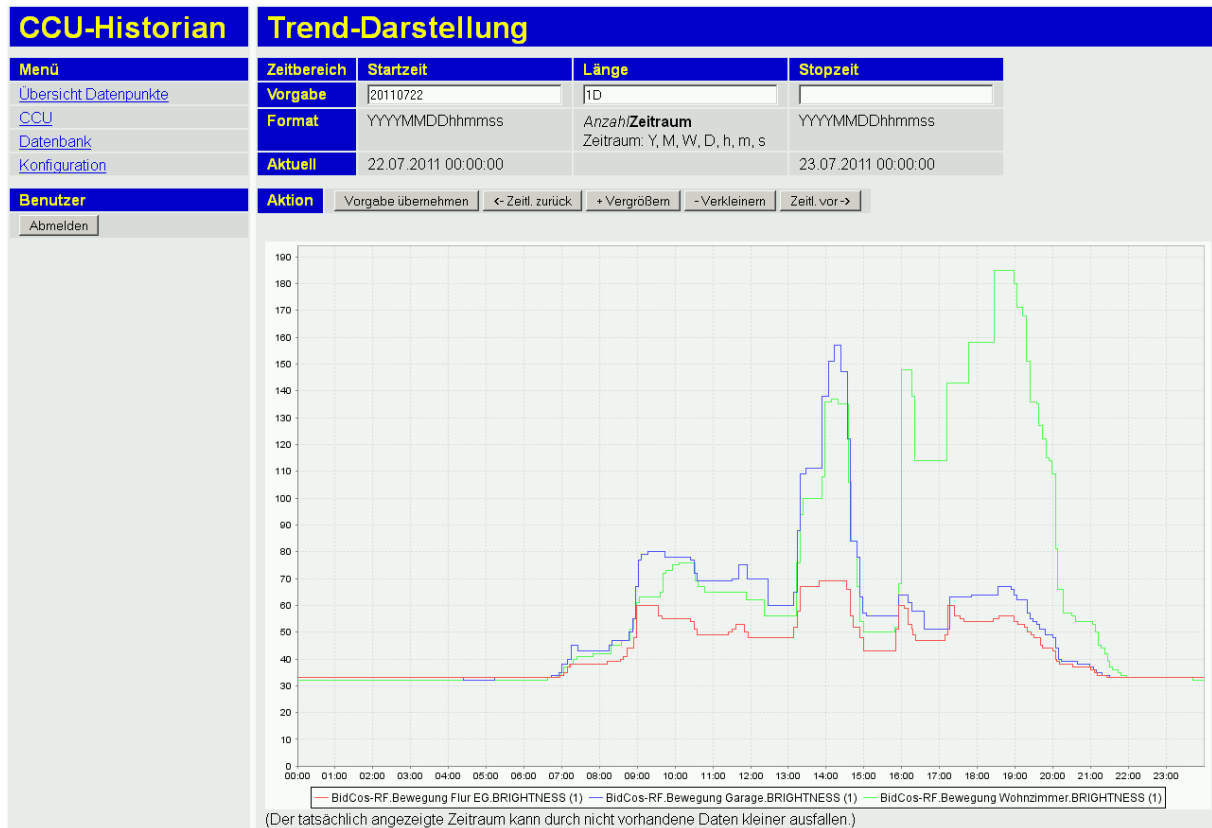
| Zeitstempel | Wert | Status |
|---------------------|--------|--------|
| 26.10.2011 18:07:29 | yxcvbn | 1 |
| 26.10.2011 18:06:56 | asdfgh | 1 |
| 26.10.2011 18:06:13 | qwertz | 1 |
| 26.10.2011 11:31:51 | ??? | 1 |

Als CSV-Datei exportieren

7.1.3 Mehrere Datenpunkte als Trend darstellen

Auf der Übersichtsseite der Datenpunkte können durch Setzen von Häkchen in der Spalte *Ausw.* beliebige Datenpunkte ausgewählt werden. Die selektierten Datenpunkte werden in einem gemeinsamen Trend-Diagramm dargestellt, wenn die Schaltfläche *Trend-Darstellung* unterhalb der Datenpunktliste betätigt wird.

Auf der Web-Seite zur Trend-Darstellung kann im oberen Bereich der Zeitbereich für das Trend-Diagramm verändert werden. Im unteren Bereich erfolgt die Trend-Darstellung der Datenpunkte:



Das Format für die Vorgabe des Zeitbereichs ist im Abschnitt 7.2 (siehe Parameter **b**, **e**, **d**) dokumentiert. Für die einfache Navigation existieren weitere Schaltflächen.

7.1.4 Verwaltung Datenpunkte

Datenpunktspezifische Einstellungen können auf der Verwaltungsseite der Datenpunkte vorgenommen werden. Zudem wird die archivierte Anzahl an Einträgen angezeigt:

| Verwaltung Datenpunkte | | | | | | | | | |
|------------------------|------------------------------|----------------------|-----|----------|--------------------------|-------------------------------------|------------|-----------|--|
| Aktion | | Änderungen speichern | | | | | | | |
| Schnittstelle | Kanal | Parameter | ID | Einträge | Inaktiv | Versteckt | Vorverarb. | Parameter | |
| [Alle] | | [Alle] | | | | | | | |
| BidCos-RF | Alarmanlage ein-/ausschalten | PRESS_LONG | 133 | 5 | <input type="checkbox"/> | <input type="checkbox"/> | - | | |
| BidCos-RF | Alarmanlage ein-/ausschalten | PRESS_SHORT | 378 | 1 | <input type="checkbox"/> | <input type="checkbox"/> | - | | |
| BidCos-RF | Außenbeleuchtung Gr. 1 | STATE | 100 | 1823 | <input type="checkbox"/> | <input type="checkbox"/> | - | | |
| BidCos-RF | Außenbeleuchtung Gr. 1 | WORKING | 101 | 1900 | <input type="checkbox"/> | <input checked="" type="checkbox"/> | - | | |
| BidCos-RF | Außenbeleuchtung Gr. 2 | STATE | 103 | 1922 | <input type="checkbox"/> | <input type="checkbox"/> | - | | |
| BidCos-RF | Außenbeleuchtung Gr. 2 | WORKING | 104 | 1893 | <input type="checkbox"/> | <input checked="" type="checkbox"/> | - | | |
| BidCos-RF | Außenbeleuchtung:0 | STICKY_UNREACH | 285 | 56 | <input type="checkbox"/> | <input checked="" type="checkbox"/> | - | | |
| BidCos-RF | Außenbeleuchtung:0 | UNREACH | 284 | 51 | <input type="checkbox"/> | <input checked="" type="checkbox"/> | - | | |
| BidCos-RF | Bewegung Flur EG | BRIGHTNESS | 27 | 222052 | <input type="checkbox"/> | <input type="checkbox"/> | Delta K. | 0,1 | |
| BidCos-RF | Bewegung Flur EG | ERROR | 88 | 182813 | <input type="checkbox"/> | <input checked="" type="checkbox"/> | - | | |
| BidCos-RF | Bewegung Flur EG | INSTALL_TEST | 29 | 39738 | <input type="checkbox"/> | <input checked="" type="checkbox"/> | - | | |
| BidCos-RF | Bewegung Flur EG | MOTION | 28 | 71010 | <input type="checkbox"/> | <input type="checkbox"/> | - | | |
| BidCos-RF | Bewegung Garage | BRIGHTNESS | 33 | 166832 | <input type="checkbox"/> | <input type="checkbox"/> | Delta K. | 0,1 | |

Zu jedem Datenpunkt können die Eigenschaften **Inaktiv** und **Versteckt** angepasst werden. Versteckte Datenpunkte werden in der Datenpunktübersicht nicht angezeigt. Inaktive Datenpunkte werden nicht archiviert.

| Inaktiv | Versteckt |
|--------------------------|-------------------------------------|
| <input type="checkbox"/> | <input type="checkbox"/> |
| <input type="checkbox"/> | <input type="checkbox"/> |
| <input type="checkbox"/> | <input type="checkbox"/> |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> |

Eine Datenvorverarbeitung kann in den Spalten **Vorverarb.** und **Parameter** konfiguriert werden. Die Einstellmöglichkeiten und Funktionalitäten sind im Abschnitt 2.1 beschrieben.

| Vorverarb. | Parameter |
|------------|-----------|
| - | |
| Delta K. | 0,01 |
| Delta K. | 0,1 |
| Delta K. | 0,1 |

7.1.5 Werkzeuge

Auf der Werkzeugseite werden verschiedene Funktionen bereitgestellt, um die Historien von Datenpunkten zu leeren oder bei einem Gerätewechsel zu kopieren.

| CCU-Historian | | Werkzeuge | | | |
|--|--|--|----------------------|--|---|
| Menü Übersicht Datenpunkte Verwaltung Datenpunkte Werkzeuge CCU Datenbank Konfiguration | | Achtung: Mit diesen Funktionen können Daten unwiderruflich gelöscht werden! Es sollte immer vorab eine Sicherungskopie der Datenbank erstellt werden (siehe Handbuch). | | | |
| Datenpunkt inkl. Historie löschen | | ID: | <input type="text"/> | <input type="button" value="Ausführen"/> | Falls der Datenpunkt in der CCU weiterhin vorhanden ist, wird er vom CCU-Historian erneut erkannt und angelegt. |
| Historie löschen | | ID: | <input type="text"/> | <input type="button" value="Ausführen"/> | Die komplette Historie des angegebenen Datenpunktes wird gelöscht. Falls der Datenpunkt nicht auf inaktiv gesetzt ist, wird die Historie mit neuen Werten erneut gefüllt. |
| Historie kopieren | | Quell-ID: | <input type="text"/> | <input type="button" value="Ausführen"/> | Die komplette Historie des angegebenen Datenpunktes wird in die Historie eines zweiten Datenpunktes eingefügt. Dies ist beispielsweise bei einem Gerätetausch sinnvoll. |
| | | Ziel-ID: | <input type="text"/> | <input type="button" value="Ausführen"/> | |
| Benutzer <input type="button" value="Abmelden"/> | | | | | |

7.1.6 Konfiguration

Auf der Konfigurationsseite können verschiedene Einstellungen bezüglich der Web-Applikation vorgenommen werden. Momentan sind nur Optionen zum Passwortschutz vorhanden:

| CCU-Historian | | Konfiguration | |
|---|-----------------------|--|--|
| Menü | Passwortschutz | <input type="checkbox"/> Passwort entfernen | Durch das Setzen dieser Option wird der Passwortschutz entfernt. |
| Übersicht Datenpunkte | | <input type="text"/> | Das Zugangspasswort für die Web-Seiten setzen oder ändern. Bei einem leeren Feld wird das Passwort nicht geändert. |
| CCU | | <input type="text"/> | Passwortwiederholung zur Überprüfung der Eingabe. |
| Datenbank | | | |
| Konfiguration | | | |
| Benutzer | | <input type="button" value="Änderungen übernehmen"/> | |
| <input type="button" value="Abmelden"/> | | | |

7.2 JSON-RPC Schnittstelle

Für die bessere Anbindung von [AJAX](#)-Anwendungen besitzt der CCU-Historian eine [JSON-RPC](#)-Schnittstelle.

Es gibt mehrere Möglichkeiten die Parameter für einen JSON-RPC-Aufruf zu übertragen. Sie sind weiter unten beschrieben. Die Antwort ist aber immer ein [JSON](#)-konformes Textdokument, das leicht mit JavaScript weiter verarbeitet werden kann.

Über die JSON-RPC Schnittstelle können die Eigenschaften, die archivierten und die aktuellen Werte aller Datenpunkte abgefragt werden. Falls die Konfigurationsoption *devices.device<G-Nr>.writeAccess = true* gesetzt ist, können auch die aktuellen Werte der Datenpunkte in den Geräten bzw. der CCU verändert werden.

Eine JSON-RPC Anfrage besteht aus den drei Komponenten Methodenname, Parameterliste und der Identifikation der Anfrage. Der Methodenname bestimmt die aufzurufende Methode. Diese wird dann mit den Parametern aus der Parameterliste aufgerufen. Das Ergebnis des Funktionsaufrufes oder ein eventueller Fehler werden dann zusammen mit der Identifikation zum Aufrufer zurück geschickt.

Als Identifikation kann ein beliebiger JSON-Datentyp übergeben werden. Sie ist optional und kann daher auch komplett entfallen.

Alle Zeitstempel werden generell als Zahl, die die Millisekunden seit dem 1.1.1970 (Zeitzone UTC) repräsentiert, übertragen. In JavaScript kann mit **new Date(Millisekunden)** ein zugehöriges Date-Objekt einfach erzeugt werden. Von einem vorhandenen Date-Objekt kann der Wert durch Aufrufen von *date.getTime()* ermittelt werden.

Ein Aufruf sieht z.B. wie folgt aus. Es werden die Eigenschaften des Datenpunktes mit der Historian-ID 39 über ein HTTP GET angefragt:

<http://localhost/query/jsonrpc.gy?m=getDataPoint&p1=39&i=123>

Die vom CCU-Historian generierte Antwort ist dann z.B.:

```
{
```

```

    "id": "123",
    "result": {
      "interfaceHmSysVar": false,
      "managementFlags": 0,
      "id": {
        "interfaceId": "BidCos-RF",
        "address": "HEQXXXXXX:1",
        "identifizier": "LEVEL"
      },
      "interfaceHmDevice": true,
      "historyHidden": false,
      "attributes": {
        "displayName": "Rollladen Terrasse",
        "comment": null,
        "paramSet": "VALUES",
        "tabOrder": 0,
        "maximum": 1.0,
        "unit": "100%",
        "minimum": 0.0,
        "control": "BLIND.LEVEL",
        "operations": 7,
        "flags": 1,
        "type": "FLOAT",
        "defaultValue": 0.0
      },
      "historyTableName": "V_HEQXXXXXX_1_LEVEL",
      "idx": 39,
      "historyDisabled": false,
      "interfaceType": 0
    }
  }
}

```

7.2.1 Anfragearten

Insgesamt werden drei Anfragearten unterstützt. Die Beispiele zu den Anfragearten rufen immer dieselbe Methode **echo** mit dem Parametern **"Hallo Welt!"** und **0.5** und der Identifikation **123** auf. Falls die Konfigurationsoption `webServer.apiKeys` gesetzt ist, muss eins von den dort angegebenen Schlüsselwörtern mit dem Parameter **k** in der aufzurufenden Adresse übergeben werden.

- **HTTP POST mit einem JSON-Dokument (Content-Type: application/json)**

Aufzurufende Adresse: **`http://localhost/query/jsonrpc.gy[?k=Schlüsselwort]`**

Es wird ein JSON-Dokument mit einem JSON-Objekt übertragen. Das JSON-Objekt besitzt die drei Komponenten **method** für den Methodennamen, **params** ein JSON-Array für die Parameterliste und **id** für die Identifikation.

Beispiel:

```

{
  "id": 123,
  "method": "echo",
  "params": ["Hallo Welt!", 0.5]
}

```

Nach der [JSON-RPC](#)-Spezifikation ist dies die einzig zulässige Aufrufart. Einige Methodenaufrufe mit komplexen Parametern können nur oder am besten in dieser Aufrufform realisiert werden.

- **HTTP GET oder (Formular-)POST mit den Parametern *m*, *p1*, *p2*, usw. und *i***

Die Aufrufparameter können gänzlich ohne JSON-Darstellung in der URL einer HTTP GET-Anfrage kodiert werden. Es muss der Parameter **m** für den Methodennamen, die Parameter **p1**, **p2**, usw. für die Parameterliste, und optional **i** für die Identifikation gesetzt werden. Ob und wie viele Parameter **p1**, **p2**, usw. angegeben werden müssen, hängt von der aufzurufenden Methode ab.

Beispiel:

`http://localhost/query/jsonrpc.gy?m=echo&p1=Hallo+Welt!&p2=0.5&i=123[&k=Schlüsselwort]`

Analog können die Parameter auch über ein HTTP POST als Formular-Parameter kodiert (Content-Type: application/x-www-form-urlencoded) übermittelt werden.

- **HTTP GET oder (Formular-)POST mit den Parameter *j***

Ein JSON-Dokument mit einer Anfrage kann auch direkt in der URL einer HTTP GET-Anfrage kodiert werden. Der Parameter **j** muss dann mit dem Inhalt versehen werden.

Beispiel:

`http://localhost/query/jsonrpc.gy?j={"id":123,"method":"echo","params":["Hallo+Welt!"],0.5}[&k=Schlüsselwort]`

Analog kann der Parameter auch über ein HTTP POST als Formular-Parameter kodiert (Content-Type application/x-www-form-urlencoded) übermittelt werden.

7.2.2 Rückgaben

Die Rückgabe ist immer ein JSON-Dokument (Content-Type: application/json). Das zurückgegebene JSON-Objekt enthält entweder die Komponenten **id** und **result**, oder im Fehlerfall die Komponenten **id** und **error**. **result** enthält das Ergebnis des Methodenaufrufs, **error** besteht aus den Komponenten **code** (numerische Fehlerkennung) und **message** (Klartextfehlermeldung). Falls keine Identifikation im Aufruf angegeben wurde, hat die Komponente **id** den Wert **null**.

Beispiel für eine erfolgreiche Ausführung einer Methode (siehe Aufrufbeispiele oben):

```
{
  "id":123,
  "result":["Hallo Welt!",0.5]
}
```

Beispiel für eine fehlgeschlagene Ausführung einer Methode:

```
{
  "id":123,
  "error":{
    "code":-32601,
    "message":"Method not found"
  }
}
```

```
}  
}
```

7.2.3 Methoden

- **echo:** Mit der Anfrage antworten.

Diese Methode dient zum Testen von JSON-RPC-Clients. Die Parameterliste wird unverändert zurückgegeben.

Beispielanfrage:

```
{  
  "id": 456,  
  "method": "echo",  
  "params": [  
    "String",  
    123  
  ]  
}
```

Beispielantwort:

```
{  
  "id": 456,  
  "result": [  
    "String",  
    123  
  ]  
}
```

- **getDataPoint:** Die Eigenschaften von Datenpunkten ermitteln.

Diese Methode gibt die Eigenschaften eines Datenpunktes zurück, wenn ihr die Historian-ID eines Datenpunktes als Parameter übergeben wird. Alternativ kann auch der ISE-Name (z.B. "BidCos-RF.ABC0123456:2.STATE") eines Datenpunktes angegeben werden. Es können die Eigenschaften mehrerer Datenpunkte angefragt werden, wenn ein JSON-Array an Historian-IDs übergeben wird. Es werden die Eigenschaften aller Datenpunkte zurückgegeben, wenn die Methode ohne Parameter aufgerufen wird.

Beispielanfrage:

```
{  
  "method": "getDataPoint",  
  "params": [  
    "1"  
  ],  
  "id": "123"  
}
```

Beispielantwort:

```
{  
  "id": "123",  
  "result": {  
    "interfaceHmSysVar": false,  
    ...  
  }  
}
```



```

        "managementFlags":0,
        "id":{
            "interfaceId":"BidCos-RF",
            "address":"HEQXXXXXX:1",
            "identifizier":"LEVEL"
        },
        "interfaceHmDevice":true,
        "historyHidden":false,
        "attributes":{
            "displayName":"Rollladen Terrasse",
            "comment":null,
            "paramSet":"VALUES",
            "tabOrder":0,
            "maximum":1.0,
            "unit":"100%",
            "minimum":0.0,
            "control":"BLIND.LEVEL",
            "operations":7,
            "flags":1,
            "type":"FLOAT",
            "defaultValue":0.0
        },
        "historyTableName":"V_HEQXXXXXX_1_LEVEL",
        "idx":39,
        "historyDisabled":false,
        "interfaceType":0
    }
}

```

- **getTimeSeries:** Eine Zeitreihe von einem Datenpunkte abfragen.

Als Parameter werden die Historian-ID, die Anfangszeit und die Endzeit des abzufragenden Zeitbereichs übergeben. Alternativ kann auch der ISE-Name (z.B. "BidCos-RF.ABC0123456:2.STATE") eines Datenpunktes anstatt der Historian-ID angegeben werden. Falls keine Einträge zur Anfangszeit oder Endzeit existieren, so werden sie zusätzlich generiert.

Beispielanfrage:

```

{
    "id": 123,
    "method": "getTimeSeries",
    "params": [
        14,
        1311414600000,
        1311414900000
    ]
}

```

entspricht 2011-07-23 11:50:00

entspricht 2011-07-23 11:55:00

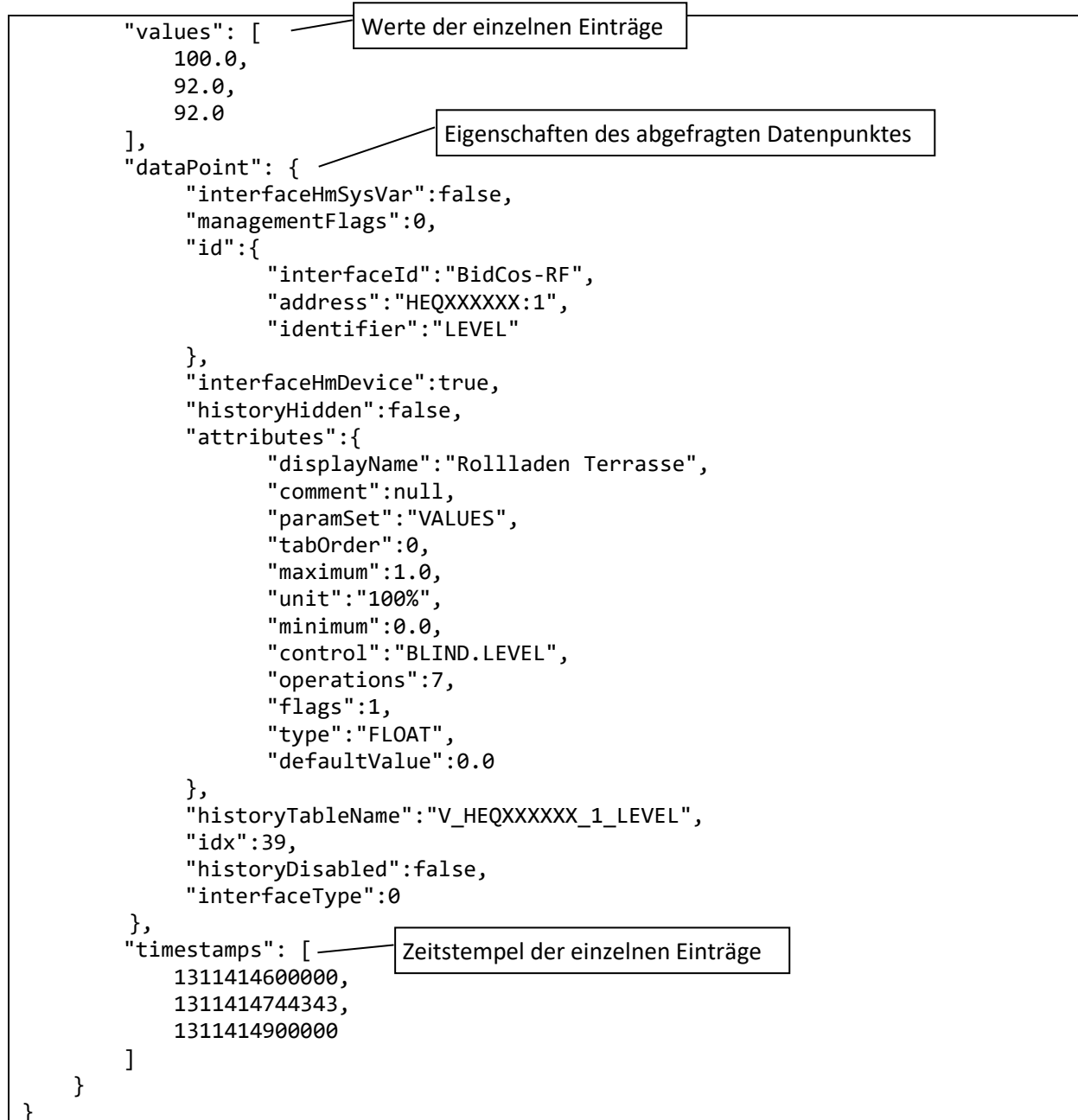
Beispielantwort:

```

{
    "id": 123,
    "result": {
        "states": [
            1,
            1,
            1
        ],

```

Status der einzelnen Einträge



- **getTimeSeriesRaw**: Die Rohdaten einer Zeitreihe von einem Datenpunkte abfragen.

Als Parameter werden die Historian-ID, die Anfangszeit und die Endzeit des abzufragenden Zeitbereichs übergeben. Alternativ kann auch der ISE-Name (z.B. "BidCos-RF.ABC0123456:2.STATE") eines Datenpunktes anstatt der Historian-ID angegeben werden. Es werden nur Einträge zurückgegeben, die der Bedingung *Anfangszeit* ≤ *Zeitstempel des Eintrags* < *Endzeit* entsprechen.

Beispielanfrage:

```
{
  "id": 123,
  "method": "getTimeSeriesRaw",
  "params": [
    14,
    1311414600000,
    1311415500000
  ]
}
```

entspricht 2011-07-23 11:50:00

entspricht 2011-07-23 12:05:00

Beispielantwort:

```
{
  "id": 123,
  "result": {
    "states": [
      1,
      1,
      1
    ],
    "values": [
      92.0,
      92.0,
      92.0
    ],
    "dataPoint": {
      "interfaceHmSysVar": false,
      "managementFlags": 0,
      "id": {
        "interfaceId": "BidCos-RF",
        "address": "HEQXXXXXX:1",
        "identifizier": "LEVEL"
      },
      "interfaceHmDevice": true,
      "historyHidden": false,
      "attributes": {
        "displayName": "Rollladen Terrasse",
        "comment": null,
        "paramSet": "VALUES",
        "tabOrder": 0,
        "maximum": 1.0,
        "unit": "100%",
        "minimum": 0.0,
        "control": "BLIND.LEVEL",
        "operations": 7,
        "flags": 1,
        "type": "FLOAT",
        "defaultValue": 0.0
      },
      "historyTableName": "V_HEQXXXXXX_1_LEVEL",
      "idx": 39,
      "historyDisabled": false,
      "interfaceType": 0
    },
    "timestamps": [
      1311414744343,
      1311415037609,
      1311415302140
    ]
  }
}
```

Status der einzelnen Einträge

Werte der einzelnen Einträge

Eigenschaften des abgefragten Datenpunktes

Zeitstempel der einzelnen Einträge

```
}
}
]
```

- **getValue:** Den aktuellen Wert eines oder mehrerer Datenpunkte abfragen.

Als Parameter wird entweder nur die Historian-ID eines Datenpunktes übergeben oder ein JSON-Array mit den Historian-IDs mehrerer Datenpunkte. Alternativ können anstatt der Historian-IDs auch die ISE-Name (z.B. "BidCos-RF.ABC0123456:2.STATE") angegeben werden.

Beispielanfrage 1 (ein Datenpunkt):

```
{
  "id": 123,
  "method": "getValue",
  "params": [14]
}
```

Beispielantwort 1 (ein Datenpunkt):

```
{
  "id": 123,
  "result": {
    "timestamp": 1331491325804,
    "value": 32.0,
    "state": 1
  }
}
```

Zeitstempel der letzten Änderung:
entspricht 2012-03-11 19:42:05

aktueller Wert

Beispielanfrage 2 (mehrere Datenpunkte):

```
{
  "id": 123,
  "method": "getValue",
  "params": [
    [12, 14, 16]
  ]
}
```

Beispielantwort 2 (mehrere Datenpunkte):

```
{
  "id": 123,
  "result": [
    {
      "timestamp": 1331491329494,
      "value": 33.0,
      "state": 1
    },
    {
      "timestamp": 1331491325804,
      "value": 32.0,
      "state": 1
    },
    {
      "timestamp": 1331488593167,
      "value": 0.0,
      "state": 1
    }
  ]
}
```

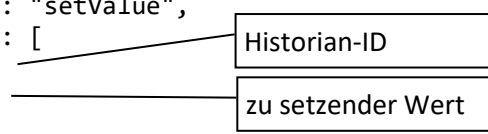
```
} ]  
}
```

- **setValue:** Den aktuellen Wert eines Datenpunktes im Gerät oder der CCU setzen.

Als Parameter werden die Historian-ID des Datenpunktes und der zu setzende Wert übergeben. Alternativ kann auch der ISE-Name (z.B. "BidCos-RF.ABC0123456:2.STATE") eines Datenpunktes anstatt der Historian-ID angegeben werden. Das Ergebnis des Methodenaufufes bei fehlerfreier Ausführung ist immer **null**. Der zu setzende Wert sollte dem Datentyp des Datenpunktes entsprechen, ansonsten wird eine Konvertierung versucht. Für das erfolgreiche Setzen muss die Konfigurationsoption *devices.device<G-Nr>.writeAccess = true* gesetzt sein und der Datenpunkt muss beschreibbar sein. Dies kann auf der Detail-Seite zu einem Datenpunkt eingesehen werden (s.a. Abschnitt 7.1).

Beispielanfrage:

```
{  
  "id": 456,  
  "method": "setValue",  
  "params": [  
    122,  
    1  
  ]  
}
```



Beispielantwort:

```
{  
  "id": 456,  
  "result": null  
}
```

7.3 Export von Zeitreihen

Über eine speziell formatierte Web-Adresse können Zeitreihen aus der Datenbank im CSV-Format auf dem lokalen Rechner abgespeichert werden. Das Format ist so gewählt, dass die Datei ohne Anpassungen in einem deutschen MS Excel geöffnet werden kann.

Die Web-Adresse muss dazu folgenden Aufbau haben. Optionale Komponenten sind in eckigen Klammern [] dargestellt:

**http://localhost[:Portnummer]/query/csv.gy?i=Historian-ID[&b=Beginn][&e=Ende][&d=Länge]
[&k=Schlüsselwort]**

Die einzelnen Komponenten haben folgende Bedeutung:

| Parameter | Bezeichnung | Beschreibung |
|-----------|-------------|--|
| | Portnummer | Die Portnummer muss angegeben werden, wenn die Konfigurationsoption <i>webServer.port</i> nicht den Standardwert 80 besitzt. |

| | | |
|----------|----------------------------|---|
| i | Historian-ID oder ISE-Name | Die Historian-ID des Datenpunktes (s.a. Abschnitt 7.1). Alternativ kann auch der ISE-Name (z.B. "BidCos-RF.ABC0123456:2.STATE") eines Datenpunktes anstatt der Historian-ID angegeben werden. |
| b | Zeitbereichsanfang | Anfang des zu exportierenden Zeitbereichs (optional) |
| e | Zeitbereichsende | Ende des zu exportierenden Zeitbereichs (optional) |
| d | Zeitbereichslänge | Länge des zu exportierenden Zeitbereichs (optional) |
| k | Schlüsselwort | Falls die Konfigurationsoption <i>webServer.apiKeys</i> gesetzt ist, muss eins von den dort angegebenen Schlüsselwörtern mit übertragen werden. |

Die Parameter **b**, **d** und **e** sind optional. Folgende Kombinationen können angegeben werden:

| Parameter | Zeitbereich |
|--------------|--|
| b, e | vom angegeben Beginn bis zum angegebenen Ende |
| b, d | vom angegeben Beginn mit der angegebenen Länge |
| e, d | vor dem angegeben Ende mit der angegeben Länge |
| keine | die letzten 24 Stunden. |
| b | vom angegebenen Beginn bis jetzt |
| d | die angegebene Länge bis jetzt |

Zeitbereichsanfang und -ende müssen das Format *YYYYMMDDhhmmss* besitzen. **YYYY** ist der Platzhalter für das Jahr, **MM** für den Monat, **DD** für den Tag, **hh** für die Stunde, **mm** für die Minute und **ss** für die Sekunde. Platzhalter am Ende können weggelassen werden, z.B. reicht auch die Angabe *YYYYMMDD*. Es folgen ein paar Beispiele:

| Zeitpunkt | Kodierung |
|----------------------------|----------------|
| 01.02.2011 12:13:14 | 20110201121314 |
| 03.02.2011 12:00:00 | 2011020312 |
| 01.02.2011 00:00:00 | 201102 |
| 01.01.2011 00:00:00 | 2011 |

Die Zeitbereichslänge besteht aus mindestens einer Komponente in dem Format *Anzahl/Zeitraum*. Für den Zeitraum sind folgende Angaben zulässig: **Y** (Jahre), **M** (Monate), **W** (Wochen), **D** (Tage), **h** (Stunden), **m** (Minuten), **s** (Sekunden). Es können mehrere Zeiträume direkt hintereinander angegeben werden, die Zeiträume werden dann addiert. Es folgen ein paar Beispiele:

| Zeitraum | Kodierung |
|--|-----------|
| 3 Tage | 3D |
| 1 Woche | 1W |
| 1 Woche und 2 Tage | 1W2D |
| 1 Tag, 4 Stunden und 10 Minuten | 1D4h10m |

Hier sind noch einige komplette Beispiele:

| Web-Adresse | Bedeutung |
|--|---|
| http://localhost/query/csv.gy?i=1 | Die letzten 24 Stunden vom Datenpunkt mit |

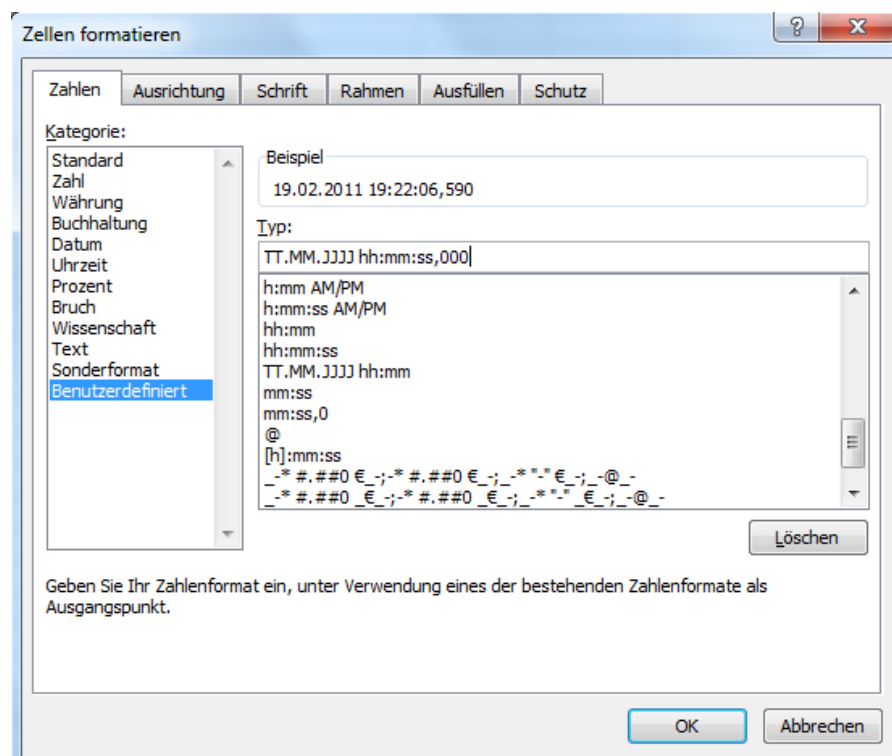
| | |
|---|--|
| http://localhost/query/csv.gy?i=3&d=1W | der Historian-ID 1 exportieren. Die letzte Woche vom Datenpunkt mit der Historian-ID 3 exportieren. |
| http://localhost/query/csv.gy?i=64&b=20110227 | Alle Daten vom 27.2.2011 bis jetzt vom Datenpunkt mit der Historian-ID 64 exportieren. |
| http://localhost/query/csv.gy?i=16&b=2011&d=1W | Die erste Woche vom Jahr 2011 vom Datenpunkt mit der Historian-ID 16 exportieren. |

Bei Eingabe der so gebildeten Web-Adresse in einen Web-Browser, wird die CSV-Datei mit den gewünschten Daten generiert und an den Web-Browser geschickt. Dieser kann sie dann speichern oder direkt von MS Excel öffnen lassen.

Eine exportierte Zeitreihe sieht in MS Excel zum Beispiel wie folgt aus:

| | A | B |
|---|-------------------------|--------|
| 1 | ABC3000000:1.VALUE0 | |
| 2 | Zeitstempel | Wert |
| 3 | 25.02.2011 19:27:00,543 | 82,598 |
| 4 | 26.02.2011 08:44:33,635 | 62,167 |
| 5 | 26.02.2011 08:45:03,705 | 80,226 |
| 6 | 26.02.2011 08:45:33,762 | 28,892 |
| 7 | 26.02.2011 08:46:03,833 | 53,934 |
| 8 | 26.02.2011 08:46:33,916 | 84,205 |

Damit die Zeitstempel in der vollen Auflösung angezeigt werden, sind die Zellen mit den Zeitstempeln wie folgt zu formatieren:



7.4 Generierung von Trend-Diagrammen

Über eine speziell formatierte Web-Adresse kann der Web-Server des CCU-Historians auch direkt eine Grafik zu einer Zeitreihe generieren. Die Grafik wird im PNG-Format an den Web-Browser geschickt. Sie kann damit einfach in eigene Web-Seiten eingebunden werden.

Die Web-Adresse muss dazu folgenden Aufbau haben. Optionale Komponenten sind in eckigen Klammern [] dargestellt:

http://localhost[:Portnummer]/query/trend.gy?i=Historian-ID[&g=Gruppe][&gh=Gruppenhöhe][&b=Beginn][&e=Ende][&d=Länge][&w=Breite][&h=Höhe][&t=Aussehen][&k=Schlüsselwort]

Die einzelnen Komponenten haben folgende Bedeutung:

| Parameter | Bezeichnung | Beschreibung |
|-----------|----------------------------|--|
| | Portnummer | Die Portnummer muss angegeben werden, wenn die Konfigurationsoption <i>webServer.port</i> nicht den Standardwert 80 besitzt. |
| i | Historian-ID oder ISE-Name | Die Historian-ID des Datenpunktes (s.a. Abschnitt 7.1); Dieser Parameter kann mehrfach angegeben werden, um mehrere Datenpunkte in einem Trend darzustellen (z.B. <i>i=10&i=12</i>). Alternativ kann auch der ISE-Name (z.B. "BidCos-RF.ABC0123456:2.STATE") eines Datenpunktes anstatt der Historian-ID angegeben werden. |
| g | Gruppe | Zu jedem Parameter i kann eine Gruppe angegeben werden. Die Gruppennummern können frei gewählt werden. Beispiel: Die Datenpunkte 1, 2 sollen in einer Gruppe dargestellt werden und die Datenpunkte 3, 4, 5 in einer weiteren. Zusammen mit den i Parametern ergibt sich folgende Web-Adresse: ...&i=1&i=2&i=3&i=4&i=5& g=1&g=1&g=2&g=2&g=2 ... (optional) |
| gh | Gruppenhöhe | Zu jeder Gruppe kann eine Höhe angegeben werden. Die Gruppenhöhe bestimmt wieviel Zeichenfläche einer Gruppe zugewiesen wird. Beispiel: Wenn die Gruppenhöhen auf 1, 2 und 4 gesetzt werden, dann betragen die relativen Anteile an der Zeichenfläche 1/7, 2/7 und 4/7 (7 ist die Summe der individuellen Höhen). Die Anzahl der gh -Parameter richtig sich nach der Anzahl der eindeutigen Gruppennummern der g -Parameter. (optional) |
| b | Zeitbereichsanfang | Anfang des zu exportierenden Zeitbereichs (optional) |
| e | Zeitbereichsende | Ende des zu exportierenden Zeitbereichs (optional) |
| d | Zeitbereichslänge | Länge des zu exportierenden Zeitbereichs (optional) |
| w | Breite | Breite der generierten Grafik in Pixeln (optional, Standardwert: 640) |
| h | Höhe | Höhe der generierten Grafik in Pixeln (optional, Standardwert: 240) |
| t | Aussehen | Bezeichner zur Auswahl des Satzes an Einstellungen für das Aussehen der generierten Grafik (optional, Standardwert: <i>default</i>). Der angegebene Bezeichner muss vorher in der |

| | | |
|----------|---------------|---|
| | | Konfigurationsdatei definiert worden sein (s.a. Abschnitt 4.1 und Abschnitt 7.4.1). |
| k | Schlüsselwort | Falls die Konfigurationsoption <i>webServer.apiKeys</i> gesetzt ist, muss eins von den dort angegebenen Schlüsselwörtern mit übertragen werden. |

Durch Mehrfachangabe des Parameters **i** können mehrere Datenpunkte in einem Trend-Diagramm dargestellt werden. Wenn sich die Wertebereiche oder die Einheiten der Datenpunkte unterscheiden, werden automatisch zusätzliche Y-Skalen angelegt.

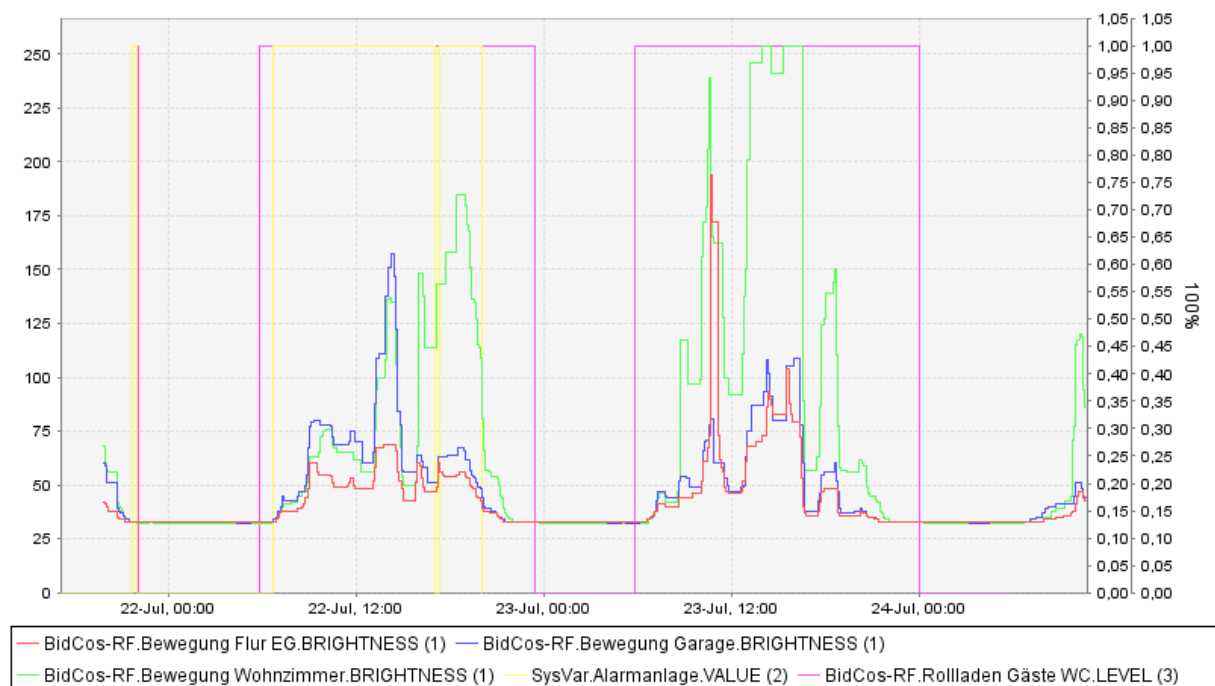
Mit dem Parameter **g** können Kurvengruppen gebildet werden. Jede Kurvengruppe wird in einem eigenen Zeichenbereich dargestellt. Die einzelnen Kurvengruppen werden dann übereinander mit einer gemeinsamen Zeitachse dargestellt. Wenn der Parameter **g** nicht verwendet wird, werden alle angegebenen Datenpunkte in einer Gruppe - also in einem Zeichenbereich – dargestellt.

Für das Format der Parameter **b**, **d** und **e** wird auf Abschnitt 7.2 verwiesen.

Hier sind noch einige Beispiele:

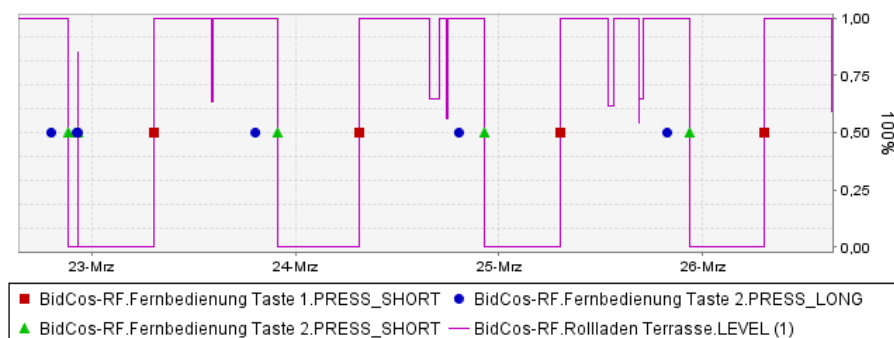
| Web-Adresse | Bedeutung |
|--|---|
| http://localhost/query/trend.gy?i=1 | Die letzten 24 Stunden vom Datenpunkt mit der Historian-ID 1 als Grafik darstellen. |
| http://localhost/query/trend.gy?i=3&d=1W | Die letzte Woche vom Datenpunkt mit der Historian-ID 3 als Grafik darstellen. |
| http://localhost/query/trend.gy?i=64&b=20110227 | Alle Daten vom 27.2.2011 bis jetzt vom Datenpunkt mit der Historian-ID 64 als Grafik darstellen. |
| http://localhost/query/trend.gy?i=16&b=2011&d=1W | Die erste Woche vom Jahr 2011 vom Datenpunkt mit der Historian-ID 16 als Grafik darstellen. |
| http://localhost/query/trend.gy?i=10&i=12&i=14&i=33 | Die letzten 24 Stunden der Datenpunkte mit den Historian-IDs 10, 12, 14 und 33 werden in einem Diagramm dargestellt. |
| http://localhost/query/trend.gy?i=10&i=20&g=1&g=2 | Die Datenpunkte mit den IDs 10 und 20 werden übereinander in zwei getrennten Zeichenbereichen dargestellt. |
| http://localhost/query/trend.gy?i=10&i=20&i=30&g=1&g=1&g=2&gh=2&gh=1 | Die Datenpunkte mit den IDs 10 und 20 werden in einer Gruppe der Höhe 2/3 und der Datenpunkt mit der ID 30 mit der Höhe 1/3 gezeichnet. |

Ein generiertes Trend-Diagramm für fünf Datenpunkte sieht zum Beispiel wie folgt aus:

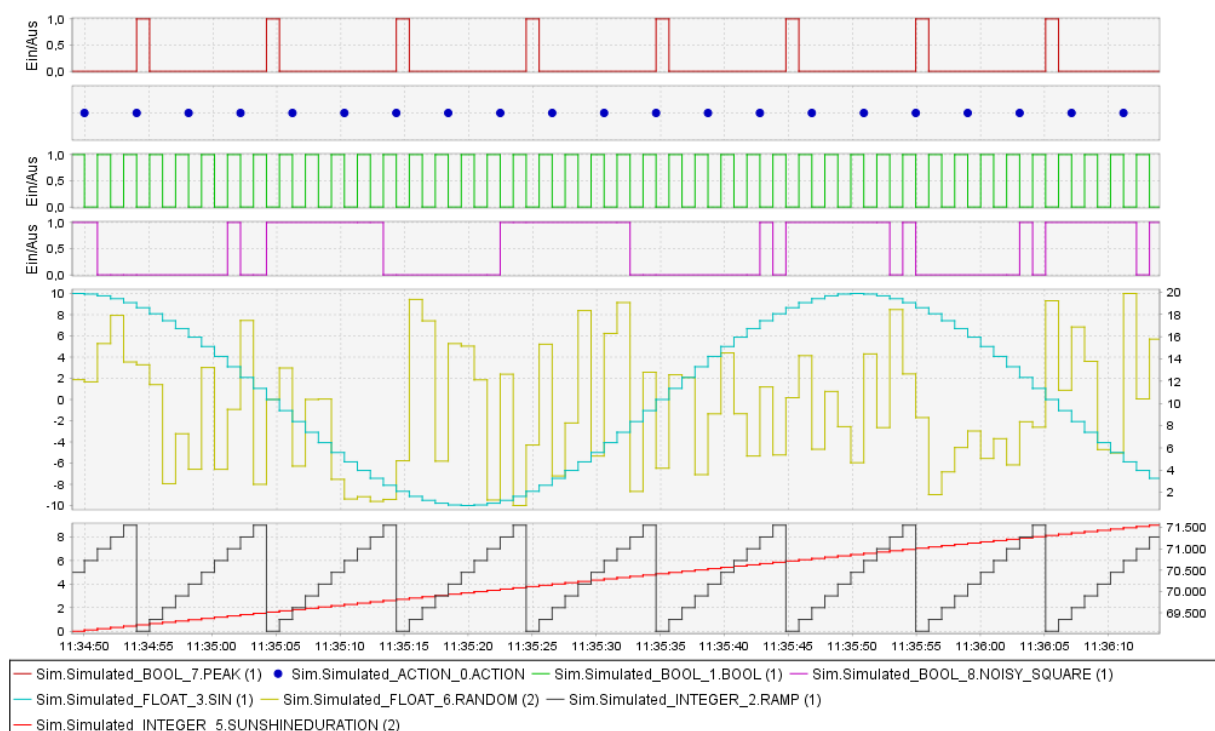


Im Diagramm werden die Trends der fünf Datenpunkte mit unterschiedlichen Farben, die aus der Legende ersichtlich sind, dargestellt. Hinter dem Datenpunktnamen wird jeweils in Klammern die zugehörige Y-Skala angegeben. Die Nummerierung der Skalen erfolgt von links nach rechts. In diesem Diagramm verwenden die ersten drei Datenpunkte die Skala links, weil sowohl die Wertebereiche als auch die Einheiten (in diesem Fall sind die Datenpunkte ohne Einheit) identisch sind. Falls ein Datenpunkt eine Einheit besitzt, wird diese an der Y-Skala mit angegeben (siehe auch Skala 3).

Aktionen (z.B. Tastendrucke) von Datenpunkten vom Typ *ACTION* werden als Symbole dargestellt. Zum Zeitpunkt jeder Aktion wird ein Symbol gezeichnet:



Durch Verwendung der URL-Parameter **g** und **gh** sind z.B. folgende Trend-Darstellungen möglich:



7.4.1 Anpassung der Trend-Darstellung

Die Trend-Darstellung kann vielfältig angepasst werden. Es können alle Farben, die Strichstärken, die Texte für die Beschriftungen und vieles mehr geändert werden. Ein Satz an Einstellungen wird unter einem frei gewählten Bezeichner in der Konfigurationsdatei definiert (s.a. Abschnitt 4).

Bei der Trend-Generierung kann dann in der Web-Adresse mit dem Parameter **t** (Aussehen) der vorher definierte Satz an Einstellungen aktiviert werden.

Es können mehrere Einstellungssätze definiert werden, und bei jeder Trend-Generierung ein anderer Satz ausgewählt werden. Die Standarddarstellung (z.B. auf den CCU-Historian Web-Seiten) kann durch Definition eines Einstellungssatzes mit dem Bezeichner *default* geändert werden.

Die Grundstruktur von einem Satz an Einstellungen sieht in der Konfigurationsdatei wie folgt aus. Die Bereiche *Bezeichner*, *Einstellungen zur Diagrammfläche*, usw. werden dann je nach Anforderung gefüllt:

```
webServer.trendDesigns.Bezeichner.displayName='Anzeigename dieses Designs'
webServer.trendDesigns.Bezeichner.chart={
    Einstellungen zur Diagrammfläche
}
webServer.trendDesigns.Bezeichner.plot={
    Einstellungen zur Kurvenfläche
}
webServer.trendDesigns.Bezeichner.timeAxis={
    Einstellungen zur Zeitachse
}
webServer.trendDesigns.Bezeichner.rangeAxes=[
    {
        Einstellungen zur ersten Y-Skala
    },
    {
```

```

        Einstellungen zur zweiten Y-Skala
    },
    {
        usw.
    }
]
webServer.trendDesigns.Bezeichner.series=[
    {
        Einstellungen zur ersten Datenserie
    },
    {
        Einstellungen zur zweiten Datenserie
    },
    {
        usw.
    }
]
webServer.trendDesigns.Bezeichner.renderers=[
    {
        Einstellungen zur ersten Kurve
    },
    {
        Einstellungen zur zweiten Kurve
    },
    {
        usw.
    }
]

```

Der Bezeichner eines Einstellungssatzes kann aus Buchstaben und Ziffern bestehen, er muss aber mit einem Buchstaben beginnen (z.B. default, design3).

Es können beliebig viele Einstellungssätze definiert werden. Der Bezeichner muss allerdings eindeutig sein.

Alle Bereiche sind optional. Es kann auch nur ein Bereich (**chart**, **plot**, **timeAxis**, **rangeAxes**, **series** oder **renderers**) angegeben werden. Für die nicht angegebenen Bereiche werden die Standardeinstellungen verwendet.

Wenn im generierten Trend-Diagramm mehr Y-Skalen/Kurven vorkommen, als im Bereich **rangeAxes/series/renderers** angegeben worden sind, so werden die Einstellungen zyklisch durchlaufen. Wenn z.B. drei Y-Skalen konfiguriert worden sind, so bekommt eine eventuelle vierte Y-Skala die Einstellungen von der ersten Y-Skala. Oder wenn z.B. nur die Einstellungen für eine Kurve konfiguriert worden ist, werden die Einstellungen für alle Kurven verwendet.

Beliebige Farben werden wie folgt angegeben:

```
new ChartColor(Rotanteil, Grünanteil, Blauanteil)
```

Die Farbanteile werden im Bereich von 0 bis 255 angegeben. Es kann auch eine Farbkonstante der folgenden Liste verwendet werden:

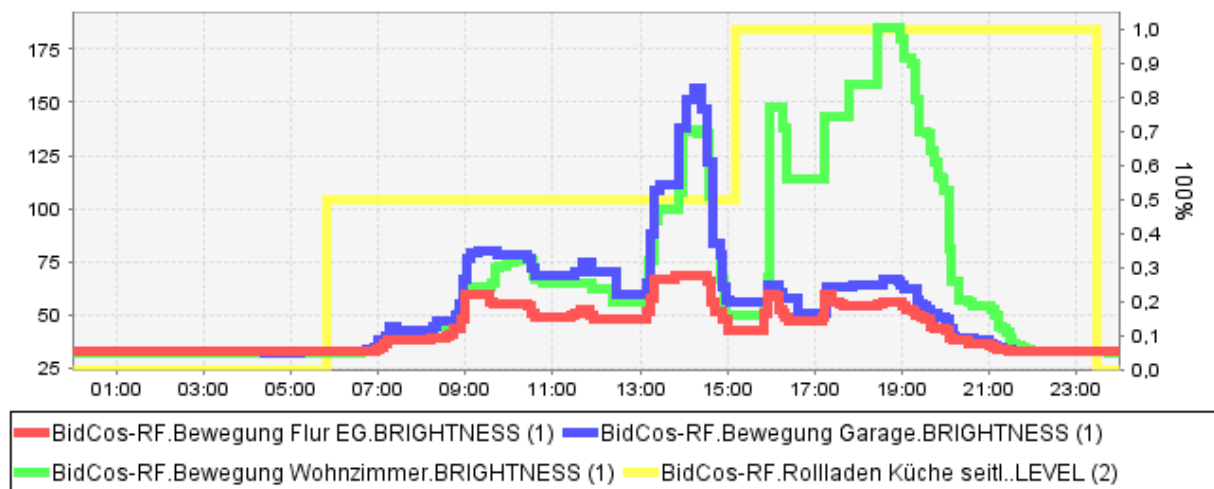
| | |
|------------------|----------------------|
| ChartColor.BLACK | ChartColor.BLUE |
| ChartColor.CYAN | ChartColor.DARK_BLUE |

| | |
|-------------------------------|-----------------------------|
| ChartColor.DARK_CYAN | ChartColor.DARK_GRAY |
| ChartColor.DARK_GREEN | ChartColor.DARK_MAGENTA |
| ChartColor.DARK_RED | ChartColor.DARK_YELLOW |
| ChartColor.GRAY | ChartColor.GREEN |
| ChartColor.LIGHT_BLUE | ChartColor.LIGHT_CYAN |
| ChartColor.LIGHT_GRAY | ChartColor.LIGHT_GREEN |
| ChartColor.LIGHT_MAGENTA | ChartColor.LIGHT_RED |
| ChartColor.LIGHT_YELLOW | ChartColor.MAGENTA |
| ChartColor.ORANGE | ChartColor.PINK |
| ChartColor.RED | ChartColor.VERY_DARK_BLUE |
| ChartColor.VERY_DARK_CYAN | ChartColor.VERY_DARK_GREEN |
| ChartColor.VERY_DARK_MAGENTA | ChartColor.VERY_DARK_RED |
| ChartColor.VERY_DARK_YELLOW | ChartColor.VERY_LIGHT_BLUE |
| ChartColor.VERY_LIGHT_CYAN | ChartColor.VERY_LIGHT_GREEN |
| ChartColor.VERY_LIGHT_MAGENTA | ChartColor.VERY_LIGHT_RED |
| ChartColor.VERY_LIGHT_YELLOW | ChartColor.WHITE |
| ChartColor.YELLOW | |

Die Möglichkeiten sollen nun anhand von Beispielen aufgezeigt werden. Die jeweiligen Ausschnitte können zum Test in die Konfigurationsdatei vom CCU-Historian kopiert werden. Der CCU-Historian übernimmt eine geänderte Konfiguration nach spätestens 15 Sekunden. Die Beispiele enthalten Kommentare, um die jeweiligen Einstellungen zu beschreiben. Nach jedem Beispiel ist eine mit diesen Einstellungen generierte Grafik abgebildet.

Beispiel 1: Strichstärken der Kurven anpassen

```
// Mit dem Bezeichner default werden die Standardeinstellungen für
// die Darstellung geändert.
webServer.trendDesigns.default=new TrendDesign(
    renderers: [
        // Es wird nur eine Kurve definiert. Die Einstellungen werden
        // also für alle Kurven übernommen.
        {
            // Die Strichstärke wird auf 5 Pixel gesetzt.
            stroke=new BasicStroke(5)
        }
    ]
)
```

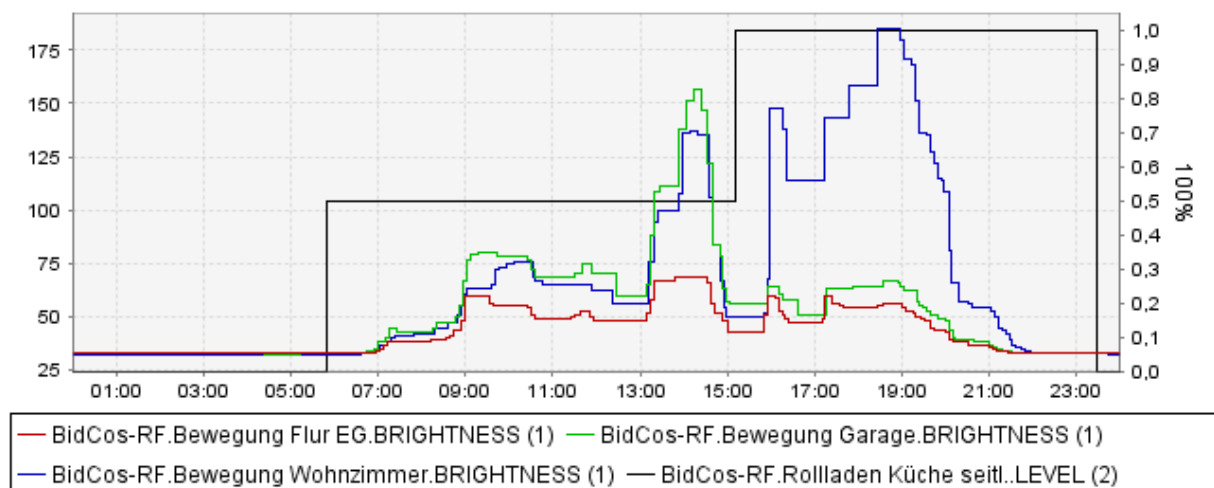


Beispiel 2: Kurvenfarben anpassen

```

webServer.trendDesigns.default=new TrendDesign(
    renderers: [
        // Die Kurvenfarben von vier Kurven werden gesetzt.
        {
            // dunkles Rot
            paint=new ChartColor(190, 0, 0)
        },
        {
            // dunkles Blau
            paint=new ChartColor(0, 190, 0)
        },
        {
            // dunkles Grün
            paint=new ChartColor(0, 0, 190)
        },
        {
            // Schwarz
            paint=ChartColor.BLACK
        }
    ]
)

```



Beispiel 3: Hintergrund- und Skalenfarben, Skalenbeschriftungen ändern, Legende entfernen

```

webServer.trendDesigns.default=new TrendDesign(
    chart: {
        // Farbe der Diagrammfläche
        backgroundPaint=ChartColor.GRAY

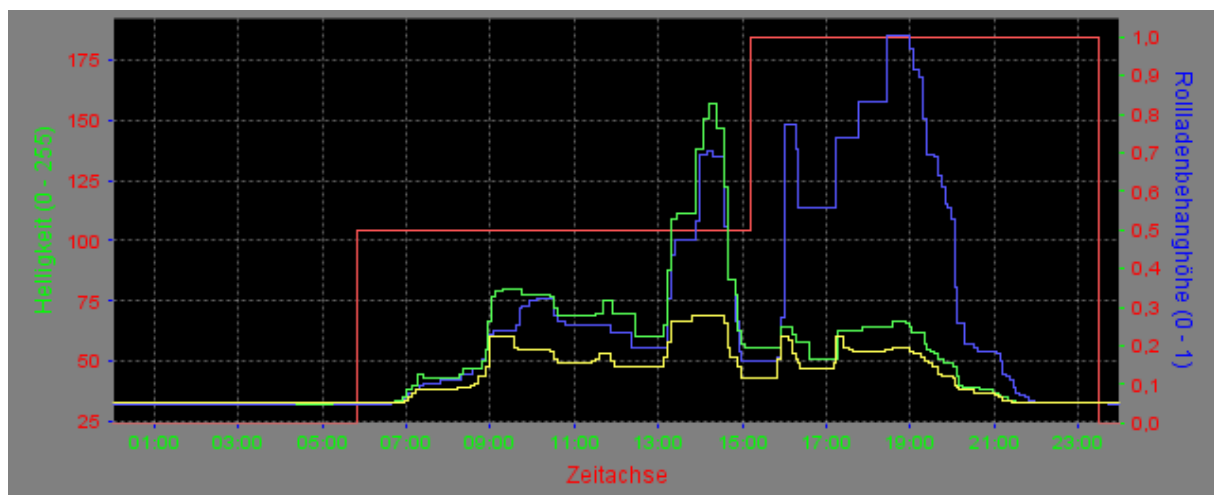
        // Diagramm ohne Legende
        removeLegend()
    },
    plot: {
        // Farbe der Kurvenfläche
        backgroundPaint=ChartColor.BLACK
    },
    timeAxis: {
        // Zeitachse mit Beschriftung
        label='Zeitachse'

        // Farbe der Beschriftung
        labelPaint=ChartColor.RED

        // Farbe der Skalen-Werte
        tickLabelPaint=ChartColor.GREEN

        // Farbe der Markierungen
        tickMarkPaint=ChartColor.BLUE
    },
    rangeAxes: [
        {
            label='Helligkeit (0 - 255)'
            labelPaint=ChartColor.GREEN
            tickLabelPaint=ChartColor.RED
            tickMarkPaint=ChartColor.BLUE
        },
        {
            label='Rollladenbehanghöhe (0 - 1)'
            labelPaint=ChartColor.BLUE
            tickLabelPaint=ChartColor.RED
            tickMarkPaint=ChartColor.GREEN
        }
    ]
)

```

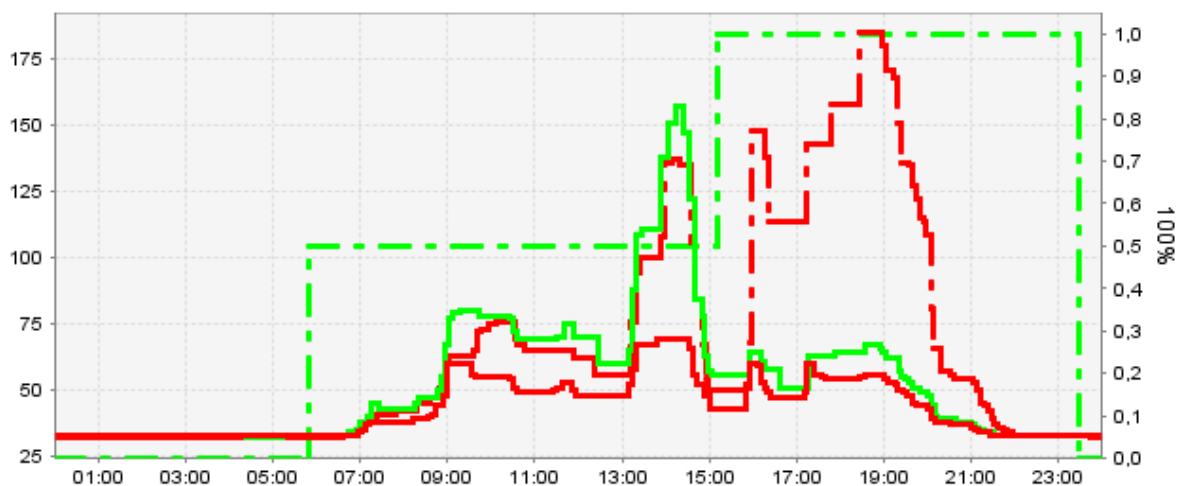


Beispiel 4: Gestrichelte Kurven

```

webServer.trendDesigns.default=new TrendDesign(
    chart: {
        removeLegend()
    },
    renderers: [
        {
            // 3 Pixel Strichstärke
            stroke=new BasicStroke(3)
            // Farbe Rot
            paint=ChartColor.RED
        },
        {
            stroke=new BasicStroke(3)
            // Farbe Grün
            paint=ChartColor.GREEN
        },
        {
            // 3 Pixel Strichstärke
            // gestrichelt (15 Pixel gezeichnet, 10 Pixel Leerraum,
            // 3 Pixel gezeichnet, 10 Pixel Leerraum)
            stroke=new BasicStroke(3, BasicStroke.CAP_ROUND,
                BasicStroke.JOIN_ROUND, 0, [15, 10, 3, 10] as float[], 0)
            // Farbe Rot
            paint=ChartColor.RED
        },
        {
            stroke=new BasicStroke(3, BasicStroke.CAP_ROUND,
                BasicStroke.JOIN_ROUND, 0, [15, 10, 3, 10] as float[], 0)
            paint=ChartColor.GREEN
        }
    ]
)

```



Beispiel 5: Farbverläufe, Titel und Diagrammrahmen

```

webServer.trendDesigns.default=new TrendDesign(
    chart: {
        // Titel des Diagramms
        title=new TextTitle('Titel des Diagramms')
        // Legende entfernen
        removeLegend()
    }
)

```

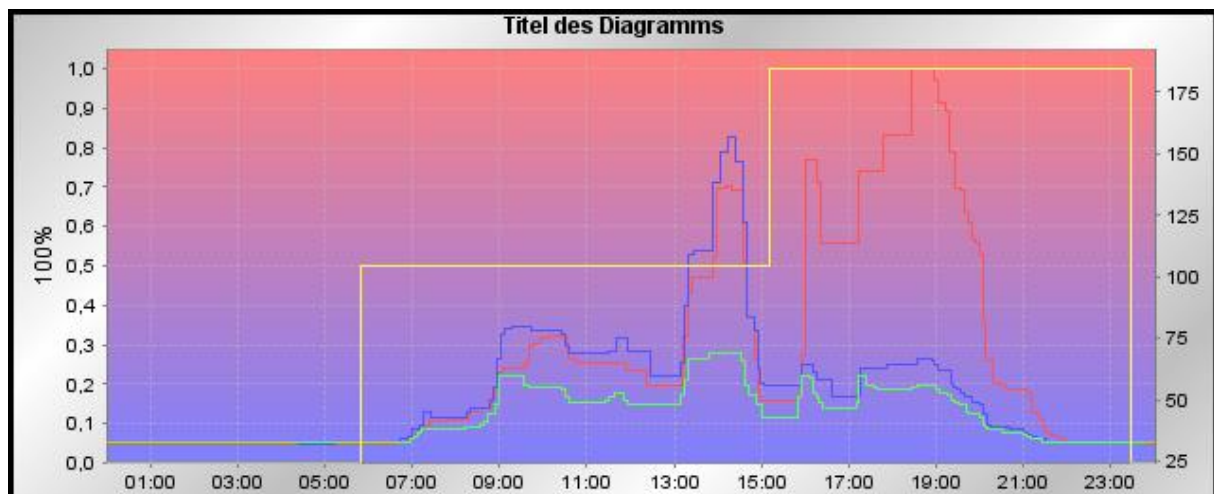


```

        // Diagonaler Farbverlauf in der Diagrammfläche
        // 0, 0: Startpunkt
        // ChartColor.LIGHT_GRAY: Startfarbe
        // 100, 100: Endpunkt
        // ChartColor.WHITE: Endfarbe
        // true: Farbverlauf wiederholen
        backgroundPaint=new GradientPaint(
            0, 0, ChartColor.LIGHT_GRAY,
            100, 100, ChartColor.WHITE, true
        )

        // Rand um das Diagramm zeichnen
        borderVisible=true
        // in Schwarz
        borderPaint=ChartColor.BLACK
        // 3 Pixel breit
        borderStroke=new BasicStroke(5)
    },
    plot: {
        // Farbverlauf in der Diagrammfläche
        // new ChartColor(128, 128, 255): Farbe unten
        // new ChartColor(255, 128, 128): Farbe oben
        // (Diagonale Verläufe sind nicht möglich.)
        backgroundPaint=new GradientPaint(
            0, 0, new ChartColor(128, 128, 255),
            0, 0, new ChartColor(255, 128, 128)
        )
    }
}
)

```



Beispiel 6: Farben und Kurvennamen in der Legende anpassen

```

webServer.trendDesigns.default=new TrendDesign(
    chart: {
        // Hintergrundfarbe der Legende
        legend.backgroundPaint=ChartColor.LIGHT_GRAY

        // Textfarbe der Legende
        legend.itemPaint=ChartColor.DARK_RED

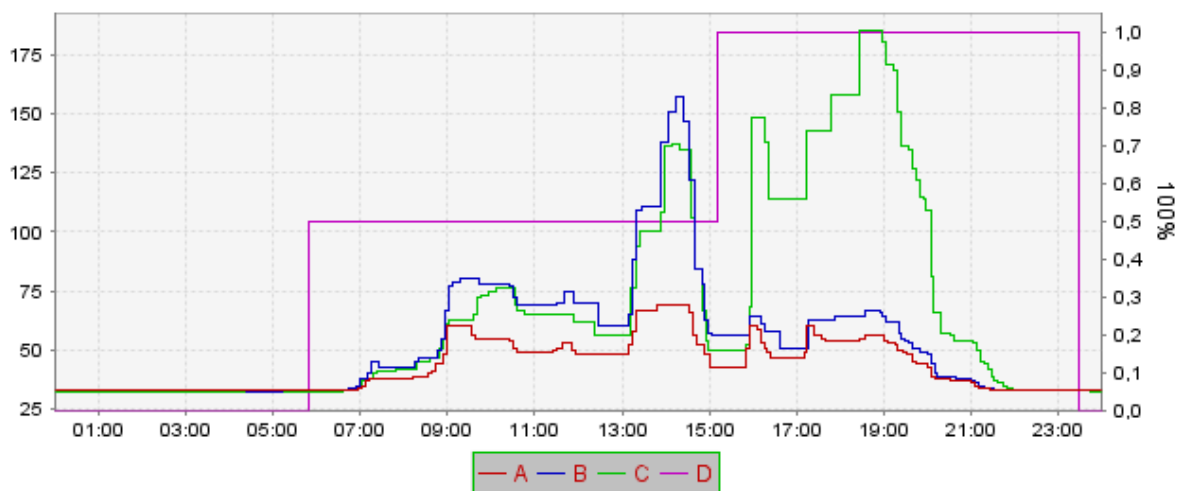
        // Rahmen der Legende
        legend.frame.paint=ChartColor.DARK_GREEN
    }
)

```

```

    },
    series: [
        {
            // Name der Datenserie (auch in der Legende)
            key='A'
        }, {
            key='B'
        }, {
            key='C'
        }, {
            key='D'
        }
    ]
)

```

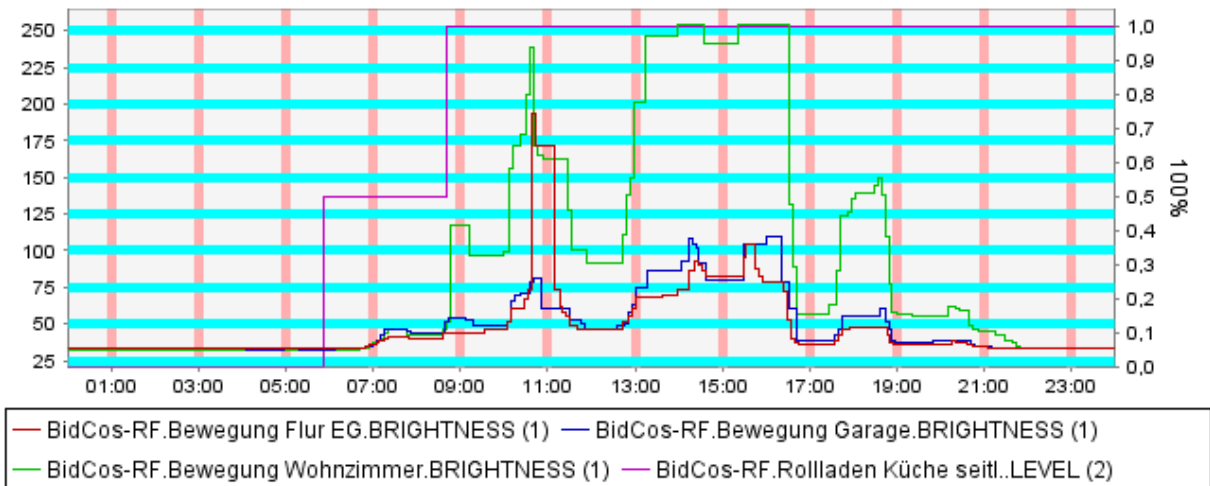


Beispiel 7: Gitterlinien in der Diagrammfläche anpassen

```

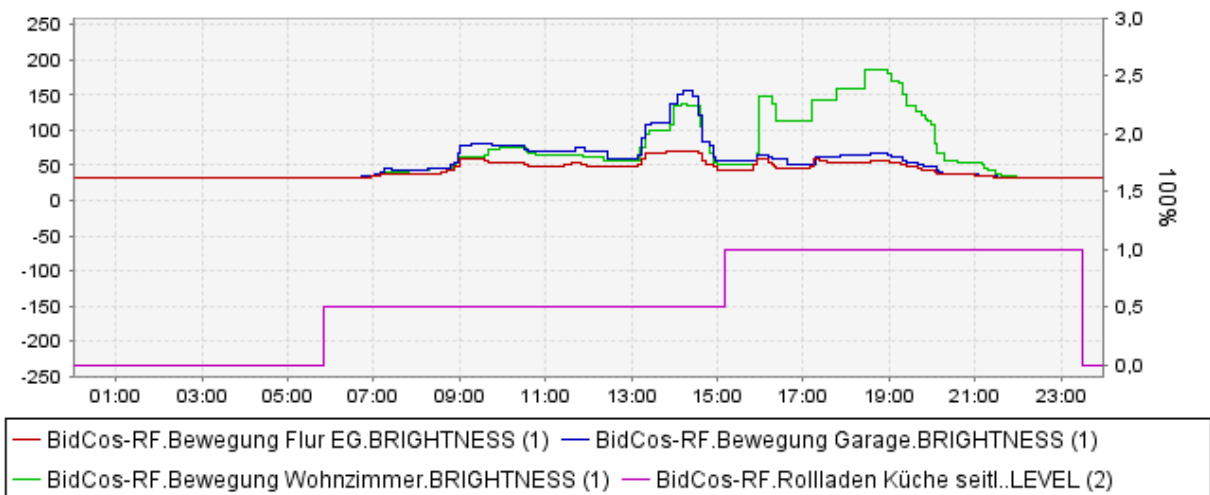
webServer.trendDesigns.default=new TrendDesign(
    plot: {
        // Linien für die Zeitachse
        domainGridlinePaint=ChartColor.PINK
        domainGridlineStroke=new BasicStroke(5)
        // Linien für die Messwertachse
        // (Das Raster orientiert sich immer an der ersten Y-Skala.)
        rangeGridlinePaint=ChartColor.CYAN
        rangeGridlineStroke=new BasicStroke(5)
    }
)

```



Beispiel 8: Anfang und Ende der Y-Skalen festsetzen

```
webServer.trendDesigns.default=new TrendDesign(
  rangeAxes: [
    {
      // 1. Y-Skala (links) von -250 bis 260
      setRange(-250, 260)
    },
    {
      // 2. Y-Skala (rechts) von -0,1 bis 3
      setRange(-0.1, 3)
    }
  ]
)
```



7.5 Eigene Web-Seiten

Eigene Web-Seiten können im Verzeichnis *webapp/custom* abgelegt werden. Dort liegende Dateien werden über den eingebauten Web-Server des CCU-Historians zum Herunterladen in einem Web-Browser angeboten.

Dateien mit folgenden Endungen werden vom CCU-Historian vorverarbeitet, um dynamische Web-Seiten erzeugen zu können:

| Dateiendungen | Beschreibung |
|--------------------------|---|
| .html, .htm, .gsp | Template-Dateien (HTML-Dateien mit eingebetteten Groovy-Skripten) |
| .gy, .groovy | Groovy-Dateien (Groovy-Skripte die beliebigen Inhalt generieren können) |

Unzählige Beispiele für diese Dateien sind im CCU-Historian selbst zu finden. Diese befinden sich alle im *webapp*-Unterverzeichnis.

7.5.1 Weiterführende Dokumentation

| Thema | Adressen |
|-----------------------------|---|
| Skriptsprache Groovy | http://groovy-lang.org/ |
| Groovy-Dateien | http://docs.groovy-lang.org/latest/html/documentation/servlet-userguide.html |

7.5.2 Vordefinierte Variablen für die Skripte

Der CCU-Historian fügt zu den bereits vorhandenen globalen Variablen folgende hinzu:

| Name | Bedeutung |
|-------------------------|--------------------------|
| database | Datenbank |
| utils | Hilfsfunktionen |
| interfaceManager | Schnittstellenverwaltung |
| webServer | Web-Server |

8 Unterstützung

Bei Fragen oder Problemen gibt es vielfältige Möglichkeiten der Unterstützung. Als erste Informationsquelle ist dieses Handbuch zu nennen. Hier ist insbesondere ein genaues Studium des Abschnitts 4 *Konfiguration* zu empfehlen.

Des Weiteren wird ein Besuch der Internet-Seiten zum CCU-Historian empfohlen (<http://www.ccu-historian.de/>). Auf diesen Seiten sind auch aktuellere Informationen zu finden, die noch nicht in dieses Handbuch eingepflegt worden sind. Speziell für die Unterstützung wurde die Seite *Tipps & Tricks* angelegt (<http://www.ccu-historian.de/index.php?n=CCU-Historian.TippsAmpTricks>).

Die größte Benutzergemeinde und auch der Entwickler des CCU-Historians ist im HomeMatic-Forum zu finden (<http://homematic-forum.de/>). Dort existiert ein eigenes Forum für den CCU-Historian (HomeMatic Addons → CCU-Historian). Viele Fragen und Probleme wurden in diversen Themen bereits behandelt. Bevor eine neue Frage gestellt wird, sollte kurz mit Hilfe der Foren-Suche festgestellt werden, ob bereits ein entsprechendes Thema mit einer Antwort existiert. Bei vielen Problemen ist die Veröffentlichung des Fehlerprotokolls hilfreich (s.a. Abschnitt 8.1).

Zu guter Letzt kann der Entwickler unter der E-Mail-Adresse info@ccu-historian.de auch direkt erreicht werden. Allerdings ist dort unter Umständen mit längeren Antwortzeiten zu rechnen, da der CCU-Historian ein reines Freizeitprojekt ist.

8.1 Erstellung eines Fehlerprotokolls

Je nach Problem ist ein Fehlerprotokoll zur Analyse sehr hilfreich. Damit kann in den meisten Fällen die Fehlerursache schnell eingegrenzt werden. Um ein Fehlerprotokoll zu erstellen ist folgende Einstellungen in der Konfigurationsdatei vorzunehmen (s.a. Abschnitt 4.1):

```
logSystem.fileLevel=Level.FINEST
```

```
logSystem.binRpcLevel=Level.FINER
```

Dadurch werden Log-Dateien (z.B. *ccu-historian-0.log*) im Installationsverzeichnis des CCU-Historians erstellt. Für die Fehlersuche sind auch die Log-Meldungen **vor** Auftreten des Fehlers notwendig.

9 Anhang

9.1 Änderungshistorie

| Version | Änderungen / Hinweise |
|---------------|---|
| V0.1.0 | <ul style="list-style-type: none">• Erste öffentliche Version mit der Basisfunktionalität |
| V0.2.0 | <ul style="list-style-type: none">• Neue Tabelle <i>DATA_POINTS</i> mit Meta-Informationen zu den Datenpunkten• Neue Web-Schnittstelle:<ul style="list-style-type: none">○ Standardsatz an Web-Seiten zur Erkundung der Datenpunkte und Trend-Darstellung○ Export von Zeitreihen im CSV-Format○ Generierung von Trend-Diagrammen |
| V0.2.1 | <ul style="list-style-type: none">• Überläufe von Zählern (wie z.B. <i>RAIN_COUNTER</i> oder <i>SUNSHINEDURATION</i>) werden behandelt. |
| V0.3.0 | <ul style="list-style-type: none">• Die Systemvariablen der CCU werden zyklisch erkundet und mit Meta-Informationen in der Tabelle <i>DATA_POINTS</i> abgelegt.• Die aktuellen Werte der Systemvariablen werden in Historientabellen abgelegt.• Zu Gerätedatenpunkten werden die vom Benutzer vergebenen Kanalnamen ermittelt.• Neue Konfigurationsoption: <i>historian.sysVarDataCycle</i>• Bei einem Update einer älteren Installation wird die Tabelle <i>DATA_POINTS</i> gelöscht und neu erstellt. |

| | |
|------------------|---|
| V0.3.1 | <ul style="list-style-type: none"> • Neue Konfigurationsoption: <i>ccu.disabled</i> (Betrieb ohne CCU) • Etwas überarbeitete Web-Seiten. • Auswertung der Konfigurationsoption <i>logSystem.fileLevel</i> korrigiert. • Bessere Fehlerbehandlung wenn Systemvariablen gelöscht oder Geräte entfernt worden sind. • XML-Zeichenreferenzen in den Einheiten der Datenpunkte werden dekodiert. Dies wird auch in einer schon vorhandenen Datenbank nachträglich durchgeführt. |
| V0.4.0 | <ul style="list-style-type: none"> • Neue Konfigurationsoptionen: <i>ccu.historianAddress</i>, <i>webServer.historianAddress</i> (Das Setzen dieser Option ist nur sinnvoll, wenn der CCU-Historian-Rechner über mehrere Netzwerkschnittstellen (z.B. auch VPN-Verbindungen) verfügt.) • Generierung von Trend-Diagrammen über die Web-Schnittstelle mit mehreren Datenpunkten • Web-Seite zur interaktiven Darstellung von Trend-Diagrammen • Einfacher Passwortschutz der Web-Seiten • Fehlerkorrekturen: Umlaute auf den Web-Seiten, Link zur Datenbank |
| V0.4.0+HF | <ul style="list-style-type: none"> • Fehler bei der Behandlung von abgelaufenen Web-Sitzungen behoben |
| V0.5.0 | <ul style="list-style-type: none"> • Datenpunkte und Systemvariablen vom Typ Zeichenkette werden archiviert. • Zeitstempelbehandlung von Systemvariablen überarbeitet. |
| V0.5.1 | <ul style="list-style-type: none"> • Steuerzeichen (z.B. \t, \n, usw.) in Zeichenkettensystemvariablen werden unterstützt. |

V0.6.0

- Verschiedene Trend-Darstellungsformen können definiert und je Trend-Generierung ausgewählt werden.
 - Die zu verwendende Konfigurationsdatei kann über einen Startparameter angegeben werden.
 - Der Bereich der Zeitachse wird auf den Abfragezeitraum gesetzt und orientiert sich nicht mehr an den tatsächlich vorhandenen Daten.
 - Trendkurven werden ab dem letzten vorhandenen Datensatz bis zur aktuellen Zeit horizontal verlängert.
 - Die Standardfarben der Kurven sind dunkler.
 - Zusätzlich zu den drei Standard-XMLRPC-Schnittstellen einer CCU können beliebig viele zusätzliche XMLRPC-Schnittstellen abgefragt und archiviert werden.
 - Zeitstempel von Systemvariablen, die mehr als 30 Minuten in der Zukunft liegen, werden verworfen und durch die aktuelle Zeit ersetzt.
 - Filterbare Datenpunktliste nach einer Vorlage des Benutzers *impedance* aus dem FHZ-Forum.
 - Die Schriften auf den Web-Seiten des CCU-Historians wurde etwas verkleinert.
 - Für die bessere Anbindung von [AJAX](#)-Anwendungen wurde der CCU-Historian um eine [JSON-RPC](#)-Schnittstelle erweitert.
 - Über den CCU-Historian können nun auch Datenpunkte in Geräten bzw. in der CCU geändert werden.
 - Neue Konfigurationsoption: *ccu.writeAccess* (Schreibzugriff auf die CCU erlauben)
 - Mehr Informationen auf der Details-Seite zu einem Datenpunkt.
 - Jeder Datenpunkt besitzt nun die zusätzlichen Eigenschaften "Inaktiv" und "Versteckt". Versteckte Datenpunkte werden in der Datenpunktübersicht nicht angezeigt. Inaktive Datenpunkte werden nicht archiviert. Eine neue Web-Seite zur Konfiguration dieser Eigenschaften wurde erstellt.
 - Datenpunkte, die Fehler generieren, werden auf inaktiv gesetzt.
 - Der Export von Zeitreihen, die Generierung von Trend-Diagrammen und die JSON-RPC Schnittstelle können durch Schlüsselwörter geschützt werden.
 - Auf der neuen Web-Seite *Werkzeuge* werden Funktionen bereitgestellt, um Historien von Datenpunkten zu leeren oder zu kopieren.
 - Auf der Web-Seite *Trend-Darstellung* werden zusätzliche Datumsformate akzeptiert.
 - Die Web-Seiten und die Datei *ccu-historian.exe* besitzen nun ein kleines Icon.
 - Auf der Web-Seite *Trend-Darstellung* wird nun die Länge des aktuellen Zeitbereichs angezeigt. Die Zeitbereichslänge kann nun durch Addition oder Subtraktion mehrerer Komponenten angegeben werden.
-

| | |
|---------------|---|
| V0.7.0 | <ul style="list-style-type: none"> Die vom CCU-Historian verwendeten Netzwerk-Ports können nun alle angepasst werden. Neue Konfigurationsoptionen: <i>database.webPort</i>, <i>database.tcpPort</i>, <i>database.pgPort</i>, <i>ccu.historianRpcPort</i> Abschnitt 7.5 zum Handbuch hinzugefügt. Speicher- und Performance-Optimierungen bei der Abfrage und Darstellung von großen Datenmengen. Aktualisierung des <i>Groovy</i>-Compilers auf die Version 2.0 und der <i>H2 Database Engine</i> auf die Version 1.3.170. Neue Abschnitte 4.5 <i>Firewall-Einstellungen</i> zum Handbuch hinzugefügt. Umstellung auf das BIN-RPC-Protokoll für die Geräteschnittstellen. Ab dieser Version wird z.B. CUXD nativ unterstützt. Multi-CCU-Betrieb für die gleichzeitige Archivierung von bis zu 10 Zentralen. Neue Konfigurationsoption: <i>logSystem.binRpcLevel</i> (zusätzlicher Filter für die BINRPC-Schnittstelle) Der Tabellenname für die Historie eines Datenpunktes wird nun wie folgt gebildet: D_Schnittstellename_Kanalname_Datenpunkt für numerische Zeitreihen, bzw. mit dem Prefix C_, wenn der Wert des Datenpunktes als Zeichenkette gespeichert wird. Die Einbeziehung des Schnittstellennamens ist nötig, damit bei einem Multi-CCU-Betrieb die Tabellennamen eindeutig sind. Neuen Abschnitt 4.2 <i>Konfiguration der angeschlossenen Geräte</i> hinzugefügt. Aktualisierung der eingebetteten JFreeChart-Bibliothek auf die Version 1.0.17. Kleinbuchstaben im Dateinamen werden nun beim CSV-Export (csv.gy) zugelassen. Über die JSON-RPC Schnittstelle werden die Eigenschaften eines Datenpunktes anders strukturiert zurückgegeben (s.a. Abschnitt 7.2). |
| V0.7.1 | <ul style="list-style-type: none"> Abschnitt 4.3 Firewall-Einstellungen korrigiert Fehler bei der Behandlung der Minimum- und Maximum-Eigenschaften von Zeichenkettendatenpunkten behoben. |
| V0.7.2 | <ul style="list-style-type: none"> Systemvariablen, die in Skripten mit <code>sysvar.Variable(newValue)</code> auf einen neuen Wert gesetzt werden, ändern nicht ihren Zeitstempel. Dieser Umstand wird nun vom CCU-Historian erkannt und der falsche Zeitstempel durch die aktuelle Uhrzeit ersetzt. (Hinweis: Systemvariablen sollten immer mit der Methode <code>sysvar.State(newValue)</code> auf einen neuen Wert gesetzt werden.) |
| V0.7.3 | <ul style="list-style-type: none"> Fehler bei der Behandlung der DefaultValue-Eigenschaft von Zeichenkettendatenpunkten behoben. |
| V0.7.4 | <ul style="list-style-type: none"> Die Trend-Darstellung von Datenpunkten vom Typ ACTION (z.B. Tastendrucke) wurde angepasst. Zum Zeitpunkt der Aktion wird nun ein Symbol gezeichnet, eine Trend-Linie wird nicht mehr gezeichnet. Abschnitt 6.4 des Handbuchs wurde entsprechend aktualisiert. Einige verwendete Bibliotheken/Werkzeuge wurden aktualisiert (z.B. Datenbank, Groovy-Compiler). In einem konfigurierbaren Zeitintervall können nun automatisiert Backups der Datenbank erstellt werden (s.a. Abschnitt 4.3). Die Funktionalität wird mit der neuen Option <i>database.backup</i> konfiguriert. |
| V0.7.5 | <ul style="list-style-type: none"> Verbesserte Behandlung von Systemvariablen mit ungültigen Zeitstempeln. Zusätzliche Zeitüberwachung aller BIN-RPC-Anfragen an die CCU. |
| V0.7.6 | <ul style="list-style-type: none"> Umstellung auf <i>Groovy</i> v2.3 und <i>Java 7</i>, Update der <i>H2 Database Engine</i> auf v1.3.176 Bei einer Neuinstallation (ohne vorhandene Datenbank) wurde die Tabelle <i>DATA_POINTS</i> nicht angelegt. |

V1.0.0

- Neue Konfigurationsoption *devices.device<G-Nr>.timeout* für Geräte mit BIN-RPC-Schnittstellen.
 - Anzahl der Datenbankschreibzugriffe durch Optimierungen vermindert.
 - Für jeden Datenpunkt kann eine Vorverarbeitung aller Wertaktualisierungen aktiviert werden. Zurzeit stehen folgende Algorithmen zur Auswahl: Nur Wertaktualisierungen, die sich mindestens um einen bestimmten Betrag ändern, sollen in der Datenbank abgelegt werden (Delta Kompression). Nur Wertaktualisierungen, die einen zeitlichen Mindestabstand zur vorhergehenden Aktualisierung besitzen, sollen abgelegt werden (Zeitliche Kompression).
 - Mit der neuen Konfigurationsoption *webServer.menuLinks* können eigene Web-Verweise (Links) zum Menü der Web-Applikation des CCU-Historians hinzugefügt werden.
 - Dokumentiertes Beispiel für eine eigene Web-Seite hinzugefügt: Vorjahresvergleich
 - Abschnitt 2 *Zusätzliche Funktionalität* zum Handbuch hinzugefügt.
 - Abschnitt 7.1.4 *Verwaltung Datenpunkte* ergänzt.
 - Abschnitt 2.3 *Web-Server* und 4.5 *Eigene Web-Seiten* hinzugefügt.
 - Bei der Generierung von Trend-Diagrammen werden nun Kurvengruppen unterstützt. Jede Kurvengruppe wird in einem eigenen Zeichenbereich dargestellt. Die einzelnen Kurvengruppen werden dann übereinander mit einer gemeinsamen Zeitachse dargestellt. Die Gruppenzuordnung eines Datenpunktes erfolgt mit dem neuen URL-Parameter **g**, die Höhen der Zeichenbereiche der einzelnen Gruppen können über den URL-Parameter **gh** gesetzt werden (s.a. Abschnitt 7.4).
 - Zwischenspeicherung von Wertänderungen im Arbeitsspeicher (s.a. Abschnitt 2.2)
 - Die Serien- und Kanalnummern werden nun auf der Web-Seite *Verwaltung Datenpunkte* angezeigt und können durchsucht werden.
 - In den Log-Meldungen auf der Konsole wird das Datum der Meldung ebenfalls angezeigt.
 - Einfachere Konfiguration der Trend-Designs (s.a. Abschnitt 7.4.1)
 - Der CCU-Historian unterstützt nun verschiedene Kommandozeilenparameter um bereits Einstellungen vor dem Lesen der Konfigurationsdatei zu setzen und verschiedene Wartungsarbeiten durchzuführen (s.a. Abschnitt 5.1).
 - Über Kommandozeilenparameter können verschiedene Wartungsarbeiten durchgeführt werden (s.a. Abschnitt 5.1): Aufräumen der Datenbank, Kompaktieren der Datenbank, Neuberechnen von komprimierten Zeitreihen, Speichern der kompletten Datenbank als SQL-Skript und Ausführen von SQL-Skripten
 - Zeichenkodierung der Web-Seiten korrigiert, damit diese auch auf einem Linux-System richtig ausgeliefert werden.
 - Bei der JSON-RPC-Schnittstelle, dem CSV-Export und der Generierung von Trend-Diagrammen kann nun auch der ISE-Name (z.B. "BidCos-RF.ABC0123456;2.STATE") anstatt der Historian-ID zur Identifikation eines Datenpunktes verwendet werden.
 - Fehler bei der Erkundung der Datenpunktnamen aus der Logikschicht der CCU behoben.
 - Eine Java-Laufzeitumgebung in der Version 8 wird nun benötigt.
 - Die Paketierungsart des CCU-Historians wurde geändert, wodurch er jetzt schneller startet.
-

-
- Die Fehlererholung wurde angepasst, sodass keine Folgefehler mehr verursacht werden.
 - Im Handbuch wurde der Abschnitt 4.3 "Fehlersuche in der Konfigurationsdatei" hinzugefügt.
 - Räume und Gewerke werden nun aus der CCU ausgelesen und in der Datenbank abgelegt. In der Datenpunktübersicht der Web-Oberfläche werden sie als zusätzliche Spalten angezeigt.
 - Dokumentation um die Einbindung eigener Web-Seiten erweitert. (Abschnitte 2.3, 4.5 und 7.5)
-