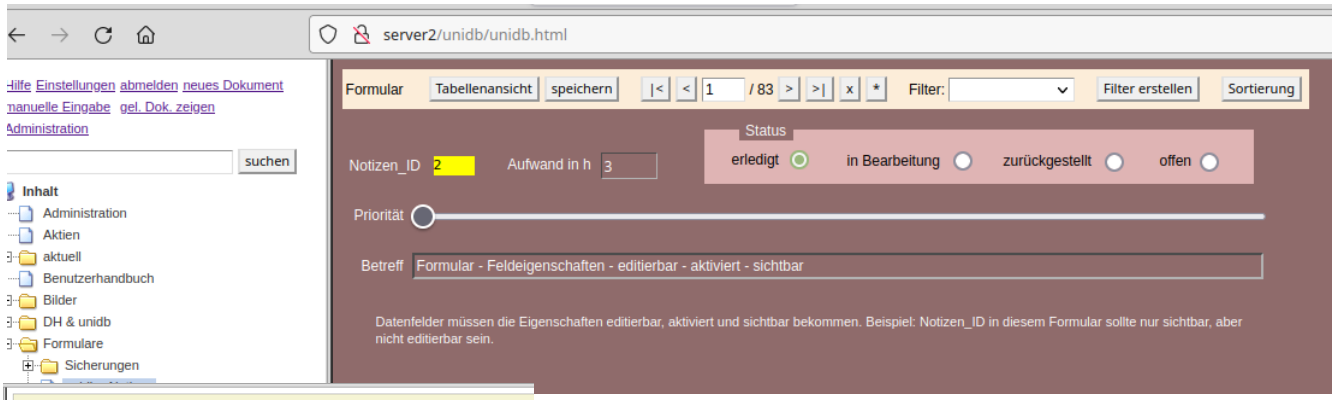
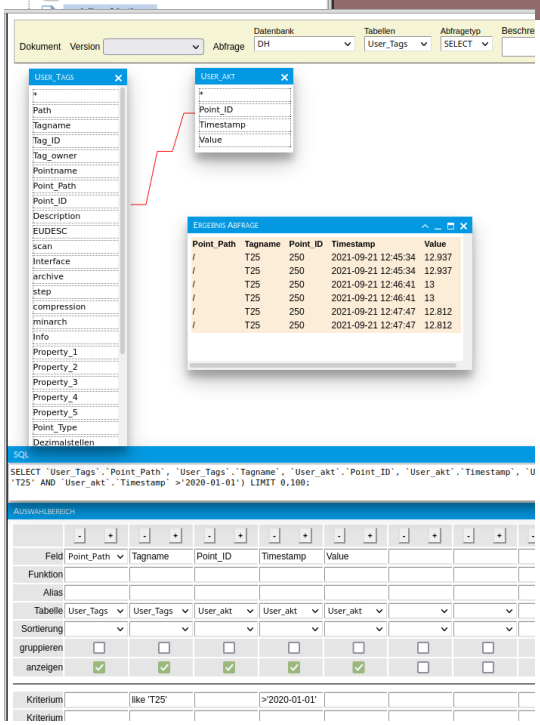
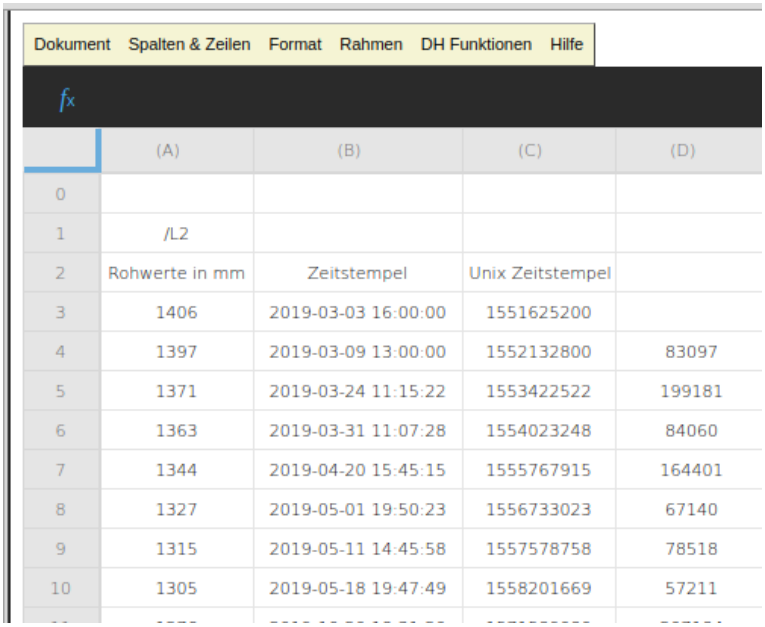
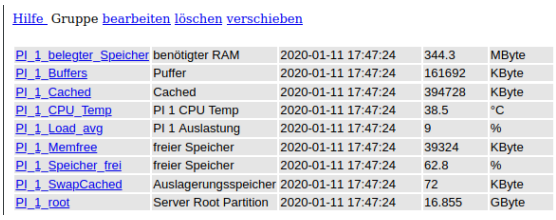
# unidb und DH Administrationshandbuch

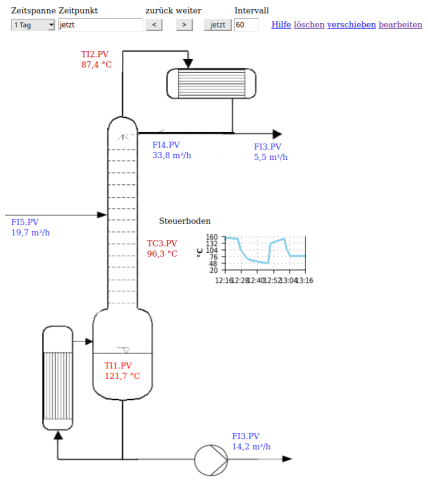
**Version 1.0**

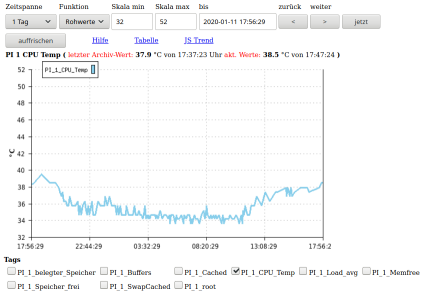


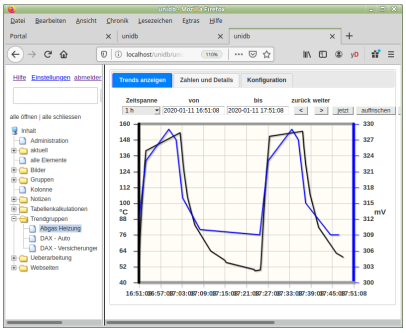



# Inhaltsverzeichnis


Begriffe und Abkürzungen ..... 2

Verwendete Formatierungen ..... 2

1. allgemein ..... 3
   1.1. Archiv ..... 3
   1.2. Kollektiv und Ausfallsicherheit ..... 3
   1.3. Planung ..... 4
   1.4. Benutzer ..... 5
      1.4.1. Pfade und Benutzerrechte ..... 5
      1.4.2. Benutzer der MariaDB und unidb ..... 6
      1.4.3. Lese- und Schreibrechte ..... 6

2. unidb ..... 6
   2.1. Benutzerverwaltung ..... 6
   2.2. Einstellungen ..... 9
   2.3. Kollektiv ..... 9
   2.4. Pufferstände ..... 10
   2.5. Themen bearbeiten ..... 11
   2.6. MariaDB Editor ..... 12

3. DH ..... 13
   3.1. Point- und Tagkonfiguration ..... 13
   3.2. calc Konfiguration ..... 17
   3.3. Schnittstellen ..... 20
      3.3.1. Programmablauf ..... 20
      3.3.2. Schnittstellen starten & stoppen ..... 22
      3.3.3. Konfiguration ..... 23
      3.3.4. Points zur Überwachung der Schnittstellen ..... 24
      3.3.5. Störungen und deren Behebung ..... 25
      3.3.6. Standardschnittstellen ..... 25
   3.4. Kollektiv ..... 26
      3.4.1. Schnittstellen im Kollektiv ..... 27
      3.4.2. Schnittstellenrechner zum Kollektiv hinzufügen oder entfernen. ..... 28
      3.4.3. Server zum Kollektiv hinzufügen oder entfernen. ..... 29
   3.5. weitere Formulare im Admin - Bereich ..... 29
      3.5.1. Einstellungen DH ..... 29
      3.5.2. Geräte Einstellungen ..... 30
      3.5.3. Bausteine verwalten ..... 30
      3.5.4. Meldungen ..... 30
      3.5.5. Pufferstände ..... 30
      3.5.6. Log ..... 31

4. DH_SMT ..... 31
   4.1. Verbindungen ..... 31
   4.2. Archiveditor ..... 31
   4.3. Rekalkulator ..... 34
   4.4. Export ..... 34
   4.5. Archive bereinigen. ..... 34

# Begriffe und Abkürzungen

| | |
|---|---|
| DataHistorian | Datenbankanwendung, welche fortlaufend Daten im Format Datenpunkt – Zeitstempel – Wert sammelt und diese in aufbereiteter Form wieder zur Verfügung stellt. |
| DH | Abkürzung für DataHistorian |
| TimeSeriesDatabase | Eine alternative Bezeichnung für einen DataHistorian. |
| DB | Abkürzung für Datenbank. |
| Tag | Datenpunkt in einem DataHistorian |
| Point | Andere Bezeichnung für einen Tag / Datenpunkt. |
| Point_ID | Eindeutige Nummer eines Tags, vergleichbar einer Seriennummer. |
| WYSIWYG | **W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et (Was Du siehst, ist das was Du bekommst.) Man kann zwei Arten von HTML Editoren unterscheiden. Die einfachste Art ist die, bei der man den HTML Code sieht und auch selbst schreiben muss. Die zweite Art ist die Art, die für den Anwender einfacher ist. Der HTML Code bleibt verborgen und stattdessen sieht man den fertig formatierten Text (WYSIWYG). |

# Verwendete Formatierungen

| | |
|---|---|
| **fette Schrift** | Name eines Steuerelementes (Schalter, Textfeld, ...) |
| *kursiv* | Namen von Eigenschaften o.ä. |
| farbig hinterlegt | Einem farbig hinterlegten Text sollte man etwas mehr Aufmerksamkeit widmen. |

# 1. allgemein

Die Administration erfolgt fast ausschließlich über die Administrationsseite. Diese kann entweder direkt aufgerufen werden, indem Sie */admin* an die Webadresse der unidb anhängen oder aber über den Link **Administration**. Der Link befindet sich rechts neben dem Logo. Sollten Sie den Link dort nicht finden, dann haben Sie auch keine Administrationsrechte. Ein direkter Aufruf der Admin Seite zeigt Ihnen in diesem Fall nur das Anmeldeformular.
In Ausnahmefällen und wenn Sie genau wissen, was Sie tun, können Sie die beiden Datenbanken (DH und unidb) auch mit einem allgemeinen Werkzeug für die Administration von Datenbanken verwenden. Typische Vertreter solcher Anwendungen sind phpMyAdmin, DBeaver oder DBGate. Bedenken Sie dabei aber, dass Sie in einem Kollektiv jede Änderung auf jedem Server des Kollektives separat durchführen müssen.

## 1.1. Archiv

Das Datenarchiv befindet sich gewöhnlich im Ordner */var/lib/DH*. Die Daten eines jeden Points stehen in Dateien, deren Namen der Point_ID des Points entspricht. Dabei enthält jede Datei die Werte eines Monats. Die Dateien jede Datenart liegen jeweils in einem eigenen Ordner. Somit ergibt sich folgende Ordnerhierarchie unter /var/lib/DH:

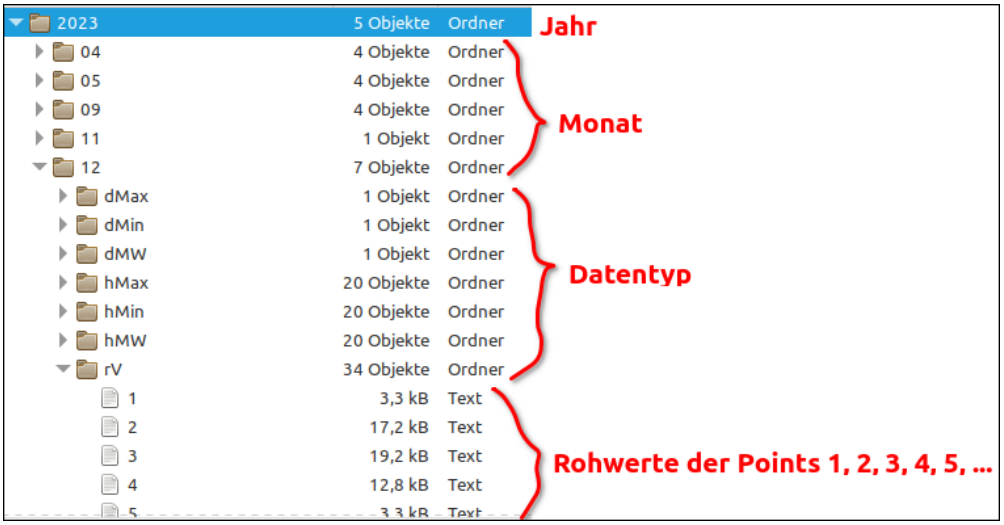


Abb. 1: Hierarchischer Aufbau des Archivs.

## 1.2. Kollektiv und Ausfallsicherheit

Sowohl die unidb als auch der DH arbeiten immer in einem Kollektiv. Im einfachsten Fall besteht das Kollektiv aus nur einem Rechner. Das klingt zunächst nicht sinnvoll, spielt aber bei der Pufferung von Daten eine Rolle. Gewöhnlich betreibt man aber ein Kollektiv mit mehreren Rechnern. Dabei handelt es sich um Server (DH und / oder unidb) und um Schnittstellenrechner. Auf Letzteren laufen lediglich Schnittstellen für den DH.

**<u>DH - Kollektiv:</u>**
Eine Schnittstelle für den DH liefert die Daten parallel an alle Server des Kollektivs. Ist ein Server nicht erreichbar, werden die Daten für diesen Server auf dem Schnittstellenrechner lokal gespeichert. Sobald der Server wieder erreichbar ist, werden die zwischengespeicherten Daten an den Server ausgeliefert.
Alle Änderungen, die Sie auf der Administrationsseite durchführen, werden auf alle Server des Kollektivs übertragen. Ist ein Server nicht erreichbar, dann wird die geänderte Konfiguration nachträglich auf den betreffenden Server geschrieben. Sie können zwar somit Änderungen vornehmen, wenn ein Server nicht erreichbar ist, aber es ist immer sicherer, wenn man das vermeidet.
Parallel arbeitende Server im Kollektiv sorgen für eine relativ hohe Ausfallsicherheit. Um diese Sicherheit auch bei den Schnittstellenrechnern zu erreichen, gibt es drei Möglichkeiten:

1. *cold backup*: Für jeden Schnittstellenrechner existiert ein zweiter Rechner, welcher die gleiche Konfiguration aufweist, wie der erste Schnittstellenrechner. Er wird manuell gestartet, wenn der erste Schnittstellenrechner ausfällt.
2. *warm backup*: Der zweite Schnittstellenrechner läuft parallel zum ersten Rechner. Ein Script auf

diesem Rechner schaut ständig nach, ob der erste Rechner noch Daten liefert. Ist das nicht mehr der Fall, startet es die eigenen Schnittstellen.

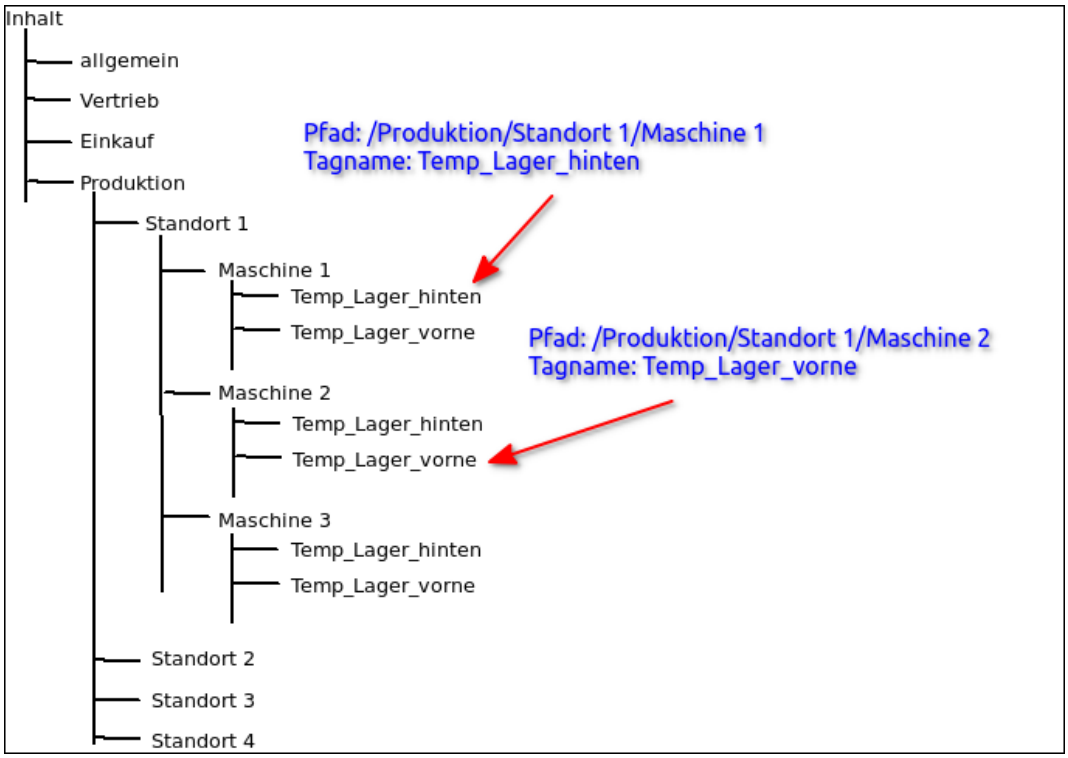
3. *hot backup*: Beide Schnittstellenrechner laufen parallel zueinander und liefern ständig Daten an die Server. Der Kompressor auf dem Server sorgt dafür, dass die Daten nicht doppelt ins Archiv gehen.

Wichtig: Wenn Sie die Methode 2 oder 3 anwenden, müssen Sie den Schnittstellen unterschiedliche Namen geben und auch jede Schnittstelle separat konfigurieren.

**unidb - Kollektiv:**
Alle Änderungen, die an den Dokumenten von den Benutzern durchgeführt werden, werden auf alle Mitglieder des Kollektivs übertragen. Ist ein Server nicht erreichbar, dann werden die Änderungen auf dem Server zwischengespeichert, auf dem die Änderung durchgeführt wurde. Sobald der betreffende Server wieder erreichbar ist, werden die Änderungen nachgetragen. Selbst wenn mehrere Server nicht erreichbar sind und an mehreren Servern Änderungen an den Dokumenten vorgenommen wurden, erfolgt der Abgleich, sobald ein Server wieder erreichbar ist. Alle Server aus dem Kollektiv schreiben ihre Änderungen in einen Eingangspuffer auf dem zuvor ausgefallenen Server. Sobald die Puffer (Ausgang) auf allen Mitgliedern des Kollektivs leer sind, beginnt der zuvor ausgefallene Server damit, die Änderungen in chronologischer Reihenfolge für sich aus dem Eingangspuffer zu übernehmen. Das bedeutet, dass die letzte Änderung eines Dokumentes gewinnt. Wurden also mindestens zwei Änderungen an einem Dokument auf unterschiedlichen Servern vorgenommen, dann wird die jüngste Änderung auf dem vormals ausgefallenen Server übernommen. Das ist nicht immer ganz korrekt, sollte aber sehr selten vorkommen.

## 1.3. Planung

Bevor Sie einen neuen Server oder ein neues Kollektiv aufsetzen, sollten Sie sich Gedanken um die Organisation machen. Es geht dabei um den Aufbau der Baumstruktur. Stellen Sie später fest, dass der Aufbau der Baumstruktur nicht so gelungen ist, wie Sie das gerne hätten, dann kommt entweder viel Arbeit auf Sie zu oder Sie müssen damit leben. Also, wer nicht gerne arbeitet, der muss viel nachdenken!
Sämtliche Berechtigungen der Benutzer hängen von der Baumstruktur ab. Auch bei der Vergabe der Namen für Points und Tags aus dem DH spielt die Baumstruktur eine große Rolle. Beispiel: Sie füttern den DH mit Messwerten von gleichartigen Maschinen an verschiedenen Orten. Alle Maschinen verfügen über einen Generator. An jedem Generator werden zwei Lagertemperaturen gemessen. Der Aufbau der Baumstruktur könnte dann etwa so aussehen:




oder

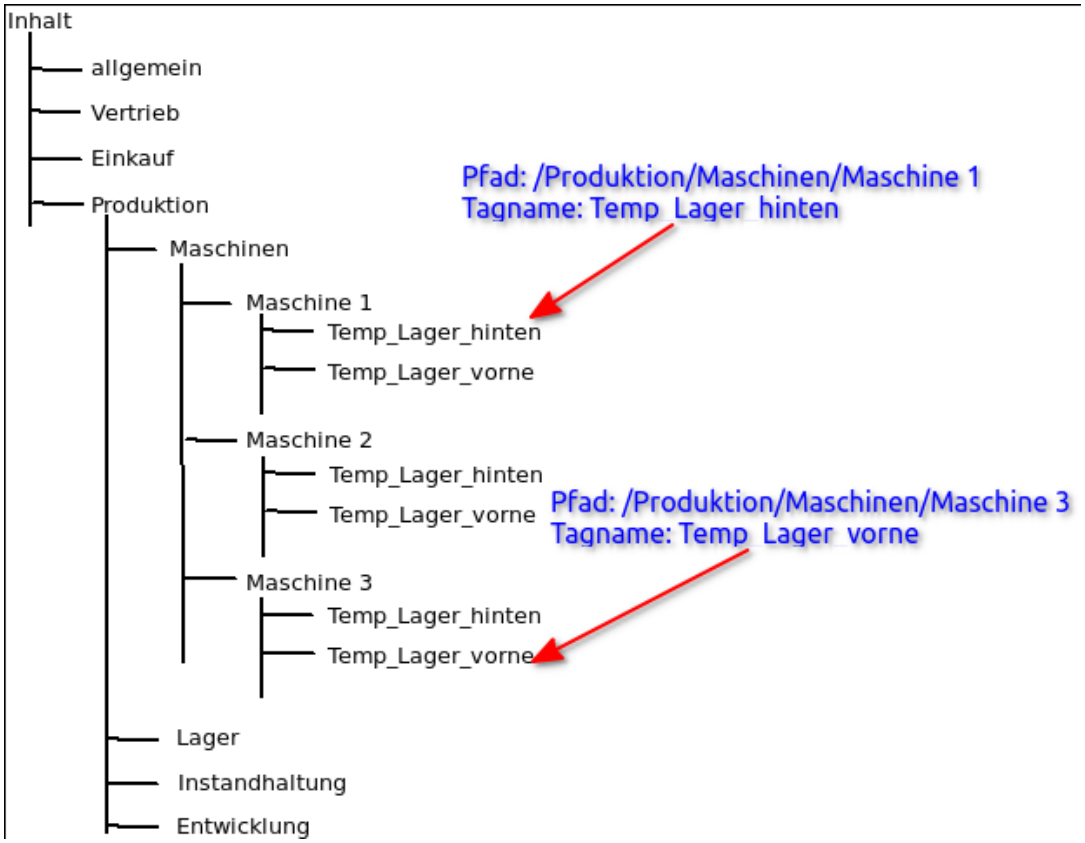


Abb. 2: Organisation der unidb und des DH

Wenn Sie später einmal die Baumstruktur ändern möchten, dann müssen Sie die Pfade der betroffenen Tags und Points ändern. Zusätzlich müssen Sie die Zugriffsrechte der Benutzer anpassen. Der Aufwand für die Umorganisation der Dokumente in der Baumstruktur ist hingegen meistens schnell erledigt.

## 1.4. Benutzer

### 1.4.1. Pfade und Benutzerrechte

Sowohl in der unidb, als auch beim DH ist alles ist hierarchisch geordnet. An welchem Punkt in der Hierarchie sich ein Dokument aus der unidb, ein Tag oder ein Point aus dem DH sich befindet, wird durch den zugeordneten Pfad bestimmt. Dabei kann ein Tag, ein Point oder ein Dokument nur einem Pfad zugeordnet sein. Die Kombination aus Pfad und Tag, Point oder Document ist immer eindeutig. Es können somit z.B. mehrere Tags mit dem gleichen Namen existieren, wenn sie unterschiedliche Pfade haben. Es handelt sich immer um eine 1:1 Beziehung. Ein Benutzer ist hingegen immer eindeutig. Es kann kein Benutzername doppelt vergeben werden. Allerdings können einem Benutzer mehrere Pfade zugeordnet werden. Hier haben wir also eine 1:n Beziehung.

Ein Beispiel macht den Grund hierfür deutlich:
Angenommen, Sie haben die Hierarchie der Pfade an die Struktur Ihrer Firma angepasst. Die Firma ist auf drei Standorte verteilt. Jeder Standort ist in mehrere Abteilungen unterteilt. Feder Standort verfügt über eine Versandabteilung.
Der Mitarbeiter Mustermann arbeitet im Versand am Standort 1, muss aber auch die Daten der Versandabteilungen der beiden anderen Standorte sehen können. Hier nun die Zuordnung der Pfade für Herrn Mustermann:

*/Firma allgemein* <-- Dieser Pfad ist jedem Mitarbeiter zugeordnet.
*/Benutzer/Mustermann* <-- Das persönliche Verzeichnis des Mitarbeiters.
*/Standort 1* <-- Alle Daten vom Standort 1.
*/Standort 2/Versand* <-- Versanddaten Standort 2.
*/Standort 3/Versand* <-- Versanddaten Standort 3.

### 1.4.2. Benutzer der MariaDB und unidb

Wird ein neuer Benutzer in der unidb angelegt, dann wird automatisch auch ein gleichnamiger neuer Benutzer für die MariaDB erzeugt. Der Benutzername in der MariaDB wird dabei komplett klein geschrieben.
Darüber hinaus existieren im Zusammenhang mit der unidb und dem DH zwei weitere Benutzer in der MariaDB. Dabei handelt es sich um die Benutzer *DH_readonly* und *DH_write*. *DH_readonly* wird von der Weboberfläche zum Lesen der Daten aus dem DH genutzt. *DH_write* kann auch Daten in den DH schreiben, nicht aber die Datenbankstruktur ändern.

### 1.4.3. Lese- und Schreibrechte

Jeder Benutzer der unidb kann grundsätzlich Daten in die Hierarchie der unidb schreiben oder auch Daten löschen. Dabei kann der Benutzer alle Daten aus seinen zugeordneten Pfaden lesen. Er kann auch in die zugeordneten Pfade schreiben. Löschen oder editieren kann er aber nur Dokumente, die er selbst erstellt hat. Ausgenommen hiervon sind Benutzer, welche als Administrator konfiguriert wurden. Diese können alle Dokumente, welche in den ihnen zugeordneten Pfaden liegen löschen oder editieren. Der mächtigste Benutzer ist somit ein Administrator, welchem der Pfad / zugeordnet wurde. Bei dem / handelt es sich um die Wurzel der ganzen Hierarchie.
Auch wenn ein Benutzer der unidb als Administrator konfiguriert wurde und damit möglicherweise Zugriff auf alle Points und Tags im DH hat, kann er keine Tags oder Points im DH anlegen, editieren oder löschen, da die unidb Benutzer in der MariaDB so konfiguriert sind, dass sie keine Schreibberechtigung auf die entsprechenden Tabellen im DH haben.

# 2. unidb

## 2.1. Benutzerverwaltung

Neue Benutzer können ein Benutzerkonto über den Link **Benutzerkonto einrichten** im Anmeldeformular der unidb beantragen.

Abb. 3: Anmeldeformular

Der neue Benutzer erscheint im Administrationsbereich der unidb unter dem Link **Benutzerverwaltung**.

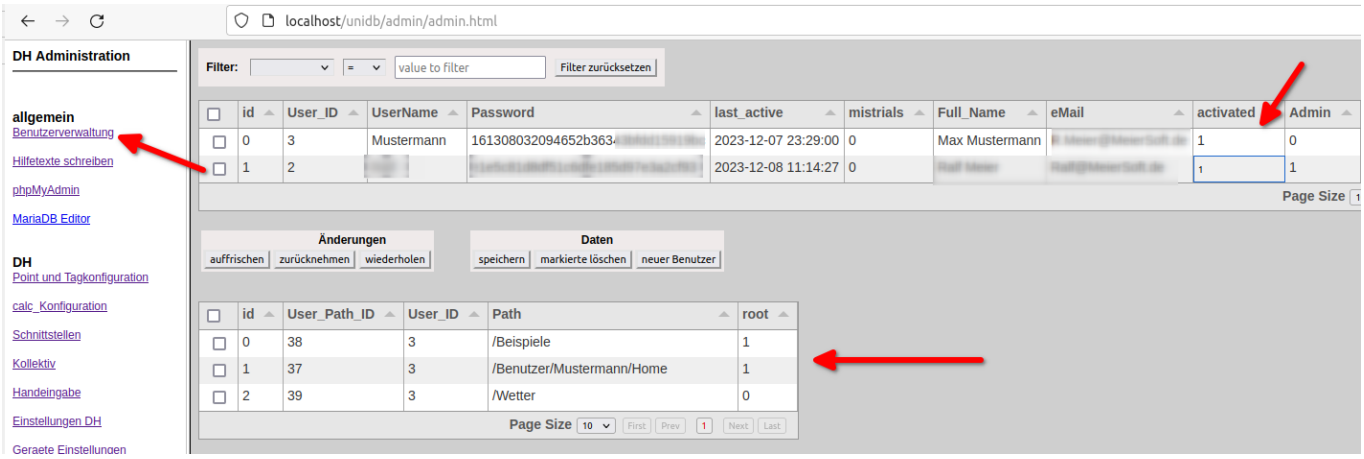


Abb. 4: Neuer Benutzer in der Benutzertabelle.

Gleichzeitig wird eine eMail zur Info über den neuen Benutzer an den Administrator gesendet.

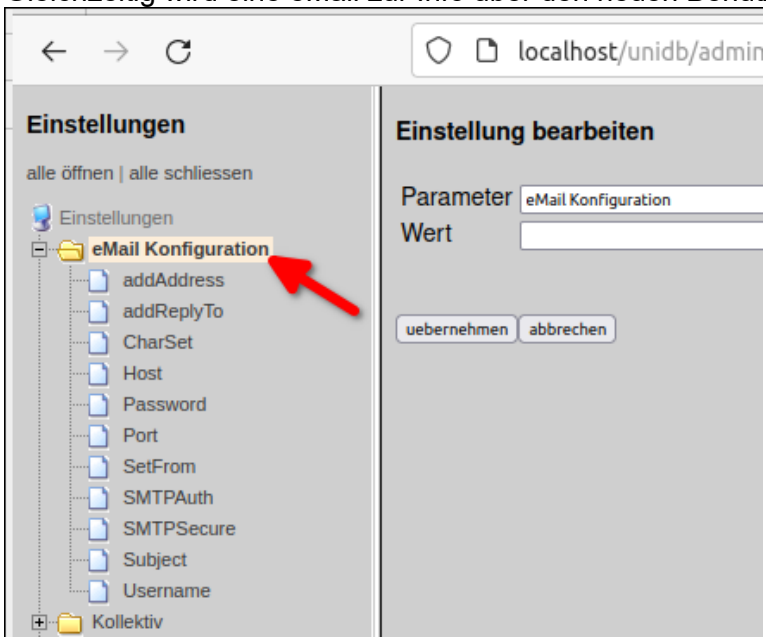


Abb. 5: Konfiguration des eMail Versands an den Administrator

Die Aufgabe des Admins ist es nun, dem neuen Benutzer mindestens einen Pfad zuzuweisen und das Konto anschließend freizugeben. Die Freigabe erfolgt in der Benutzertabelle in der Spalte **activated**. Wird der Wert in dieser Spalte auf 1 gesetzt, dann kann der Benutzer sich anmelden. Vergessen Sie bitte nicht, den Benutzer über die Aktivierung zu informieren.
In Abb. 4 wurden dem neuen Benutzer „Max Mustermann“ einige Pfade zugewiesen. Die Zuweisung der üblichen Pfade eines Benutzers kann automatisiert werden. Welche Pfade zugewiesen werden, ist in den Einstellungen unter dem Punkt *Vorlagen neuer Benutzer* festgelegt. Jeder Eintrag unter diesem Punkt steht für einen automatisch zuzuweisenden Pfad. Enthält ein Pfad den Namen des neuen Benutzers, wird dieser durch die Variable *$Benutzer* repräsentiert. Siehe hierzu Abb. 6.

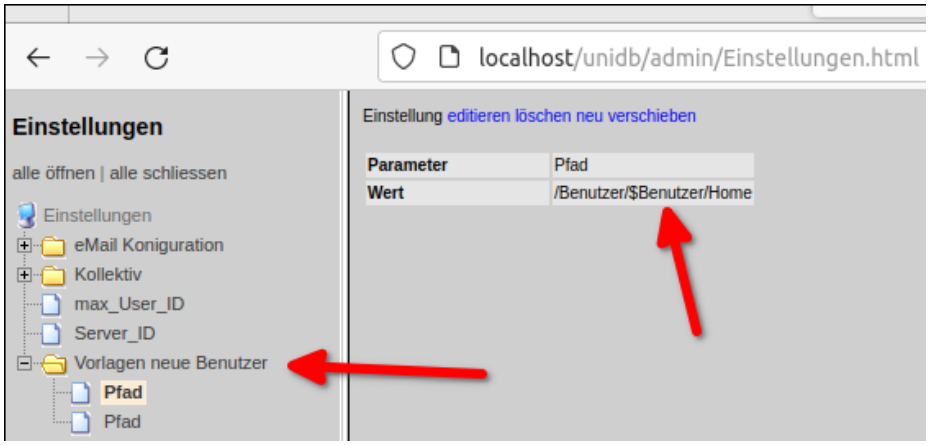


Abb. 6: Einstellungen für die automatische Zuweisung von Pfaden für neue Benutzer.

Das Formular **Benutzerverwaltung**:
Das Formular enthält zwei Tabellen. Die obere Tabelle zeigt die vorhandenen Benutzer und die untere Tabelle listet die Pfade auf, die dem jeweils ausgewählten Benutzer aus der oberen Tabelle zugeordnet sind.
Für beide Tabellen gilt: Ändert man den Inhalt eines Feldes, dann wird das Feld farbig hinterlegt. Die Änderung wird erst dann geschrieben, wenn man auf den Schalter **„speichern**“ unterhalb der Tabelle klickt.
Die erste Spalte **id** dient in den Tabellen nur der Verwaltung innerhalb des Formulars. Für sie gibt es kein entsprechendes Feld in der Datenbank.

Tabelle Benutzer:
Das Feld **mistrials** enthält die Anzahl der fehlerhaften Anmeldeversuche. Wenn die Anmeldung 5x fehlgeschlagen ist, dann wird das Benutzerkonto gesperrt, indem das Feld **activated** auf 0 gesetzt wird. Bei einer erfolgreichen Anmeldung wird das Feld **mistrials** auf 0 gesetzt. Im Feld **Admin** wird festgelegt, ob der Benutzer für seine Pfade Administrationsrechte bekommt. Die Standardeinstellung hierfür ist 0, also keine Administrationsrechte.

Tabelle Pfade:
Die Bedeutung der Spalten in dieser Tabelle sollte eigentlich klar sein. Nur die letzte Spalte **root** bedarf einer Erklärung. Der Name root kann man hier wörtlich nehmen. Für jede Zeile, in der *root = 1* steht, erscheint bei dem Benutzer eine eigene Baumstruktur.

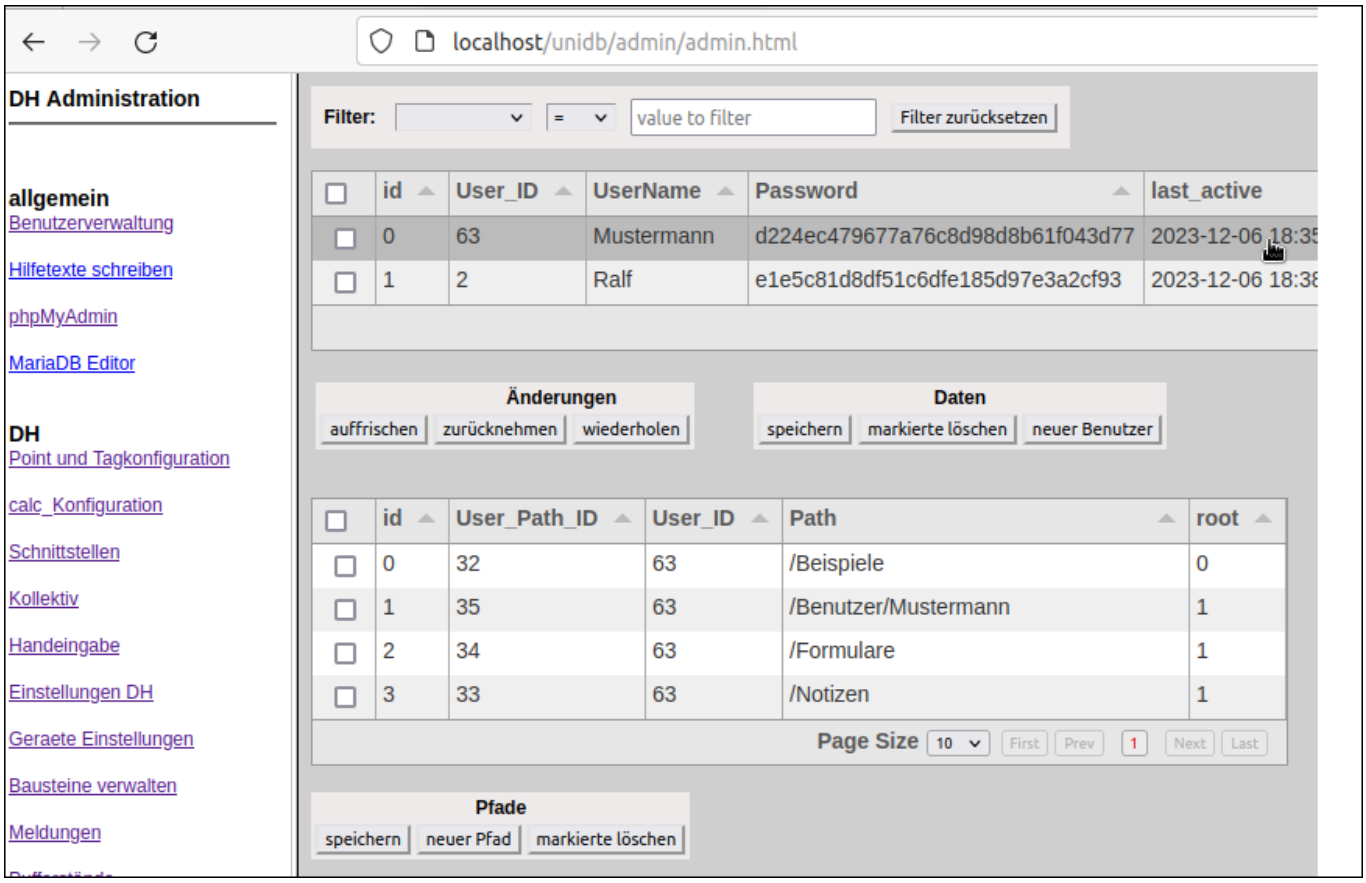


Abb. 7: Pfade des Benutzers

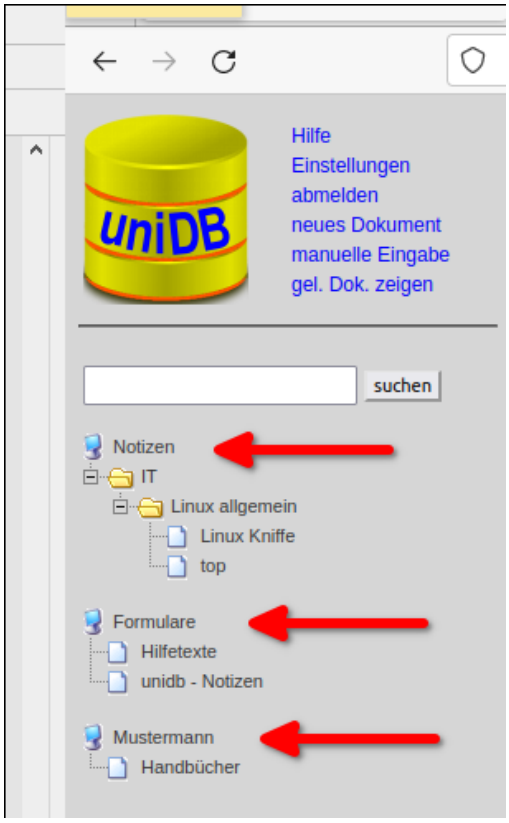


Abb. 8: root - Pfade des Benutzers

Der Pfad */Beispiele* ist in Abb. 4 mit *root = 0* markiert und erscheint daher nicht im Navigationsbereich des Benutzers.

**Achtung:**
Wenn Sie einem Benutzer einen Pfad zuweisen, dann muss dieser auch wirklich in der Hierarchie vorhanden sein. Ansonsten kann das zu einem Eintrag ohne Eltern_ID führen und die Baumstruktur wird dem Administrator nicht mehr korrekt angezeigt. Pfade, die beim Erstellen eines neuen Benutzkontos automatisch zugewiesen werden, werden automatisch angelegt, falls sie noch nicht existieren.

## 2.2. Einstellungen

Der Link *Einstellungen* öffnet einen neuen Tab im Browser, in dem auf der linken Seite alle Einstellmöglichkeiten in einer Baumstruktur aufgelistet werden. Dieses Formular sollte nur verwendet werden, wenn es sonst keine Möglichkeit gibt, eine Einstellung zu ändern. Das ist z.B. der Fall, wenn Sie die Zugangsdaten für die eMails an den Administrator ändern möchten. Auch die Änderung der automatisch zugewiesenen Pfade für neue Benutzer können nur hier gemacht werden.

**Achtung:**
Alle Änderungen werden auf alle Server des Kollektivs geschrieben. Ändern Sie den Wert der *Server_ID*, dann haben alle Server die gleiche *Server_ID*. Dies führt zu einer korrupten Baumstruktur.

## 2.3. Kollektiv

Der Zweck und die Funktion eines Kollektives wurde bereits in Kapitel 1.1 beschrieben. Hier geht es um die Konfiguration.

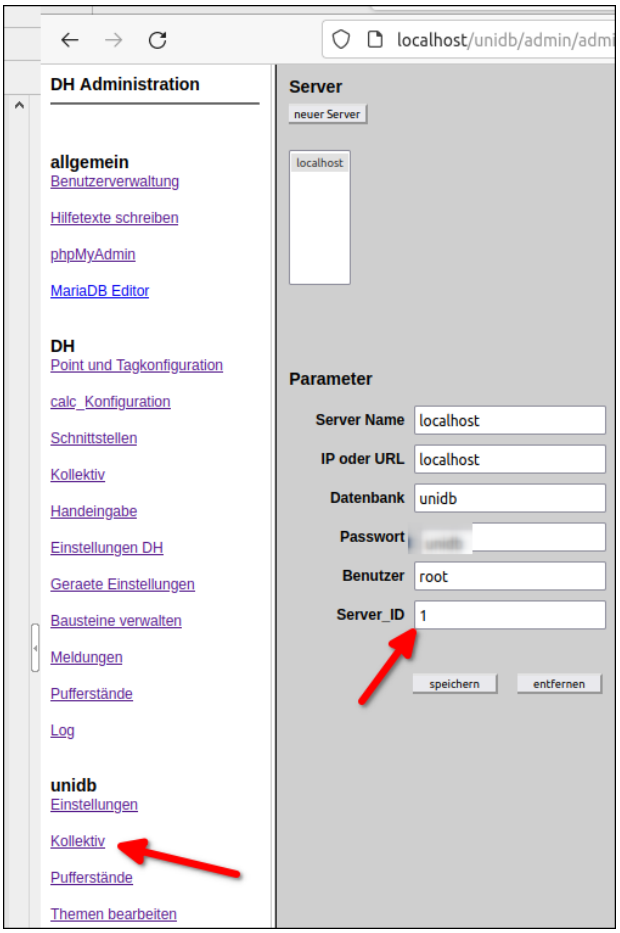


Abb. 9: Konfiguration des Kollektives für die unidb.

Zu dem Formular gibt es eigentlich nicht viel zu erklären. Nur das Feld **Server_ID** könnte etwas verwirren, da man im Formular *Einstellungen* einen gleichnamigen Eintrag findet. Unter *Einstellungen* ist die Server_ID des Servers zu sehen, von dem die Einstellungen geladen wurden. Das ist normalerweise der Server, auf dem Sie gerade angemeldet sind. Diesen Wert bitte nur mit einem Werkzeug wie phpMyAdmin auf diesem Server einstellen. Es handelt sich dabei um ein Feld in der Tabelle *Einstellungen* der unidb.
Die *Server_ID*, die Sie im Formular aus Abb. 9 sehen, sollte die gleiche ID sein, die Sie mit phpMyAdmin o.ä. in die Datenbank des Servers eingetragen hatten. Der Eintrag hier dient nur dazu, den Kollektivmitgliedern mitzuteilen, welcher Server welche *Server_ID* hat.

## 2.4. Pufferstände

Idealerweise sind alle Puffer leer. Sollte sich irgendwann mal ein Puffer nicht vollständig entleert haben, dann wird Ihnen im Formular Pufferstände zunächst angezeigt, wie viele Einträge in den Puffern der Server sind. Ein Klick auf einen Servernamen öffnet eine Tabelle, welche die Einträge im Puffer dieses Servers anzeigt. Hier lassen sich die Einträge bearbeiten oder löschen.

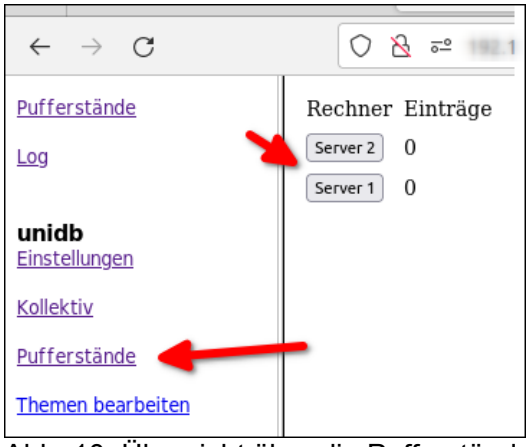


Abb. 10: Übersicht über die Pufferstände.

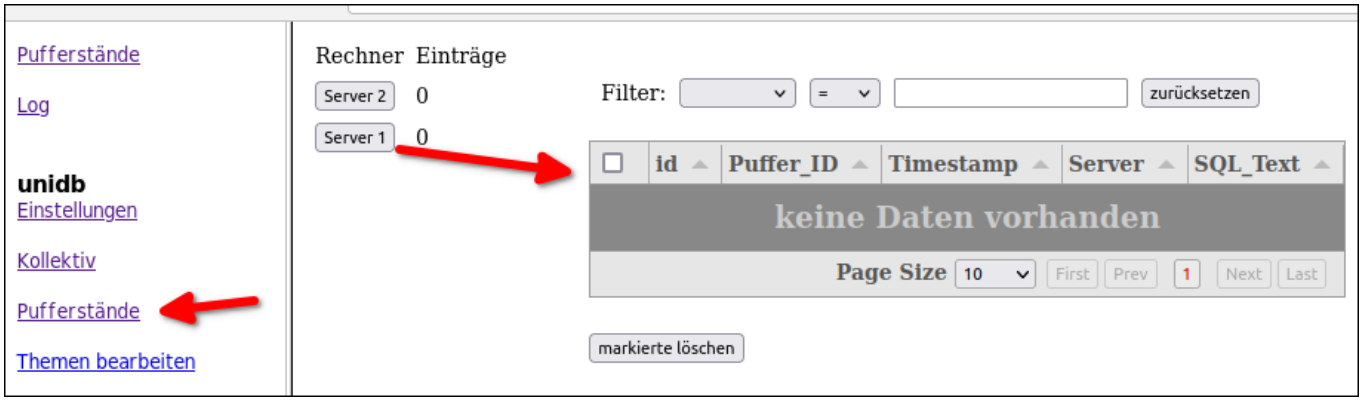


Abb. 11: Tabelle der Einträge im Puffer des angeklickten Servers.

Mögliche Störungen:
Sollten Änderungen an Dokumenten nicht auf andere Server des Kollektives übertragen werden, dann liegt das wahrscheinlich daran, dass nicht alle Puffer leer sind und sich deshalb die Eingangspuffer der Kollektivmitglieder nicht leeren.
Abhilfe:
Schauen Sie nach, ob irgend ein Server noch Einträge im Puffer stehen hat. Wenn ja, dann überprüfen Sie bitte die Inhalte der Spalte SQL_Text. Sollte hier ein fehlerhafter SQL Test stehen, dann bitte korrigieren oder die betreffenden Zeilen löschen. Die Eingangspuffer der Kollektivmitglieder sollten sich jetzt innerhalb kurzer Zeit leeren. Sollte das nicht der Fall sein, dann kontrollieren Sie bitte auf jedem Kollektivmitglied, ob dort die Scripte **unidb_Abgleich.py** und **unidb_watchdog.py** laufen. Das erledigen Sie im Terminal mit *ps -e|grep unidb*. Wird eines der oben genannten Scripte hier nicht aufgelistet, dann starten Sie das fehlende Script im Terminal. Die Scripte sollten im Verzeichnis */opt/unidb* liegen. Schreiben Sie also im Terminal */opt/unidb/unidb_Scriptname.py &*. Das & Zeichen dient dazu, das Script im Hintergrund auszuführen.

Hinweis:
Es ist nicht nur bequem, sondern auch sicherer, wenn Sie die Scripte über einen Eintrag im Verzeichnis */etc/init.d* automatisch starten. Ansonsten vergisst man das schon mal beim Start des Rechners.

## 2.5. Themen bearbeiten

Jeder Benutzer kann in seinen persönlichen Einstellungen ein Thema für die Gestaltung der Benutzeroberfläche auswählen.
Hier können Sie die Gestaltung der Themen ändern oder ein neues Thema als Kopie des aktuell ausgewählten Themas erzeugen.

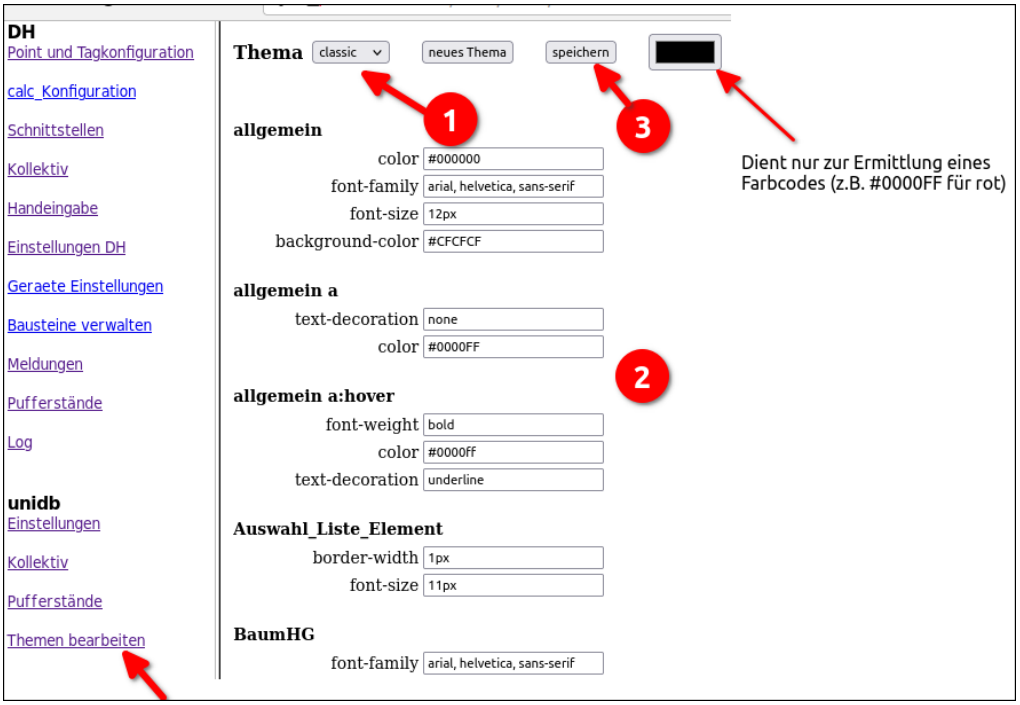


Abb. 12: Themen editieren.

Wählen Sie ein Thema aus dem Kombifeld (1) aus. Ändern Sie was Sie ändern möchten (2) und speichern Sie die Änderungen schließlich ab (3).
Die schwarze Schaltfläche oben rechts dient lediglich dazu, den gewünschten Farbton für ein Feld auszuwählen, welches einen Farbcode erwartet (z.B. *color* oder *background-color*).

## 2.6. MariaDB Editor

Bei den Dokumenten in der unidb handelt es sich um Datensätze aus der Tabelle *Baum*. Jeder Datensatz enthält ein Feld mit dem Namen *Inhalt*. Dieses Feld enthält weitere Felder, welche wie Felder einer NOSQL - Datenbank behandelt werden. Die Dokumenttypen unterscheiden sich lediglich durch die dynamischen Felder innerhalb des Feldes *Inhalt*. Die meisten SQL Editoren können die „normalen“ Datenbankfelder problemlos bearbeiten. Diese Felder werden daher in diesem Formular nur angezeigt. Der Editor ist gebaut worden, um die dynamischen Felder auswählbar zu machen, damit sie editiert werden können.
Wird ein Dokument der unidb editiert, dann wird die bisherige Version des Dokumentes in der Tabelle *Baumhistorie* abgelegt. Auch diese Dokumente lassen sich mit dem Formular bearbeiten.

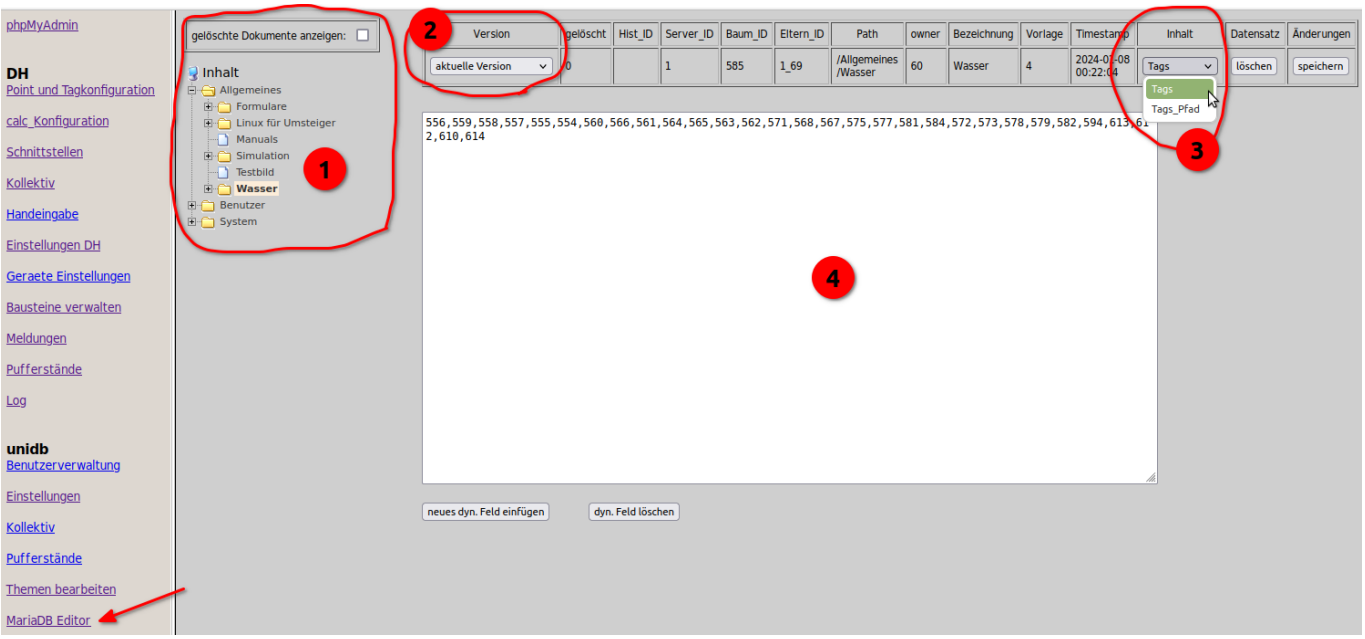


Abb. 13: Formular **MariaDB Editor**

Bereich 1 in Abb. 13 zeigt die Baumstruktur der Dokumente, so wie Sie sie auch von der normalen Benutzeroberfläche her kennen. Bereich 2 enthält ein Listenfeld, über das man entweder die aktuelle Version des Dokumentes aus der Tabelle *Baum* oder eine archivierte Version aus der Tabelle *Baumhistorie* auswählen kann. Gewöhnlich möchte man eher die aktuelle Version eines Dokumentes bearbeiten. Daher ist diese auch vorausgewählt. Im Bereich 3 befindet sich ein Listenfeld, welches alle dynamischen Felder des Dokumentes auswählbar macht. In Abb. 13 wurde das Dokument *Wasser* ausgewählt. Dabei handelt es sich um ein Dokument des Typs *Gruppe*. Dieser Dokumenttyp enthält nur zwei dynamische Felder. Das Feld Tags enthält sämtliche Tag_IDs der Tags aus der Gruppe, welche jeweils durch ein Komma voneinander getrennt sind. Das Feld Tags_Pfad enthält den Pfad zu den Tags. Näheres zum Dokumenttyp *Gruppe* entnehmen Sie bitte dem Entwicklerhandbuch. Im Bereich 4 können Sie nun Ihre Änderungen vornehmen. Bitte vergessen Sie nicht, die Änderungen mit dem Schalter ganz oben rechts im Formular zu speichern.

**Hinweis:**
Ebenso, wie Änderungen auf dem „normalen“ Weg über den Link *bearbeiten* im Dokument in der Benutzeransicht, wird auch hier die bisherige Version des Dokumentes ins Archiv geschrieben und eine neue Version erstellt.

**Achtung:**
Löschen Sie hier aber ein Dokument, dann ist es wirklich weg, während beim löschen in der „normalen“ Benutzeransicht nur eine Löschmarkierung gesetzt wird.

# 3. DH

Der DataHistorian besteht aus drei Komponenten. Das sind die Scripte der Schnittstellen und diverse Hilfsprogramme im Verzeichnis */opt/DH*, die Archivdateien im Verzeichnis */var/lib/DH* und die DH Datenbank (MariaDB).
Die Schnittstellen lesen die Daten aus ihren Quellen und legen sie in der Datenbanktabelle *akt* ab. Der Kompressor (*/opt/DH/DH_comp.py*) komprimiert die Daten und schreibt sie ins Verzeichnis */var/lib/DH*. Visualisiert werden die Daten schließlich von der Benutzeroberfläche der unidb.

## 3.1. Point- und Tagkonfiguration

Points sind die Objekte, denen die Daten zugeordnet sind. Tags sind lediglich Links auf diese Objekte. Es kommt häufig vor, dass Anwender Tags benötigen, die zwar unterschiedliche Namen oder Pfade haben, denen aber der gleiche Point zugrunde liegt. Meist sind das organisatorische Gründe. Bei DataHistorians, die nur über Points verfügen, bleibt dem Admin nichts anderes übrig, als einen neuen Point anzulegen, wenn ein Anwender die Daten eines Points unter einem anderen Namen benötigt. Das hat dann zwei Nachteile. Der größte Nachteil ist, dass der neu angelegte Point keine Historie hat, weil er erst einmal Daten sammeln muss. Der zweite Nachteil ist, dass man auf diese Art und Weise Datenmüll produziert, weil man ab dem Zeitpunkt der Erstellung des neuen Points die gleichen Daten doppelt sammelt. Hat man stattdessen die Möglichkeit einfach einen neuen Tag anzulegen, so hat dieser die komplette Historie und sammelt auch keine doppelten Daten.
Legt man im DH einen neuen Point an, wird automatisch auch ein gleichnamiger Tag erstellt, welcher auch den gleichen Pfad wie der Point hat. Die Anwender sehen immer nur die Tags, nie den dazugehörigen Point.

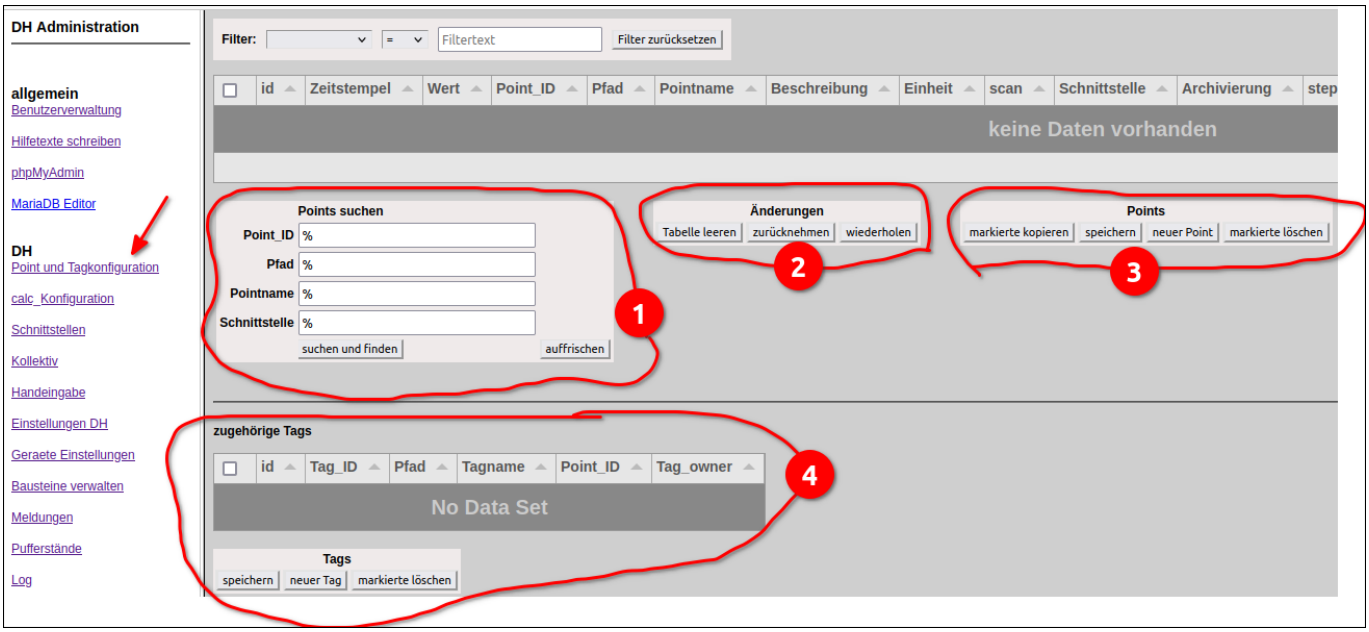


Abb. 14: Formular zur Konfiguration von Points und Tags.

Ruft man das Formular für die Point- und Tagkonfiguration auf, dann enthält es zunächst zwei leere Tabellen. Im Bereich Points suchen (1) hat man nun die Möglichkeit, gezielt nach einem oder mehreren Points zu suchen. Alle angebotenen Kriterien werden bei der Suche ODER - verknüpft. Das % Zeichen dient als Platzhalter für beliebig viele Zeichen. nach einem Klick auf die Schaltfläche „suchen und finden“ erscheinen die gefundenen Points in einem Popup - Fenster. Dort können beliebig viele Points markiert und ins Formular übernommen werden.

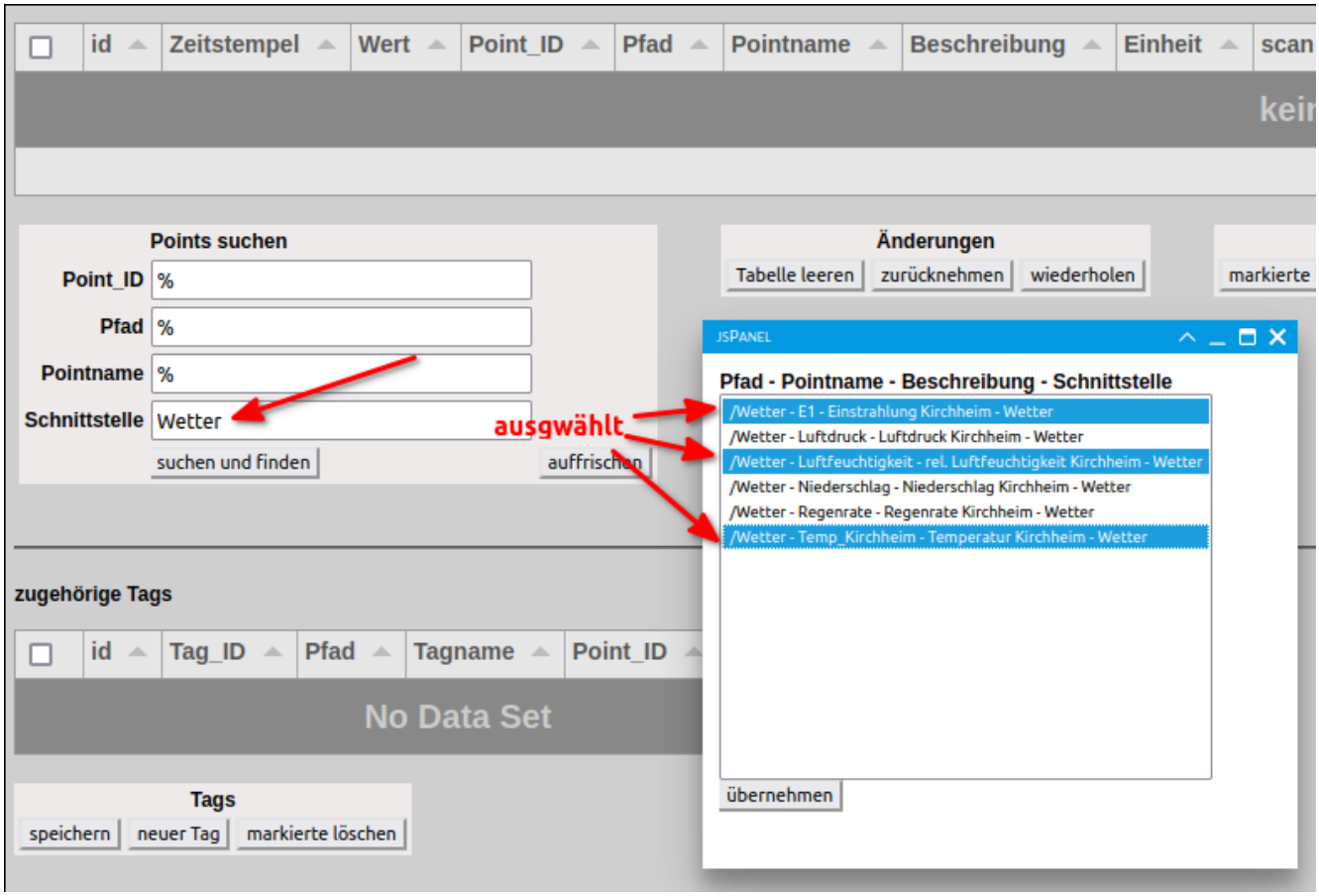



Abb. 15: Auswahl der zu bearbeitenden Points.

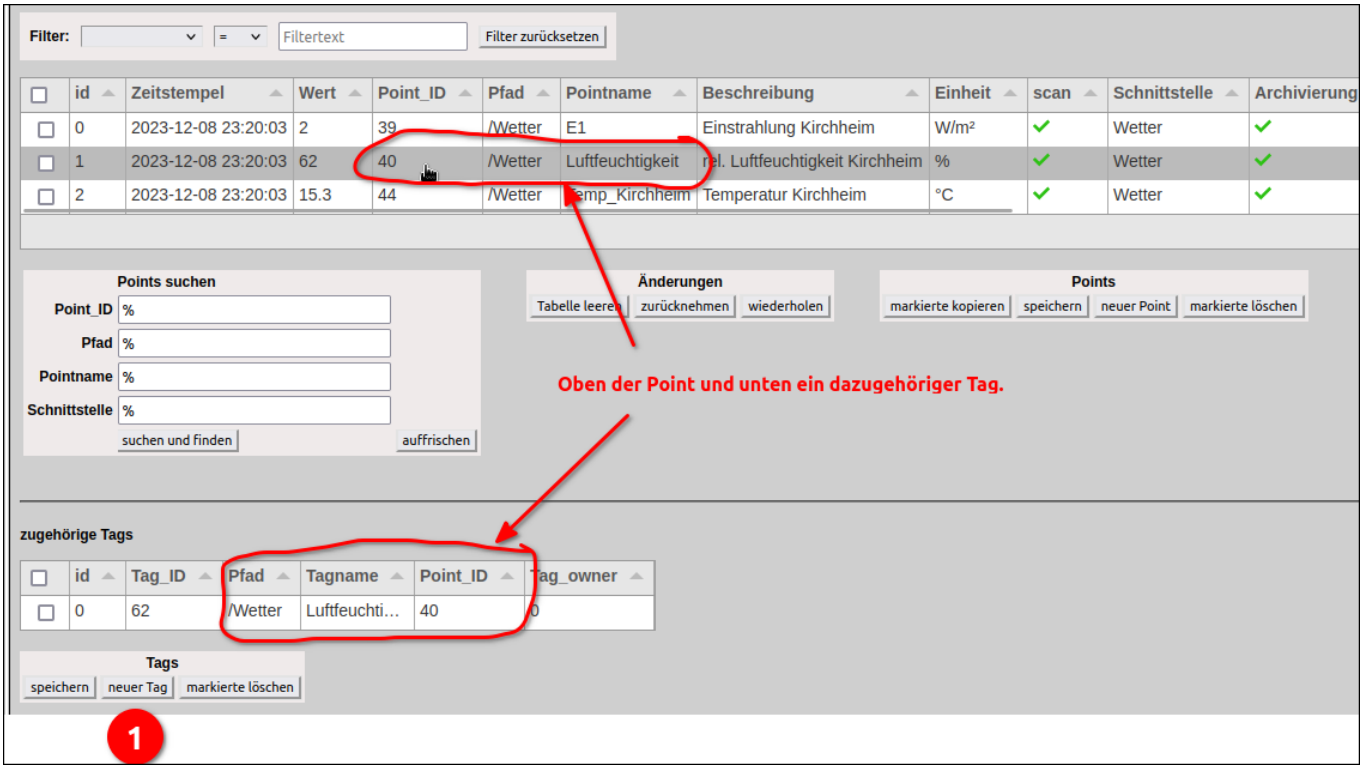



Abb. 16: Die ausgewählten Points aus Abb. 15 werden in der oberen Tabelle aufgelistet.

Die zweite Zeile ist in der oberen Tabelle markiert. In der unteren Tabelle wird ein zugehöriger Tag zum markierten Point angezeigt. Möchte man einen weiteren Tag anlegen, drückt man die Schaltfläche „**neuer Tag**“ (1).

zugehörige Tags

| | id | Tag_ID | Pfad | Tagname | Point_ID | Tag_owner |
|---|---|---|---|---|---|---|
| ☐ | 0 | 62 | /Wetter | Luftfeuchti... | 40 | 0 |
| ☐ | 1 | | /Wetter | Luftfeuchti... | 40 | 0 |

Tags

speichern | neuer Tag | markierte löschen

Abb. 17: Neue Zeile für einen neuen Tag.

In der unteren Tabelle erscheint eine neue Zeile, in der fast alle Felder vorbelegt sind. Lediglich das Feld für die Tag_ID ist leer. Das muss auch so sein, da der Tag erst angelegt wird, wenn man auf „**speichern**“ geklickt hat. Die *Tag_ID* wird automatisch vergeben. Zuvor muss aber mindestens eines der Felder *Pfad* oder *Tagname* editiert werden, da der neue Tag eine eindeutige Kombination aus Pfad und Tagname haben muss. Ist das nicht der Fall, wird der Tag nicht angelegt.

<u>Konfiguration der Points:</u>

Filter: | = | Filtertext | Filter zurücksetzen

| | id | Zeitstempel | Wert | Point_ID | Pfad | Pointname | Beschreibung | Einheit |
|---|---|---|---|---|---|---|---|---|
| ☐ | 0 | 2023-12-08 23:20:03 | 2 | 39 | /Wetter | E1 | Einstrahlung Kirchheim | W/m² |
| ☐ | 1 | 2023-12-08 23:20:03 | 62 | 40 | /Wetter | Luftfeuchtigkeit | rel. Luftfeuchtigkeit Kirchheim | % |
| ☐ | 2 | 2023-12-08 23:20:03 | 15.3 | 44 | /Wetter | Temp_Kirchheim | Temperatur Kirchheim | °C |

Abb. 18: Pointkonfiguration Teil 1

Die beiden ersten Spalten in Abb. 18 dienen der Verwaltung der Tabelle und haben nichts mit der eigentlichen Konfiguration zu tun. Die Spalten *Zeitstempel* und *Wert* zeigen den neuesten Wert für den Point aus der Zeile an. Das ist praktisch, wenn man einen Point umkonfiguriert oder neu angelegt hat, um zu erfahren, ob der Point auch funktioniert. Die *Point_ID* wird für einen neuen Point automatisch vergeben. Sie kann hier auch nicht editiert werden. Die beiden nächsten Parameter *Pfad* und *Pointname* müssen in ihrer Kombination eindeutig sein. Der Parameter *Beschreibung* bedarf sicherlich keiner weiteren Erläuterung. Die *Einheit* kann man zwar auch editieren, aber hier sollte man Vorsicht walten lassen. Ändert man die Einheit, dann kann es sein, dass die archivierten Werte nicht mehr passen. Beispiel: Eine Messung hat bisher die Werte in dm³/h geliefert und liefert ab jetzt in m³/h. Ändert man lediglich die Einheit in der Pointkonfiguration, dann werden die bereits archivierten Werte um den Faktor 1000 zu hoch angezeigt. Hier hilft nur ein neuer Point. Wenn es sich hingegen um eine Differenztemperatur handelt, die bisher in °C angezeigt wurde, dann kann die Einheit bedenkenlos auf die übliche Einheit K (Kelvin) geändert werden, da der Wert sich dadurch nicht ändert. Differenztemperaturen werden gewöhnlich in K angegeben.

| scan | Schnittstelle | Archivierung | step | Kompression | minarch | Info | Eigenschaft_1 | Eigenschaft |
|---|---|---|---|---|---|---|---|---|
| ✓ | Wetter | ✓ | ✗ | 1 | 3600 | | 1 | |
| ✓ | Wetter | ✓ | ✗ | 0 | 3600 | | 2 | |
| ✓ | Wetter | ✓ | ✗ | 0 | 3600 | | 6 | |

Page Size 10 | First | Prev | 1 | Next | Last

Abb. 19: Pointkonfiguration Teil 2

Ein Haken in der Spalte *scan* sorgt dafür, dass der Point von der Schnittstelle als aktiv erkannt wird und das dafür Daten gesammelt werden sollen. Anhand des Parameters *Schnittstelle* erkennen die Schnittstellen, welche Points zu ihr gehören. Mit dem Haken in der Spalte Archivierung wird dem

Kompressor mitgeteilt, dass er die Werte für diesen Point ins Archiv schreiben soll. Es kommt vor, dass die Werte eines Points nur für die Berechnung weiterer Werte benötigt werden, ansonsten aber uninteressant sind. In diesem Fall möchte man sie nicht archivieren und setzt in diese Spalte ein X. Der Parameter *step* gibt an, ob ein Wert für einen Zeitpunkt interpoliert werden soll, falls kein Wert zu exakt diesem Zeitpunkt im Archiv liegt. Bei der Trenddarstellung der Werte eines Points, der mit step = 1 (Haken) konfiguriert ist, erscheinen diese in Form einer Treppe. Steht in diesem Feld ein X, dann werden die Werte interpoliert.
Der Kompressor schreibt einen Wert nur dann ins Archiv, wenn er um den Betrag vom vorherigen Wert abweicht, den Sie hier unter *Kompression* eintragen. Damit verhindert man unnötig viele Werte im Archiv, was der Performance guttut. Außerdem wird so das Rauschen entfernt. Wird hier ein zu hoher Wert eingetragen, dann geben die archivierten Werte kein brauchbares Abbild des tatsächlichen zeitlichen Verlaufs des Wertes mehr her. Im schlimmsten Fall sind die Daten nicht zu gebrauchen.
Beispiel: Die Außentemperatur ändert sich nur langsam. Ohne Kompression erhält man vermutlich in 24 Stunden ca. 800 Werte. Die meisten Werte unterscheiden sich von dem jeweils vorhergehenden Wert um weniger als 0,1 °C. Fragt man sich, ob es wirklich erforderlich ist, diese Werte so hochauflösend zu archivieren, dann lautet die Antwort wahrscheinlich NEIN. Trägt man für solch einen Point 0.09 in das Feld *Kompression* ein, dann werden nur Werte ins Archiv geschrieben, die sich mindestens um 0,1 °C vom vorherigen Wert unterscheiden. Die Anzahl der Werte für einen Zeitraum von 24 Stunden reduziert sich so auf ca. 200 - 300. Ruft man einen Trend über einen Zeitraum von einem Jahr ab, dann macht sich das schon sehr deutlich in der Performance bemerkbar. Trends über einen kürzeren Zeitraum lassen sich angenehmer betrachten, wenn das Rauschen verschwunden ist. Wird für die Kompression eine 0 angegeben, dann werden nur unveränderte Werte von der Archivierung ausgenommen.
Der Parameter *minarch* gibt den Zeitraum in Sekunden an, nachdem ein Wert auf jeden Fall ins Archiv geschrieben wird, auch wenn er sich nicht geändert hat. Das erleichtert spätere Berechnungen und wirkt sich positiv auf die Trenddarstellung aus.
Das Feld Info enthält zusätzliche Angaben für die Schnittstelle, die sie zum Lesen der Werte benötigt. Das kann z.B. die ID eines 1-wire - Sensors sein oder die Angabe einer URL. Ob etwas und wenn ja, was in diesem Feld steht, hängt von der Schnittstelle ab. Dieses Feld nimmt Daten vom Typ Text auf.
Bei den folgenden fünf Parametern verhält es sich ebenso, wie beim Parameter Info. Dabei erwartet der Parameter Eigenschaft_1 einen integer - Wert und die restlichen Eigenschaft - Felder jeweils einen alphanumerischen Wert.

| Pointtyp | Dezimalstellen | Skala min | Skala max | Intervall | Mittelwerte | letzte Änderung | erster Wert | Point_owner |
|---|---|---|---|---|---|---|---|---|
| double | 0 | 0 | 1000 | 300 | ✔ | 2023-04-09 23:16:28 | 2023-04-09 23:16:28 | 0 |
| double | 0 | 20 | 100 | 300 | ✔ | 2023-04-09 23:16:28 | 2023-04-09 23:16:28 | 0 |
| double | 1 | 10 | 20 | 300 | ✔ | 2023-04-09 23:16:28 | 2023-04-09 23:16:28 | 0 |

Page Size 10 First Prev 1 Next Last

Abb. 20: Pointkonfiguration Teil 3

*Pointtyp* ist stets double. Der Parameter stammt noch aus alten Zeiten und wird eigentlich nicht mehr benötigt. Dezimalstellen gibt an, mit wie vielen Nachkommastellen ein Wert den Anwendern angezeigt wird. Die Felder *Skala_min* und *Skala_max* enthalten Vorgaben für die Skalierung von Trends. Jeder Anwender kann sich für jeden Point eigene Grenzwerte für die Skalierung festlegen. Nur wenn ein Anwender für den Point keine Grenzwerte festgelegt hat, werden diese beiden Felder bei der Erstellung eines Trends herangezogen.
Der Parameter *Intervall* kann von den Schnittstellen verwendet werden, um für jeden Point einen individuellen Abfrageintervall zu verwenden. Die wenigsten Schnittstellen machen davon Gebrauch.
Im Feld *Mittelwerte* wird festgelegt, ob für diesen Point Stunden- und Tagesmittelwerte, sowie die Min- und max - Werte für jede Stunde und jeden Tag archiviert sollen. Ob man hier eine Haken oder ein X setzt, hängt vom Point selbst ab. Werden auf diesen Point Zählerstände gesammelt, dann macht es sicherlich Sinn, hier ein X zu setzen. Werden hingegen Temperaturen oder Durchflussmengen auf diesen Point geschrieben, dann macht es bestimmt Sinn, hierfür die Mittel- Min- und Maxwerte zu berechnen und zu archivieren.
Die beiden nächsten Felder dienen zur Information und können nicht editiert werden. Das Feld „*erster Wert*“ ist zudem wichtig für die Abfrage von Archivwerten. Trends können schneller erstellt werden, wenn der Rechner weiß, dass er keinen Wert vor dem angegebenen Zeitpunkt erwarten kann. Er spart sich so das Durchsuchen des gesamten Archivs.
Das letzte Feld *Point_owner* wird noch nicht verwendet. Das kann sich aber irgendwann mal ändern.

Ein paar praktische Erfahrungen:

1. Möchte man einen neuen Point erstellen, so spart man sich Arbeit, indem man zunächst einen ähnlichen, bereits existierenden Point sucht und diesen einfach kopiert. Dazu markiert man ihn in der ersten Spalte und drückt die Schaltfläche **markierte kopieren** unterhalb der Tabelle im Bereich **Points**. Es wird eine neue Zeile mit den kopierten Werten in die Tabelle eingetragen. Dabei ist das Feld *Point_ID* leer, was auf einen neuen ungespeicherten Point hinweist. Jetzt brauchen Sie wahrscheinlich nur noch die Parameter Pfad, Pointname und Beschreibung anzupassen.
2. Manchmal sucht man eine gefühlte Ewigkeit nach der Ursache, warum ein Point keine Werte liefert. Als sehr hilfreich hat sich hier erwiesen, den nicht funktionierenden Point zusammen mit mindestens einem weiteren, ähnlichen Point, in die Tabelle zu laden und dann die Points Spalte für Spalte zu vergleichen. Idealerweise sollte der Vergleichspoint zu der selben Schnittstelle gehören, wie der Problempoint.

Hinweise:

1. Im Bereich Points suchen befindet sich unten rechts ein Schalter mit der Aufschrift **auffrischen**. Zugegeben, dieser Schalter ist nicht günstig platziert. Er dient zum erneuten Einlesen der aktuellsten Werte für die Points in der oberen Tabelle. Hat man einen Point umkonfiguriert, macht man von diesem Schalter gerne regen Gebrauch, um festzustellen, ob für den betreffenden Point frische Werte eintreffen oder nicht.
2. Wenn eine geänderte Konfiguration eines Points gespeichert wird, wird automatisch eine Nachricht an die zugehörige Schnittstelle geschickt, damit diese die Konfiguration ihrer Points neu einliest. Das geschieht gewöhnlich innerhalb weniger Sekunden.

## 3.2. calc Konfiguration

Die Werte der Points, die der Schnittstelle **calc** zugeordnet sind, werden in regelmäßigen Abständen berechnet. Der zu berechnende Ausdruck steht in der Konfiguration dieser Points im Feld *Info*. Der Intervall für die Berechnung wird ins Feld *Intervall* geschrieben.

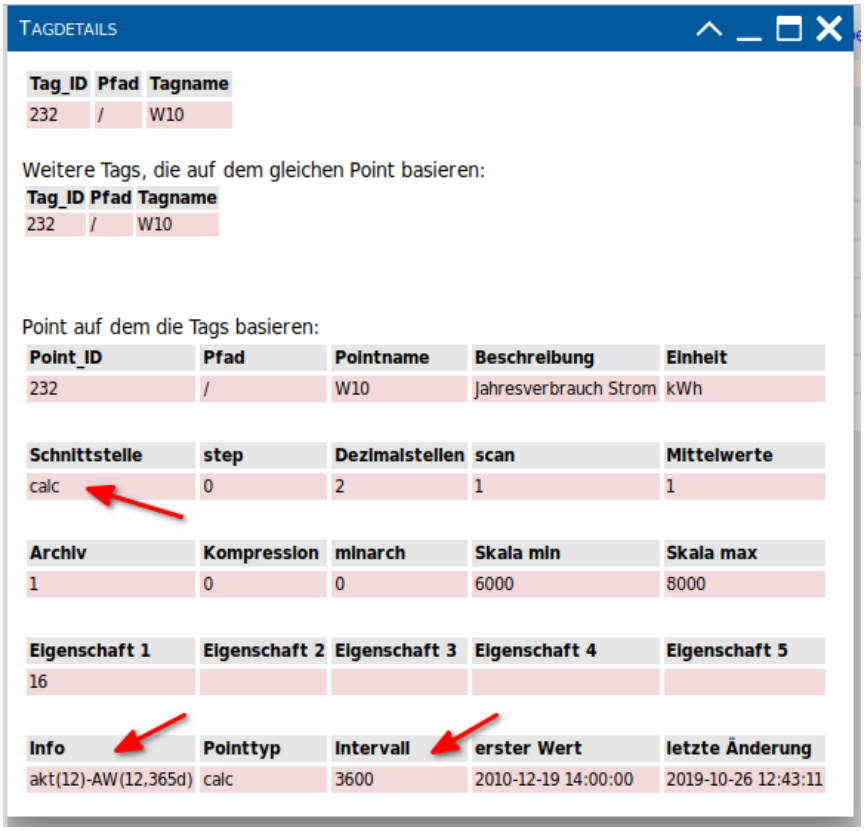



Abb. 21: Konfiguration eines Points der Schnittstelle *calc*.

Man kann somit einen zu berechnenden Point ebenso erstellen, wie jeder andere Point auch. In der Praxis kommt es jedoch häufig vor, dass eine Berechnung nicht funktioniert oder nicht das erwartete Ergebnis bringt. In solchen Fällen kann sich der Admin mit dem Formular **calc-Konfiguration** bequem weiter helfen. In diesem Formular kann man den zu berechnenden Point auswählen und die Berechnung testen. Des Weiteren werden noch ein paar weitere Konfigurationsparameter für den point angezeigt,

welche man hier ebenfalls editieren kann.
In Bildern lassen sich einige Elemente ebenfalls mit Berechnungen statt Tags konfigurieren. Die Syntax ist hierbei die gleiche, wie bei berechneten Points.

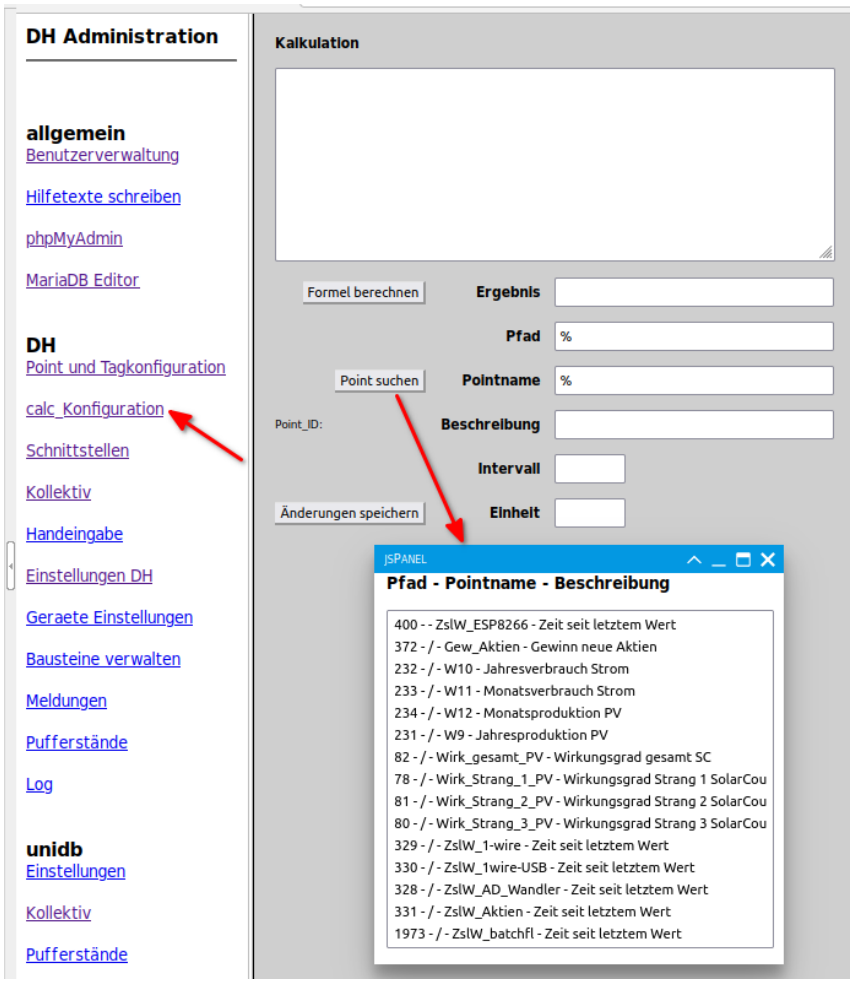


Abb. 22: Formular **calc-Konfiguration** mit Auswahldialog für berechnete Points.

Der Schalter Point suchen in Abb. 22 sucht nur nach Points, die der Schnittstelle *calc* zugeordnet sind. Über die Felder *Pfad* und *Pointname* läßt sich die Auswahl im Dialog weiter einschränken.
Wurde ein Point ausgewählt, werden die Felder mit dessen Daten ausgefüllt. Das oberste Feld enthält den Inhalt des Parameters *Info*, also den zu berechnenden Ausdruck.

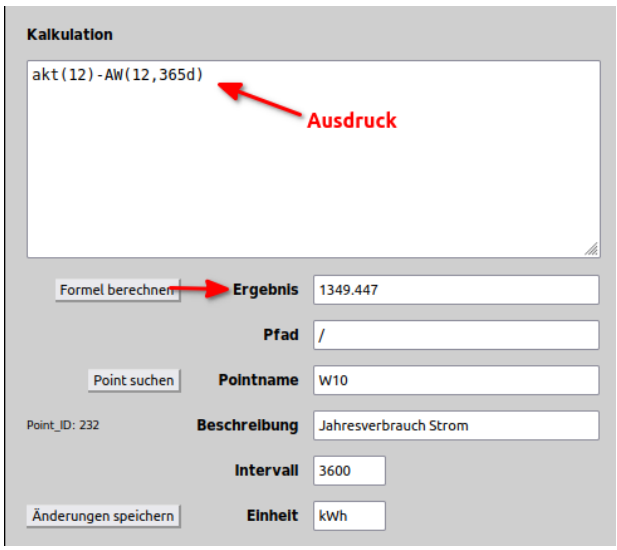


Abb. 23: Parameter des Points W10 mit dem Ergebnis der Berechnung.

Der Schalter **Formel berechnen** schreibt das Ergebnis der Berechnung in das Feld *Ergebnis*. Ist der Ausdruck fehlerhaft, dann bleibt das Feld leer.
Ist das Ergebnis plausibel, dann ist nichts weiter zu tun. Bleibt das Feld leer oder steht dort nur Unfug,

dann kann der Ausdruck im Formular editiert werden und mit einem Klick auf den Schalter **Formel berechnen** erneut getestet werden. Die geänderte Konfiguration des Points wird erst mit einem Klick auf den Schalter **Änderungen speichern** übernommen.

Um einen Ausdruck, der in einem Element eines Bildes verwendet wird, zu testen, kann man diesen in das oberste Feld kopieren und dann ebenfalls über den Schalter **Formel berechnen** berechnen lassen.

**Hinweis:**
Bei der Schnittstelle *calc* handelt es sich um ein Python - Script. Sie können daher alle Funktionen verwenden, die Python implementiert hat.
Ein Ausdruck, welcher Daten für ein Element in einem Bild berechnet, wird vom Webserver berechnet. Dieser nutzt PHP zur Berechnung. Die Unterschiede in den Funktionen von Python und PHP sind zwar nicht groß, aber es kann vorkommen, dass ein Ausdruck, welcher mit Python ausgewertet wird, im Formular calc-Konfiguration nicht ausgewertet werden kann. In solch einem Fall verwenden Sie zum Test bitte die Python Schreibweise. Die Funktion zum Abruf des aktuellen UNIX - Zeitstempels wird in Python *time.time()* geschrieben. In PHP reicht hingegen *time()*. Da diese Funktion häufig verwendet wird, enthält das PHP - Script eine Anpassung, welche *time.time()* nach *time()* übersetzt.

**Spezielle DH Funktionen für die calc - Schnittstelle:**
Funktionen, die eine oder mehrere Zeitangaben als Argument benötigen, nehmen diese auf zwei verschiedene Arten entgegen:

1. Absolute Zeit im Format YYYY-MM-DD hh:mm:ss (2023-12-15 17:45:13).
2. Relative Zeit als eine Kombination einer Zahl mit einem darauf folgenden Buchstaben. Der Buchstabe gibt an, ob es sich um Tage (d), Stunden (h), Minuten (m) oder Sekunden (s) handelt. Bei der Zahl davor handelt es sich um den Multiplikator. Die relative Zeit bezieht sich immer auf die aktuelle Zeit und liegt in der Vergangenheit.
   Beispiele:
   - Einen Tag in der Vergangenheit: 1d. Ist der aktuelle Zeitpunkt = 2023-12-1**5** 17:35:12, dann steht 1d für 2023-12-1**4** 17:35:12.
   - Drei Stunden in der Vergangenheit: 3h

- **akt(Point_ID):**
  Diese Funktion liefert den neuesten Wert für den angegebenen Point.
  Beispiel: akt(4711) liefert den neuesten Wert für den Point mit der Point_ID 4711.

- **AW(Point_ID,Zeitpunkt):**
  Liest den Wert des angegebenen Points zum gewünschten Zeitpunkt aus dem Archiv. Befindet sich für den angegebenen Zeitpunkt kein Wert im Archiv, dann wird der Wert des unmittelbar davor liegenden Eintrags zurückgegeben.
  Beispiel:
  Für den Point 4711 wurden folgende Werte archiviert:
  ...
  2023-12-15 17:10:12, 17.4, ...
  2023-12-15 17:15:26, 18.3, ...
  2023-12-15 17:18:02, 18.9, ...
  ...

  AW(4711,2023-12-15 17:14:28) liefert den Wert 17.4 zurück, weil für den gewünschten Zeitpunkt kein Wert archiviert wurde und der letzte Wert davor 17.4 von 17:10:12 Uhr beträgt.

- **intp(Point_ID,Zeitpunkt):**
  Liefert den interpolierten Wert für den Point, welcher für den angegebenen Zeitpunkt aus den archivierten Werten berechnet wurde. Es wäre purer Zufall, wenn exakt für den gewünschten Zeitpunkt ein Wert für den Point archiviert wäre. Daher wird immer dann, wenn kein Wert für den Point mit dem exakt passenden Zeitstempel im Archiv liegt, der Wert zu dem Zeitpunkt, aus dem direkt davor liegenden und dem direkt nachfolgenden Wert berechnet.
  Beispiel: AW(4711,2023-12-15 17:48:19) liefert den interpolierten Wert des Points 4711, den dieser zum angegebenen Zeitpunkt näherungsweise hatte. Bitte achten Sie darauf, dass Sie das Datum nicht in Anführungszeichen setzen.

- **MW(Point_ID,Zeitpunkt_von,Zeitpunkt_bis):**
  Berechnet den Mittelwert des betreffenden Points für den angegebenen Zeitraum. Wichtig hierbei ist, dass die Berechnung auch die zeitlichen Abstände zwischen den Werten berücksichtigt.

- **ZS(Point_ID):**
  Liefert den Zeitstempel des neuesten Wertes für den angegebenen Point.

- **ZP(Point_ID):**
  Gibt den Unix - Zeitstempel des neuesten Wertes für den angegebenen Point zurück. Der Unix - Zeitstempel gibt die Anzahl der Sekunden an, die seit dem 01.01.1970 00:00:00 vergangen sind.

## 3.3. Schnittstellen

Ich glaube, es hat sich mittlerweile herumgesprochen, dass die Schnittstellen Daten von irgendwo her sammeln und in die Tabelle *akt* eines jeden Servers im DH Kollektiv schreiben.
Einzige Ausnahme ist die Schnittstelle *comp*. Diese Schnittstelle liest die Daten aus der Tabelle *akt* und schreibt sie komprimiert ins Archiv. Diese Schnittstelle muss auf jedem Server laufen. Ebenfalls ein Sonderfall ist die Schnittstelle *watchdog*. Sie schaut auf dem Rechner, auf dem sie läuft, nach, welche Schnittstellen dort laufen sollten und welche tatsächlich laufen. Zudem liest sie die Meldungen vom Server und gibt sie an die anderen Schnittstellen weiter. Wenn die Meldung es verlangt, startet sie auch eine Schnittstelle. Die Schnittstelle *watchdog* läuft auf jedem Rechner im Kollektiv.

### 3.3.1. Programmablauf

Abb. 25 zeigt den Programmablauf, dem jede Schnittstelle folgt. Nach dem Start folgt zunächst die Initialisierung (1), bei der die Points für die Schnittstelle geladen werden. Es werden alle Points eingelesen, deren Parameter *Interface* gleich dem Namen der Schnittstellen ist und die mit *scan = 1* konfiguriert sind. Dazu werden noch die Einstellungen der Schnittstelle eingelesen. Dazu gehört immer der Parameter *Intervall*. Der *Intervall* legt fest, in welchem zeitlichen Abstand die Daten eingelesen werden. Eine Schnittstelle kann optional noch weitere Parameter laden. Die Einstellungen werden über das Formular **Schnittstellen** im Admin Bereich vorgenommen.

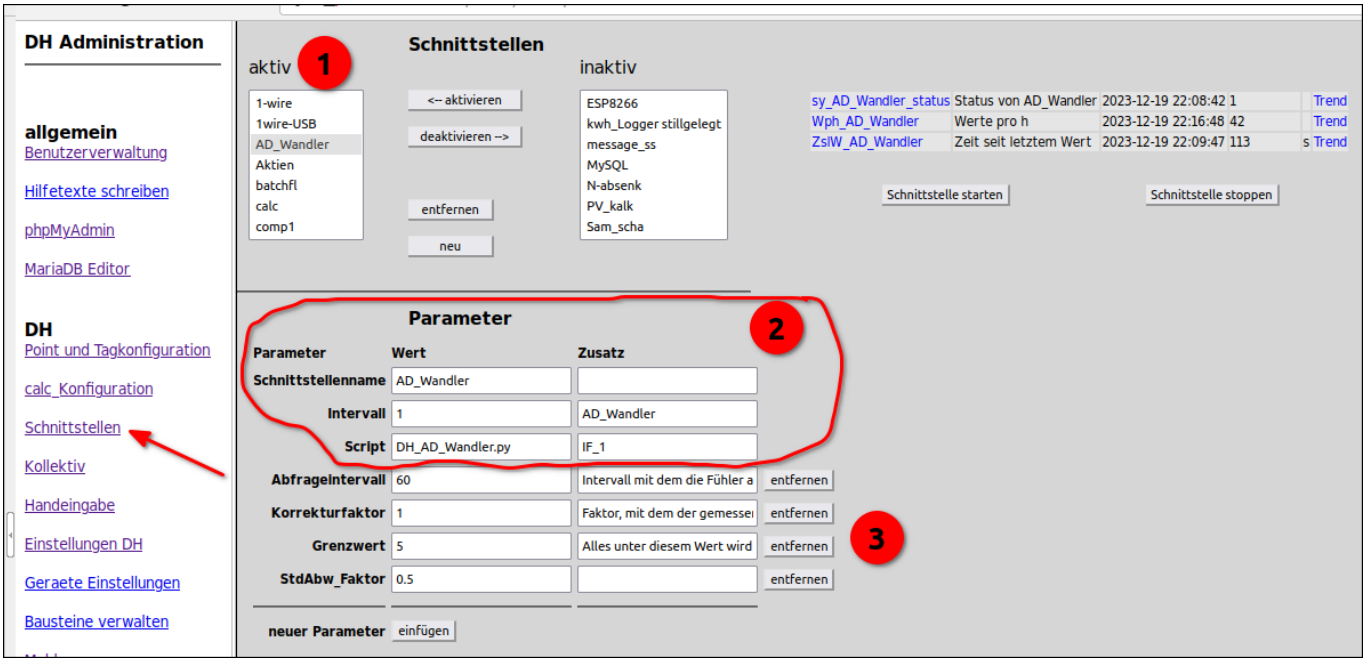


Abb. 24: Formular zur Konfiguration der Schnittstellen.

Im Formular aus Abb. 23 werden unter (2) und (3) die Parameter der Schnittstelle angezeigt, die unter (1) ausgewählt wurde. Im Beispiel oben ist das „AD_Wandler“. Die Parameter im markierten Bereich (2) werden von jeder Schnittstelle benötigt. Die letzten vier Zeilen (3) sind spezielle Einstellungen für die ausgewählte Schnittstelle.

Der Schnittstellenname kann frei vergeben werden (Zeile 1, Bereich (2)). Der Intervall muss immer angegeben werden. Meistens liegt der Wert bei 60, 300 oder 600 Sekunden. Im Feld Zusatz muss der Name des Scripts ohne den Teil DH_ am Anfang und ohne die Dateierweiterung stehen (Zeile 2, Bereich (2)). Das hat sich irgendwann so ergeben. Versuchen Sie es bitte nicht zu verstehen. Ich verstehe es auch nicht ;-). In die letzte Zeile des Bereichs (2) wird als Wert der vollständige Dateiname des Scripts und als Zusatz der Name des Rechners erwartet, auf dem das Script läuft. Im obigen Beispiel ist das der Schnittstellenrechner IF_1. Zur Erinnerung: Die Namen der Server und der Schnittstellenrechner des Kollektivs werden im Formular **Kollektiv** vergeben.

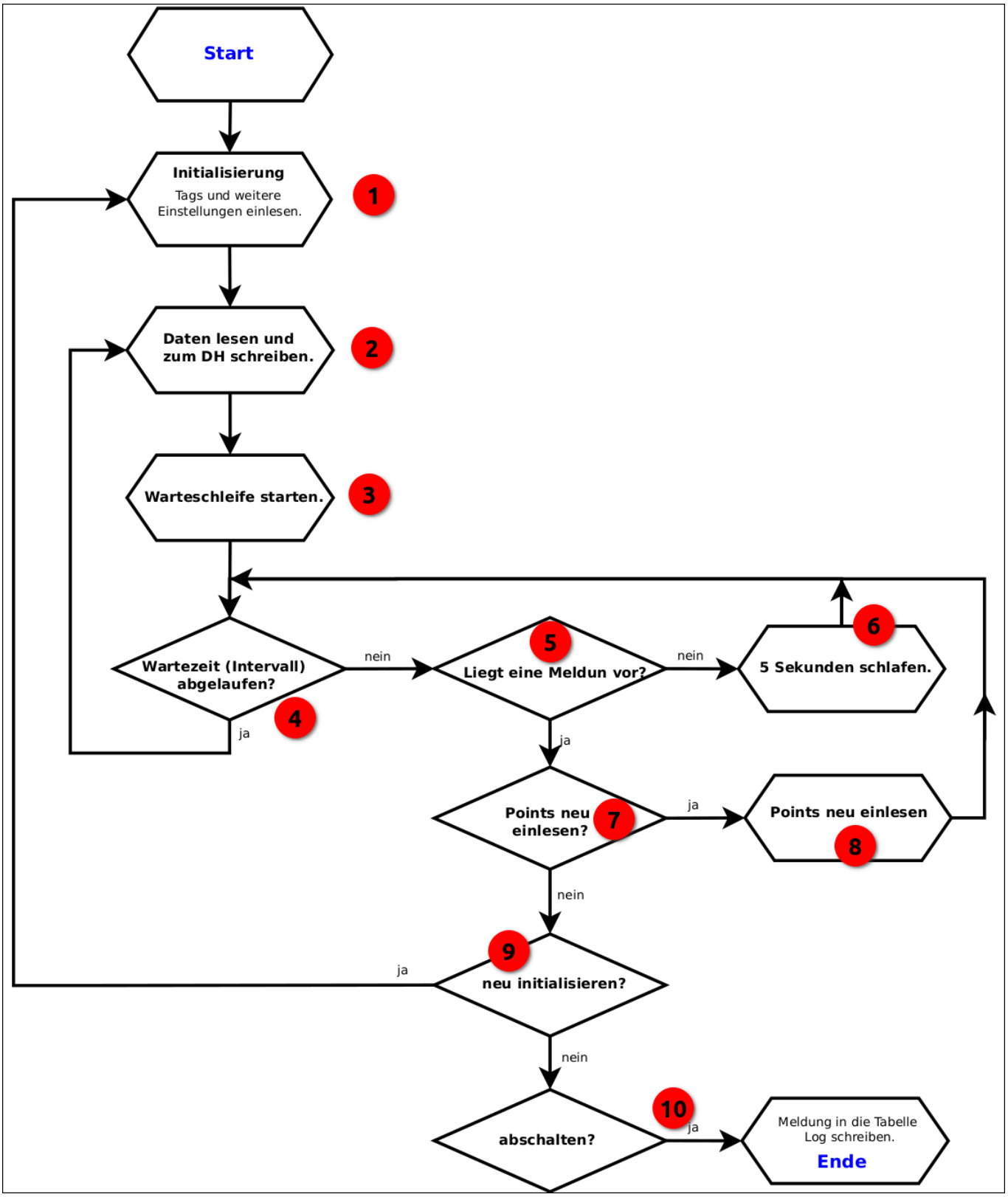


Abb. 25: Programm Ablaufplan Schnittstellen

Wir sind wieder bei Abb. 25. Nachdem die Initialisierung (1) abgelaufen ist, beginnt die Schleife aus Daten lesen und zum DH schreiben (2), Pause (3) bis (9), erneutem lesen und schreiben, ... .

Nachdem Punkt (2) abgearbeitet wurde, wartet die Schnittstelle maximal die Zeit ab, die als Intervall in Sekunden konfiguriert sind. Während dieser Zeit schaut sie alle 5 Sekunden nach, ob irgend eine Meldung für sie vorliegt. Meldungen werden von der Schnittstelle *watchdog* im temporären Verzeichnis abgelegt. Gewöhnlich wird hierfür das Verzeichnis /opt/DH/tmp verwendet. Sie können im Admin Bereich unter *Einstellungen / Serverkomponenten / temp Verzeichnis* jedoch auch ein anderes Verzeichnis festlegen. Meldungen, auf die die Schnittstellen reagieren sind „neu initialisieren“, „Points einlesen“ und „abschalten“. Der watchdog liest die Meldungen aus der Datenbank und erzeugt für jede Meldung eine Datei, deren Name mit DH_ce_ beginnt. Danach folgen der Schnittstellenname und schließlich der aktuelle Unix Zeitstempel.

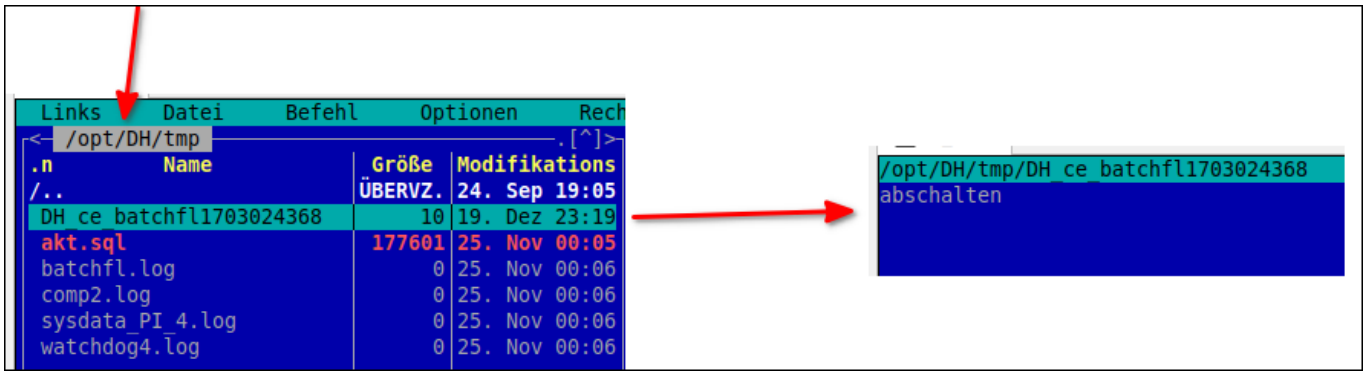


Abb. 26: Meldung an die Schnittstelle *batchfl*. Meldung: abschalten

Sobald sich eine Schnittstelle in der Warteschleife befindet, schaut sie nach, ob eine der drei Meldungen (7), (9), (10) vorliegt und handelt entsprechend. Liegt keine Meldung vor, dann legt sie sich für 5 Sekunden schlafen (6).
Ist die Pausenzeit (Intervall) abgelaufen, dann beginnt der Zyklus erneut, indem das Programm an den Punkt (2) zurück springt.

## 3.3.2. Schnittstellen starten & stoppen

Es gibt drei Möglichkeiten um Schnittstellen zu starten oder anzuhalten.

1. Direkt aus der Konsole:
   Öffnen Sie eine Konsole (Terminal) und starten Sie die Schnittstelle wie jedes andere Programm auch. Damit Sie die Konsole nicht geöffnet lassen müssen, sollten Sie dem Befehl ein & Zeichen anhängen, um das Programm im Hintergrund laufen zu lassen. Die Scripte der Schnittstellen liegen gewöhnlich im Ordner */opt/DH*. Ihr Dateiname fängt stets mit DH_ an. Das hat einen guten Grund, wie Sie weiter unten erfahren werden. Angenommen, das zu startende Script heißt *Wasser*. Der Befehl lautet: */opt/DH/DH_Wasser.py &*.
   Anhalten können Sie es über die Konsole nur, indem Sie den Prozess killen. Der Befehl *ps -e|grep DH_* listet alle Scripte auf, die auf dem Rechner laufen. Neben dem Namen des Scripts steht jeweils die PID. Der Befehl *kill 4711* beendet nun das Script mit der PID 4711. Das ist allerdings die brutale Methode, die man eigentlich vermeiden sollte. Macht man das im falschen Moment, dann kann das zu einer kaputten Tabelle in der Datenbank führen.
2. Im Ordner /opt/DH befinden sich zwei Scripte zum starten und stoppen aller Schnittstellen auf dem Rechner. Das Script **DH_start.py** startet jede Schnittstelle, die im Formular Schnittstellen (Admin Bereich) in der Lister **aktiv** stehen. Das Script **DH_stop.py** schickt einfach an jede laufende Schnittstelle die Meldung **abschalten**. Nachdem Sie das Script **DH_stop.py** gestartet haben, sollte der Befehl *ps -e|grep DH_* nach spätestens 15 Sekunden keine laufenden Schnittstellen mehr auflisten.
   Möchten Sie die Schnittstellen beim Start des Rechners automatisch starten, dann erstellen Sie einfach einen Service, welcher das Start-Script ausführt. Ebenso können Sie beim Herunterfahren verfahren. Das stellt sicher, dass alle Schnittstellen sauber beendet werden.
3. Sie können eine Schnittstelle über die Schaltfläche Schnittstelle starten, bzw. Schnittstelle stoppen im Formular Schnittstellen (Admin Bereich) starten und stoppen. Siehe Punkt (4) und Punkt (5) in Abbildung 27.

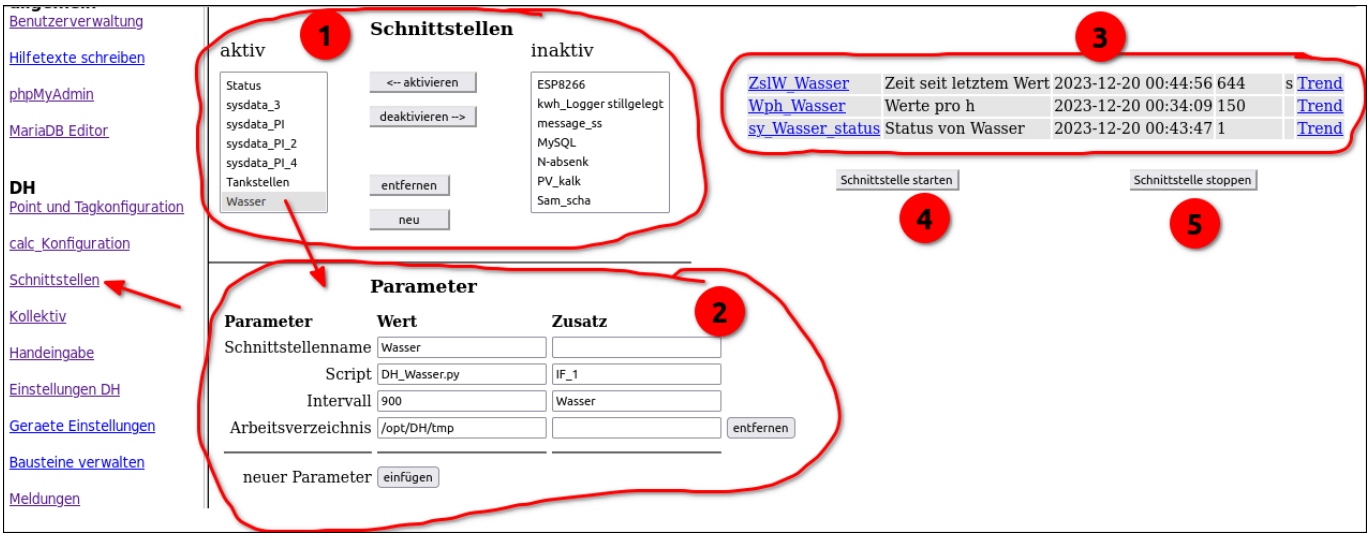


Abb. 27: Formular *Schnittstellen* im Admin Bereich

## 3.3.3. Konfiguration

Die Konfiguration wird im Formular **Schnittstellen** im Admin Bereich vorgenommen.
Im Bereich (1) sehen Sie zwei Listen *aktiv* & *inaktiv*. Die Schnittstellen in der Liste *aktiv* werden durch das Script *DH_start.py* gestartet und von der Schnittstelle *watchdog* überwacht. Mit den beiden Schaltern **<--- aktivieren** und **deaktivieren --->** schieben Sie eine Schnittstelle in die jeweils andere Liste. Wird eine Schnittstelle deaktiviert, werden auch die drei nebenstehenden Tags (3) auf *scan = 0* gesetzt. Bei der Aktivierung werden die Tags im Hintergrund wieder auf *scan = 1* konfiguriert. Vorsicht sollten Sie beim Schalter **entfernen** walten lassen, denn der löscht die komplette Konfiguration der gewählten Schnittstelle. Dabei ist es egal, in welcher der beiden Liste die Schnittstelle steht.

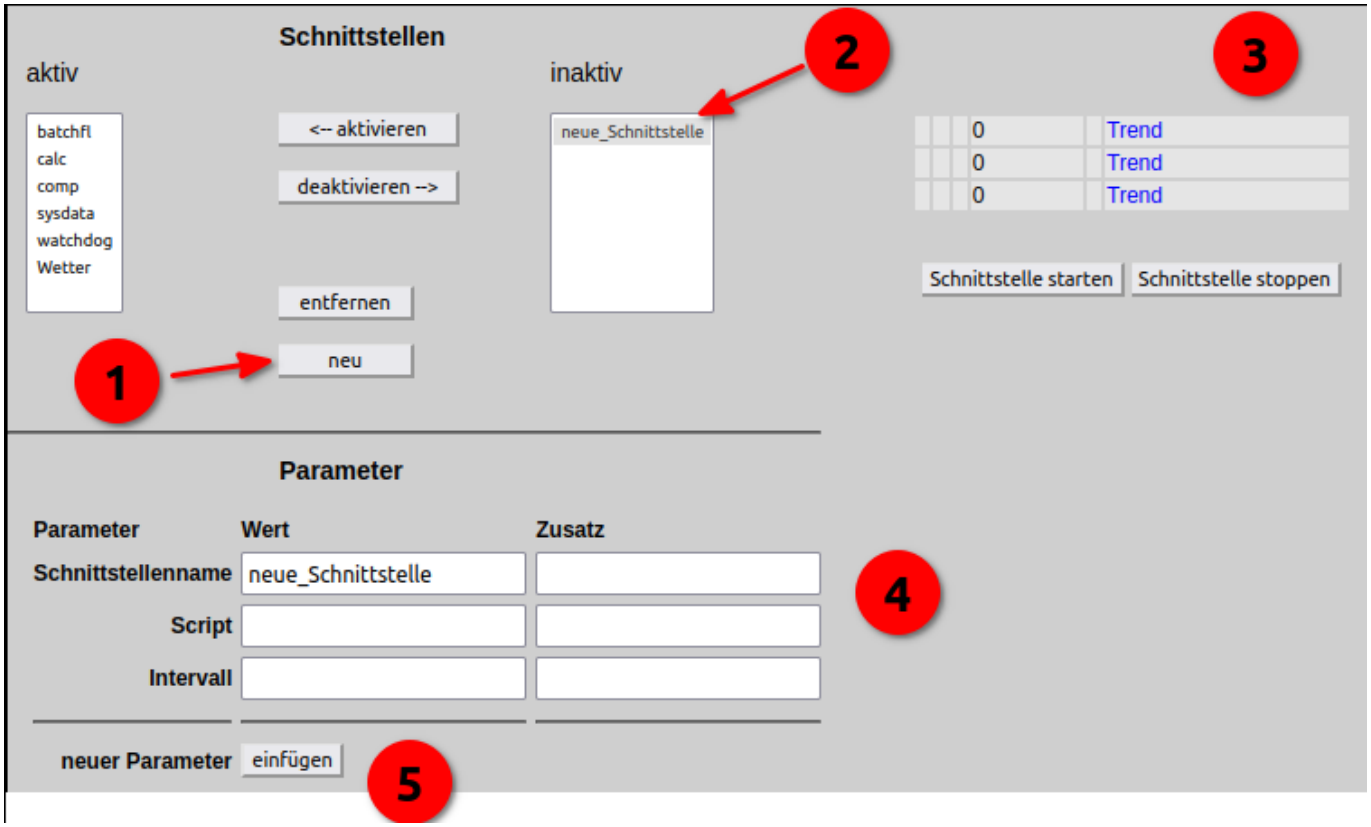


Abb. 28: neue Schnittstelle

Eine neue Schnittstelle wird mit dem Schalter **neu** (1) angelegt. Die neue Schnittstelle finden Sie in der Liste **inaktiv** (2). Im nächsten Schritt müssen die drei Standard Parameter *Schnittstellenname*, *Script* und *Intervall* konfiguriert werden. Was dort hingehört wurde bereits unter Abb. 23 erklärt. Benötigen Sie weitere Parameter für die Schnittstelle, fügen Sie sie einfach mit dem Schalter **einfügen** (5) ein.
Zu jeder Schnittstelle gehören vier Tags, welche Informationen über den Betrieb der Schnittstelle liefern.

Der bequemste Weg, diese Points anzulegen ist, Sie suchen im Formular **Point und Tagkonfiguration** die Points für eine bereits vorhandene Schnittstelle.

| Pfad | Pointname | Beschreibung | Einheit | scan | Schnittstelle | Archivierung | step | Kompression | minarch | Info |
|---|---|---|---|---|---|---|---|---|---|---|
| / | sy_1-wire_status | Status von 1-wire | | ✓ | watchdog1 | ✓ | ✓ | 0 | 3600 | IF_1 |
| / | Wph_1-wire | Werte pro h | | ✓ | System | ✓ | ✗ | 0 | 28800 | 1-wire: Werte_pro_h_Tag |
| / | ZslW_1-wire | Zeit seit letztem Wert | s | ✓ | calc | ✓ | ✓ | 0 | 0 | time.time()-akt(1999) |
| / | ZlW_1-wire | Zeitpunkt des letzten Wertes | s | ✓ | System | ✗ | ✓ | 0 | 0 | 1-wire: Zeitpunkt_letzter_Wert |

Abb. 29: Points für Schnittstelle *1-wire*.

<u>Beispiel:</u>
In Abb. 28 sehen Sie die Konfiguration der Points für die Schnittstelle *1-wire*.
Und so gehen Sie vor:

1. Im Feld Pointname den Suchtext *%Schnittstellenname%* eintragen. Es sollten vier Points gefunden werden.
2. Markieren Sie nun alle Zeilen, indem Sie in die erste Spalte jeweils ein Kreuz setzen. Mit dem Schalter **markierte kopieren** unterhalb der Tabelle lassen sich diese Points kopieren.
3. Ersetzen Sie in der Spalte Pointname den Namen der Schnittstelle. Soll die neue Schnittstelle ADWandler heißen, ändern Sie den Pointnamen *sy_1-wire_status* in *sy_ADWandler_status*. Verfahren Sie ebenso mit den anderen Points.
4. Passen Sie jeweils den Parameter **Info** an. Auch hier jeweils den Namen ersetzen.
5. Den Schalter **speichen** unterhalb der Tabelle drücken.
6. Der Parameter **Info** des Points *ZslW_ADWandler* muss noch angepasst werden. Ersetzen Sie hier die Point_ID im Ausdruck *time.time()-akt(1999)* durch die Point_ID des neuen Points *ZlW_ADWandler*.
7. Noch einmal auf **speichern** klicken.

Wenn Sie jetzt zum Formular Schnittstellen zurückkehren und dort die neue Schnittstelle anklicken, dann sollten Sie drei der vier neuen Points im Bereich (3, Abb. 27) sehen.
Die Konfiguration ist damit fertig.

**<u>Hinweis:</u>**
Auf jedem Server des Kollektivs muss eine Schnittstelle **comp** laufen. Schnittstellennamen müssen aber auch eindeutig sein. Gute Praxis ist, die comp - Schnittstellen zu nummerieren, also *DH_comp1.py*, *DH_comp2.py*, *DH_comp3.py*, usw.. Wichtig ist, dass der Name der comp-Schnittstellen immer mit **DH_comp** beginnt.

## 3.3.4. Points zur Überwachung der Schnittstellen

Wie bereits oben beschrieben, benötigt jede Schnittstelle vier Points zur Überwachung.

1. **sy_NAME_status**:
   Die Schnittstelle watchdog schaut regelmäßig nach, ob die Schnittstelle NAME noch läuft und schreibt eine 1 für *läuft* oder eine 0 für *läuft nicht*.
2. **Wph_NAME**:
   Wph steht für Werte pro Stunde. Die Schnittstelle schreibt auf diesen Point die Anzahl der Werte, die sie pro Stunde an die Server schickt. Als *Schnittstelle* muss der watchdog angegeben werden, der auf dem gleichen Rechner läuft, wie die Schnittstelle selbst. Der Name des Rechners steht im Parameter *Info*. Dieser Point liefert Infos darüber, wie regelmäßig Daten von der Schnittstelle geliefert werden.
3. **ZslW_NAME**:
   *ZslW* steht für *Zeit seit letztem Wert*. Die Schnittstelle **calc** berechnet in regelmäßigen Abständen die Zeit, die vergangen ist, seitdem der letzte Wert der Schnittstelle in die Tabelle *akt* geschrieben wurde. Wenn das Script einer Schnittstelle nicht sorgfältig geschrieben wurde, kann es vorkommen, dass sich eine Schnittstelle im Kreis dreht. Sie läuft zwar noch, aber sie liefert keine Werte mehr. Das ist so, als ob ein Hund versucht, sich selbst in den Schwanz zu beißen. Der Status - Point zeigt dabei immer brav eine 1 an. Der Wert vom ZslW - Point steigt hingegen immer weiter an.
4. **Zlw_NAME**:
   Das ist ein besonderer Point, denn seine Werte werden nicht ins Archiv geschrieben. Der Kompressor löscht die Werte regelmäßig aus der Tabelle akt. Er läßt nur den neuesten Wert stehen. Der Point dient nur als Zulieferer für den Point ZslW_NAME. Der Wert des Points ist der Unix-

Zeitstempel der letzten Übertragung von Werten der Schnittstelle. Die Point_ID des Points *ZIW_NAME* wird zur Berechnung des Points *ZslW_NAME* benötigt und muss daher im Ausdruck stehen, welcher im Feld *Info* des Points *ZslW_NAME* steht.

## 3.3.5. Störungen und deren Behebung

Liefert eine Schnittstelle keine Daten mehr, dann kann man sie einfach neu starten. Besser ist es aber, wenn man vorher nachschaut, warum die Schnittstelle nichts mehr liefert.
Liegt ein Fehler im Programm vor, dann sollte man einen Eintrag in der Log - Tabelle finden.

Abb. 30: Schnittstelle Wasser ist ausgefallen, Fehler im Script.

Es kann aber auch sein, dass die Schnittstelle keine Verbindung mehr zur Datenbank hatte, als sie ausgefallen ist. Oder das Script ist einfach nicht so programmiert, dass es eine Meldung in die Log-Tabelle schreibt. In solchen Fällen lohnt sich ein Blick in das temporäre Verzeichnis, über welches auch die Meldungen an die Schnittstellen geschrieben werden. Gewöhnlich ist das der Ordner */opt/DH/tmp*. In diesem Ordner finden Sie eine Datei mit dem Namen der Schnittstelle und der Erweiterung *.log*. Darin befindet sich der Text, den ein Script normalerweise auf die Konsole schreibt. Die Schnittstellen leiten die Ausgabe an die Konsole in diese Datei um.
Bevor Sie eine Schnittstelle neu starten, vergewissern Sie sich bitte, dass nicht noch eine Instanz davon läuft. Das erledigen Sie entweder mit dem Befehl **ps -e|grep DH_** über die Konsole oder über den Admin Bereich auf der Webseite im Formular *Schnittstellen*.

## 3.3.6. Standardschnittstellen

- **comp:**
  Diese Schnittstelle muss auf jedem Server laufen. Sie liest die Werte aus der Tabelle *akt* und schreibt sie komprimiert ins Archiv. Danach löscht sie die nicht mehr benötigten Werte in der Tabelle *akt*. Ihr sind keine Points direkt zugeordnet. Sie bearbeitet alle Points, deren Parameter **scan**=1 und **archive**=1 sind.

  Konfiguration:
  **von_Point_ID** und **bis_Point_ID** geben den Bereich der Points an, für die die Schnittstelle zuständig ist. Wenn comp zu viel Zeit benötigt, um alle Points abzuarbeiten, dann kann somit eine zweite Schnittstelle comp konfiguriert werden. Die Schnittstellen bearbeiten dann nur den Bereich, der ihnen über **von_Point_ID** und **bis_Point_ID** zugewiesen wurde.

- **watchdog:**
  Diese Schnittstelle muss auf jedem Rechner im Kollektiv laufen. Sie überwacht die anderen Schnittstellen und verteilt Nachrichten an sie. Bei Bedarf kann sie auch eine andere Schnittstelle starten.
  Die Points dieser Schnittstelle benötigen den Namen des Rechners, auf dem die Schnittstelle läuft, im Feld **Info**.

- **calc:**
  Es ist egal, auf welchem Rechner im Kollektiv *calc* läuft. Die Schnittstelle berechnet für jeden der ihr zugeordneten Points den Ausdruck, der in der Pointkonfiguration im Feld **Info** steht. In welchem zeitlichen Abstand die Werte berechnet werden, wird für jeden Point über den Parameter **Intervall** in der Pointkonfiguration festgelegt.

- **batchfl:**
  Dieser Schnittstelle sind keine Points direkt zugewiesen. Sie liest Werte aus flachen Textdateien ein und schreibt sie in die Tabelle *akt*. Die Textdateien müssen die Dateiendung **.dat** haben. Die *batchfl* ändert die Dateiendung auf .xxx, sobald sie die Werte eingelesen hat. Damit wird verhindert, dass eine Datei mehrfach eingelesen wird. Andererseits kann man immer nachvollziehen, welche Daten über die batchfl - Schnittstelle eingelesen wurden.
  In der Datei muss jeder Wert in einer separaten Zeile stehen. Zusätzlich wird noch ein Zeitstempel und die Point_ID in der gleichen Zeile benötigt. Getrennt werden die drei Angaben mit jeweils einem Komma. Daraus ergibt sich folgendes Format für die Daten in einer .dat - Datei:
  **Point_ID,Zeitstempel,Wert**

  Wichtig: Das Dezimaltrennzeichen ist ein Punkt.

  Beispiel für den Inhalt einer .dat - Datei:
  4711,2023-12-27 15:17:19,36.47
  815,2023-12-27 15:15:33,151.3

  Klartext: Für den Point mit der Point_ID wird der Wert 36.47 zusammen mit dem Zeitstempel 2023-12-27 15:17:19 in die Tabelle *akt* geschrieben. Weiterer Eintrag in die Tabelle *akt*: Point_ID: 815, Zeitstempel: 2023-12-27 15:15:33, Wert: 151.3.

  Die Schnittstelle benötigt in der Konfiguration den Parameter **Pfad**, welcher den Ordner angibt, in dem die Schnittstelle nach neuen *.dat* - Dateien sucht.

## 3.4. Kollektiv

Schnittstellen schreiben ihre Daten nicht direkt auf einen Server. Sie nutzen stattdessen eine Funktion. Diese Funktion schaut nach, welche Server im Kollektiv vorhanden sind, und versucht dann die Daten auf jeden dieser Server zu schreiben. Ist ein Server nicht erreichbar, dann schreibt sie die Daten in die eigene, lokale Datenbank und liefert sie dann an den Server nach, sobald dieser wieder erreichbar ist. Dabei ist es der Funktion egal, wie viele Server sie beliefern muss.
Daraus folgt, dass wir es aus Sicht der Schnittstellen, immer mit einem Kollektiv zu tun haben, auch wenn es nur einen Server gibt. Da eine Schnittstelle auch auf einem Server laufen kann, benötigt man somit mindestens nur einen Rechner für das Kollektiv. Eigentlich kann man einen einzelnen Rechner zwar nicht ein Kollektiv nennen, aber die Schnittstellen interessiert das nicht.

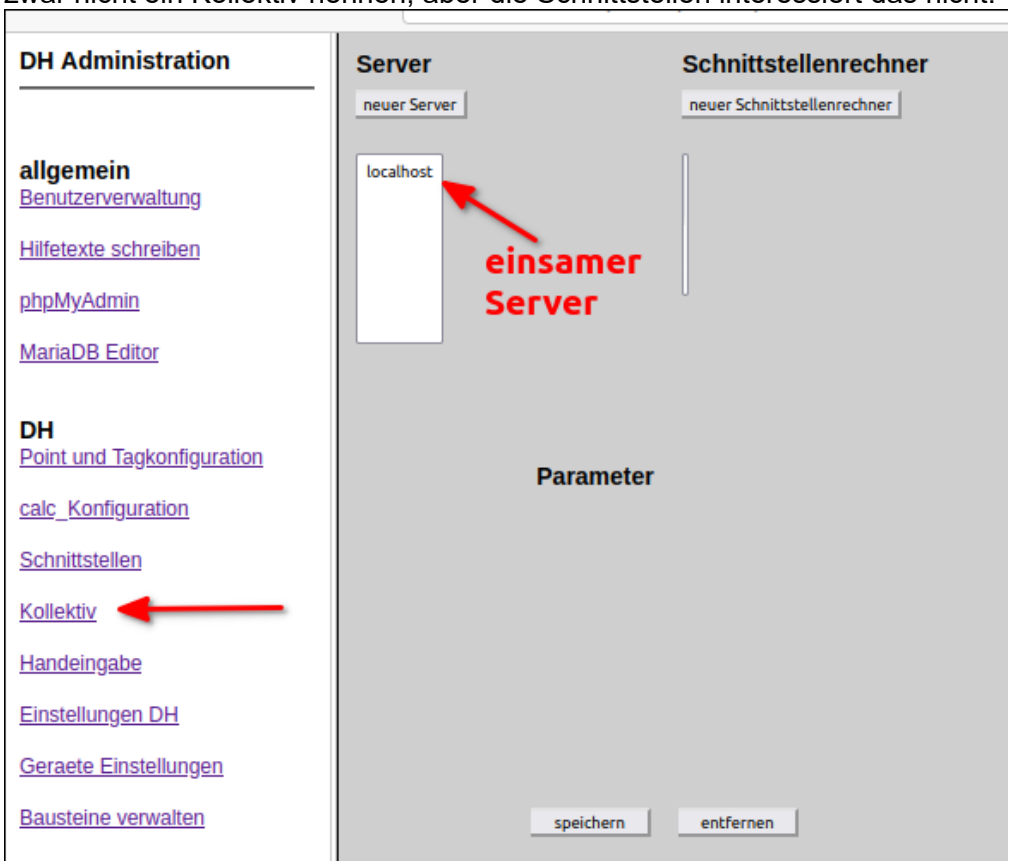


Abb. 31: Formular *Kollektiv* im Administrationsbereich

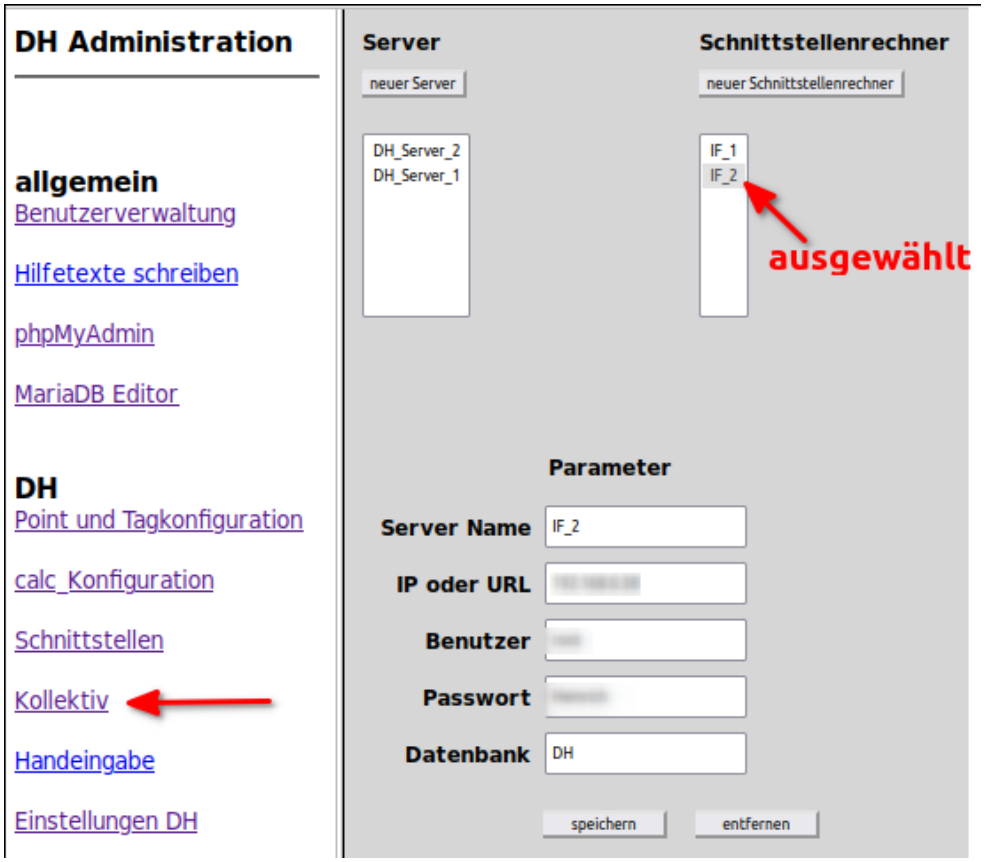


Abb. 32: Kollektiv aus zwei Servern und zwei Schnittstellenrechnern.

Der Unterschied zwischen Server und Schnittstellenrechner besteht darin, dass ein Schnittstellenrechner zwar eine lokale Datenbank benötigt, dort aber kein Archiv anlegt. Sie enthält auch keine Tabellen zur Konfiguration von Points, Tags, Schnittstellen, usw.. Die lokale Datenbank dient nur zur Pufferung von Werten, falls ein Server nicht erreichbar ist.

### 3.4.1. Schnittstellen im Kollektiv

Auf welchem Rechner eine Schnittstelle läuft, wird im Formular Schnittstellen als Zusatz zum Parameter Script angegeben.

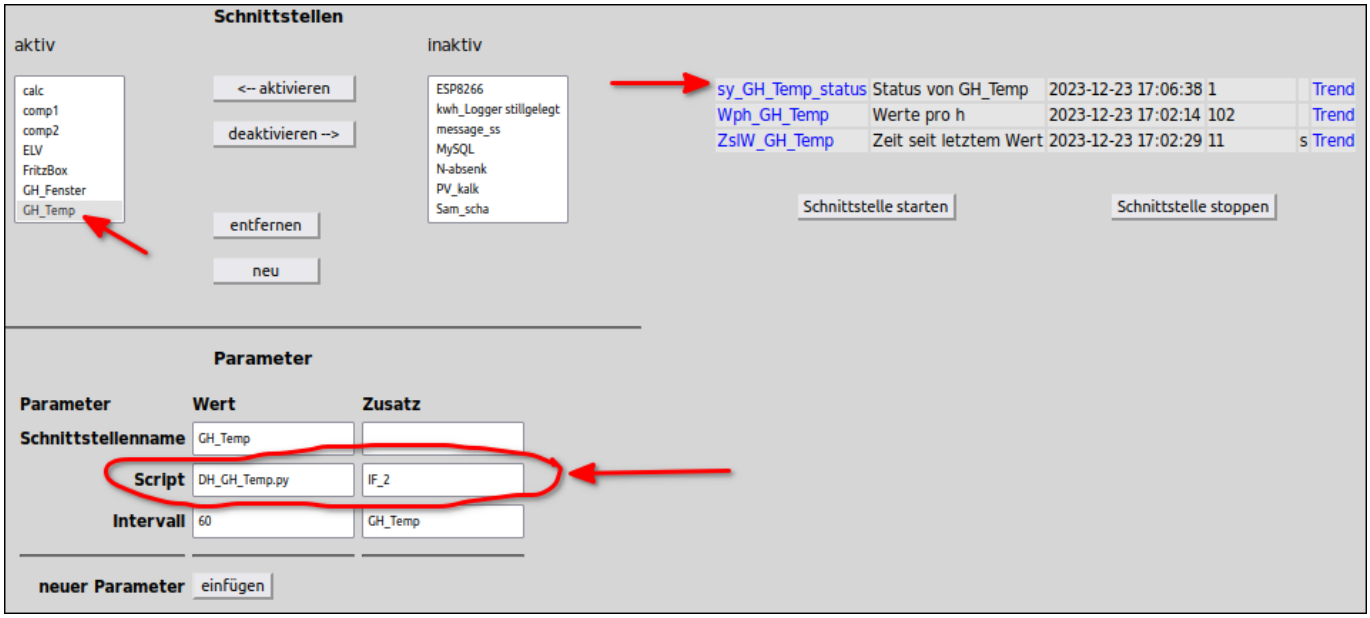


Abb. 33: Konfiguration einer Schnittstelle, welche auf dem Schnittstellenrechner IF_2 läuft.

Des Weiteren muss der Rechnername im Feld *Info* des Points *sy_NAME_status* eingetragen werden. Im obigen Beispiel sieht das dann so aus:

TAGDETAILS

| Tag_ID | Pfad | Tagname |
|---|---|---|
| 252 | / | sy_GH_Temp_status |

Point auf dem die Tags basieren:

| Point_ID | Pfad | Pointname | Beschreibung | Einheit |
|---|---|---|---|---|
| 252 | / | sy_GH_Temp_status | Status von GH_Temp | |

| Schnittstelle | step | Dezimalstellen | scan | Mittelwerte |
|---|---|---|---|---|
| watchdog2 | 1 | 1 | 1 | 0 |

| Archiv | Kompression | minarch | Skala min | Skala max |
|---|---|---|---|---|
| 1 | 0 | 3600 | 0 | 1 |

| Eigenschaft 1 | Eigenschaft 2 | Eigenschaft 3 | Eigenschaft 4 | Eigenschaft 5 |
|---|---|---|---|---|
| 1 | | | | |

| Info | Pointtyp | Intervall | erster Wert | letzte Änderung |
|---|---|---|---|---|
| IF_2 | double | 0 | 2018-11-03 20:39:32 | 2021-10-13 16:40:00 |

Abb. 34: Konfiguration des Points *sy_NAME_status*.

Die Schnittstelle *watchdog* muss auf jedem Rechner im Kollektiv laufen. Allerdings muss der Name einer Schnittstelle im gesamten Kollektiv eindeutig sein. Es ist eine gute Praxis, wenn man die Schnittstelle *watchdog* einfach durchnummeriert.
Ebenso sollte man bei der Schnittstelle *comp* verfahren. Diese muss auf jedem Server im Kollektiv laufen.

Abb. 35: Viermal *watchdog*, einfach durchnummeriert.

## 3.4.2. Schnittstellenrechner zum Kollektiv hinzufügen oder entfernen.

Um einen Schnittstellenrechner zum Kollektiv hinzuzufügen, sind nur wenige Dinge zu erledigen.

1. Kopieren Sie die Scripte für die Schnittstellen, die Scripte *DH_start.py*, *DH_stop.py* und *DH_biblio2.py* in den Ordner */opt/DH*.
2. Legen Sie im Ordner */opt/DH* ein neues Verzeichnis *tmp* an.
3. Erstellen Sie auf dem neuen Schnittstellenrechner eine Datenbank mit dem Namen *DH*. und importieren Sie die SQL - Datei für Schnittstellenrechner dorthin.
4. Drücken Sie im Formular *Kollektiv* (Admin Bereich) oben rechts auf den Schalter **neuer**

**Schnittstellenrechner**. Füllen Sie anschließend die Felder *Server Name*, *IP oder URL*, *Benutzer*, *Passwort* und *Datenbank* aus.
5. Erstellen Sie im Formular Schnittstellen eine neue Schnittstelle, die mit dem Namen *watchdog* beginnt.

Das Entfernen eines Schnittstellenrechners aus dem Kollektiv geht ganz einfach. Wählen Sie im Formular *Kollektiv* (Admin Bereich) einfach den Schnittstellenrechner aus und drücken Sie unten im Formular den Schalter **entfernen**. Das war es fast schon. Schön sieht es aus, wenn Sie nun noch im Formular Schnittstellen die Schnittstellen, die dem ehemaligen Schnittstellenrechner zugeordnet sind, deaktivieren.

### 3.4.3. Server zum Kollektiv hinzufügen oder entfernen.

Einen Server zum Kollektiv hinzuzufügen, bedeutet etwas mehr Aufwand als bei einem neuen Schnittstellenrechner.
Verfahren Sie zunächst so, wie in Kapitel 3.4.2. beschrieben. Wählen Sie in Schritt 3 allerdings die SQL - Datei für Server. In Punkt 5 erstellen Sie noch eine weitere Schnittstelle vom Typ *comp*.

Je größer das Archiv im Kollektiv ist, umso länger dauert es, dieses Archiv auf den neuen Server zu kopieren. Also kann es sein, dass das Kopieren die meiste Zeit beim Aufsetzen eines neuen Servers in Anspruch nimmt. Bevor Sie damit starten, sollten Sie sich einen Moment Zeit nehmen, um über die Synchronisation der Server nachzudenken. Alle Server sollten die gleichen Daten vorhalten. Eine Lücke in den Daten eines Servers ist nicht besonders schön. Die meisten Daten sind historisch und werden nicht mehr geändert. Daher können alle Dateien vom Verzeichnis */var/lib/DH* von einem vorhandenen Server auf den neuen Server kopiert werden. Nur das Verzeichnis des aktuellen Monats sollten Sie dabei aussparen. Also legen Sie als Benutzer root ein neues Verzeichnis *DH* im *Ordner /var/lib* an und kopieren Sie das Archiv mit Ausnahme des aktuellen Monats auf den neuen Server. Wenn das erledigt ist, stoppen Sie die Schnittstelle *comp* auf dem Server, von dem Sie das Archiv kopiert haben. Ab jetzt werden keine neuen Daten mehr ins Archiv geschrieben. Die Daten werden stattdessen in der Tabelle *akt* gepuffert. Nun können Sie auch den Ordner des aktuellen Monats auf den neuen Server kopieren.

Führen Sie auf allen Rechnern im Kollektiv die Scripte *DH_start.py* und *DH_stop.py* aus, um die Schnittstellen neu zu starten. Das ist erforderlich, damit die Schnittstellen auch erfahren, dass sie ihre Daten auch an den neuen Server senden müssen.

Kopieren Sie den Inhalt der Tabelle *akt* von dem Server, von dem Sie das Archiv kopiert haben, in die Tabelle *akt* auf dem neuen Server. Damit dürfte die Lücke der Daten auf dem neuen Server nur wenige Minuten ausmachen. Vergessen Sie bitte nicht, die Schnittstelle *comp*, die Sie ja eben gestoppt hatten, wieder neu zu starten.

Um einen Server aus einem Kollektiv zu entfernen, verfahren Sie zunächst so, wie bereits für Schnittstellenrechner beschrieben: Server im Formular *Kollektiv* auswählen, Schalter **entfernen** drücken, Schnittstellen im Formular *Schnittstellen* deaktivieren. Nun sollten Sie noch die Scripte *DH_start.py* und *DH_stop.py* auf jedem Rechner im Kollektiv ausführen. Tun Sie das nicht, dann schreiben die Schnittstellen fleißig weiter Daten in die Tabelle akt der Datenbank. Schalten Sie den Server ab, dann schreiben sie die Daten in ihren Puffer. Den Puffer werden sie aber nie leer bekommen, da der Server ja nicht mehr existiert.

## 3.5. weitere Formulare im Admin - Bereich

### 3.5.1. Einstellungen DH

Grundsätzlich können Sie sämtliche Konfiguration auch über die Seite Einstellungen_DH vornehmen. Lassen Sie dabei aber bitte die Bezeichnungen unterhalb der Wurzel der Baumstruktur unverändert. Ansonsten funktioniert so ziemlich nichts mehr.

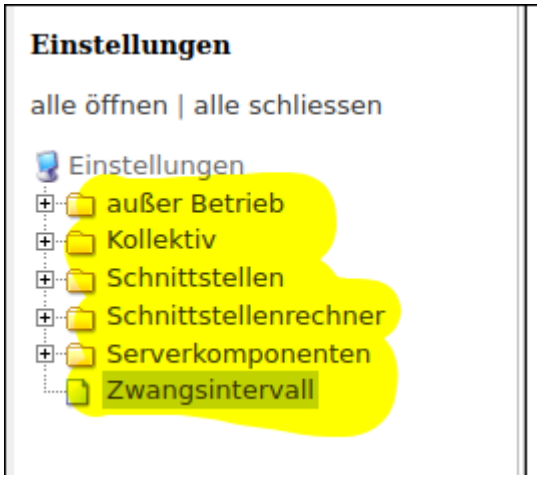


Abb. 36: Gelb markierte Bezeichnungen niemals ändern!

Erwähnenswert ist hier noch die Einstellung *Zwangsintervall*. Das hört sich zwar nach Zwang und Gewalt an, ist aber harmlos. Der *Zwangsintervall* gibt lediglich an, nach wie vielen Sekunden eine Schnittstelle auf jeden Fall einen Wert für einen Point zu den Servern schicken soll, auch wenn sich der Wert nicht geändert hat. Normalerweise werden Werte, die sich seit dem letzten Einlesen nicht geändert haben, erst gar nicht von der Schnittstelle zum Server geschickt. Nun gibt es aber Points, deren Wert sich nur selten ändert. Beispiel: Ein Preis für ein Produkt, der von irgend einer Seite im Internet gelesen wird. Die Schnittstelle fragt die Webseite zwar jede Minute ab, aber der Preis ändert sich nur 1 -2 mal am Tag. Würde es den *Zwangsintervall* nicht geben, dann wäre man sich nicht sicher, ob der Point „noch lebt“, wenn man dort einen Wert sieht, der unter Umständen schon zwei Tage alt ist. So aber weiß der Anwender, dass der Point noch mit frischen Werten beliefert wird und das der angezeigte aktuelle Wert nicht älter als 10 Minuten ist. Die Standardeinstellung für den Zwangsintervall ist 600 Sekunden.

### 3.5.2. Geräte Einstellungen

Wenn Sie möchten, dann können Sie hier gerne Einstellungen vornehmen, allerdings existieren bisher nur einige Prototypen dieser Geräte. Die Serienproduktion, und damit auch die Verfügbarkeit der Geräte, ist derzeit noch nicht geplant (Stand: Ende 2023).
Bei den Geräten handelt es sich um sparsame Mikroprozessoren mit eingebautem WLAN Modul. An diese Geräte lassen sich 1-wire - Sensoren anschließen. Sie verfügen zudem über einen analogen und mehrere digitale Eingänge. Betrieben werden sie entweder mit einem 5V Netzteil, oder über ein Akku-Pack, welches während des Betriebs ausgetauscht werden kann. Ein Akku-Pack reicht, je nach Häufigkeit der Datenübertragung, für einen Betrieb des Gerätes von bis zu vier Wochen aus. Die 1-wire - Sensoren sind mit Steckern ausgestattet. Damit lassen sie sich schnell und problemlos an die Geräte anschließen. Außerdem sind Verlängerungskabel und Verteiler für die 1-wire - Sensoren vorgesehen. Die Geräte sind dafür gedacht, vorübergehend Daten von einem Ort zu sammeln und zum DH zu schicken oder auch dauerhaft von einem Ort, an dem kein Netzwerkkabel verfügbar ist, Daten zu senden. In letzterem Anwendungsfall ist es aber bequemer, wenn wenigstens eine Stromquelle in der Nähe verfügbar ist. Alle zwei bis vier Wochen das Akku-Pack zu tauschen kann lästig werden.

### 3.5.3. Bausteine verwalten

Hier lassen sich ebenso, wie bereits im Entwicklerhandbuch unter 2.3.1. beschrieben, Bausteine für Bilder erstellen. Der Unterschied ist, dass Sie hier Bausteine erstellen, die für jeden Anwender sichtbar sind.

### 3.5.4. Meldungen

Zeigt die Tabelle *Meldungen* an. Diese Tabelle enthält alle Nachrichten, die darauf warten, von den Schnittstellen *watchdog* abgeholt zu werden. Diese Tabelle sollte möglichst leer sein. Finden Sie hier jedoch Meldungen mit einem Zeitstempel, der älter als eine Minute ist, dann läuft irgend ein *watchdog* nicht mehr. Bitte zuerst den Zustand der Schnittstelle *watchdog* prüfen, bevor Sie hier etwas löschen.

### 3.5.5. Pufferstände

Hier werden die Pufferstände aller Rechner im Kollektiv angezeigt. Die Pufferstände zeigen gewöhnlich 0 an. Sollte das nicht der Fall sein, dann kann der betreffende Rechner irgend einen Server nicht erreichen, oder im Puffer steht Unsinn. Das ist meistens der Fall, wenn Sie dort nur sehr wenige Einträge

sehen. Über die Schaltflächen, auf denen die Namen der Rechner stehen, können Sie sich den jeweiligen Inhalt des Puffers anzeigen lassen. Finden Sie dabei unsinnige Einträge, dann diese bitte löschen. Ansonsten versucht der Rechner jede Minute verzweifelt den Inhalt des Puffers los zu werden.

### 3.5.6. Log

Schnittstellen schreiben nicht nur dann in die Log - Tabelle, wenn sie ausfallen oder abgeschaltet werden. Je nachdem, wie sie programmiert sind, schreiben sie hier auch Infos rein, die sonst irgendwie relevant sind.

# 4. DH_SMT

*DH_SMT* ist ein Python Programm mit GUI. Es dient zum direkten Bearbeiten der Archive und ist mit etwas Vorsicht zu genießen. Sie finden es gewöhnlich im Ordner */opt/DH/DH_SMT*. Starten Sie dort die Datei **DH_SMT.py**.
Die Konfiguration für Verbindungen zu den Servern steht im Ordner */home/USERNAME*. Dort befindet sich ein versteckter Ordner (*./DH*), welcher eine Datei mit dem Namen *DH_SMT.ini* enthält. Die Einstellungen für die Verbindungen zu den Servern können Sie entweder hier direkt vornehmen oder aber über die Registerkarte *Verbindungen* in der Anwendung.

## 4.1. Verbindungen

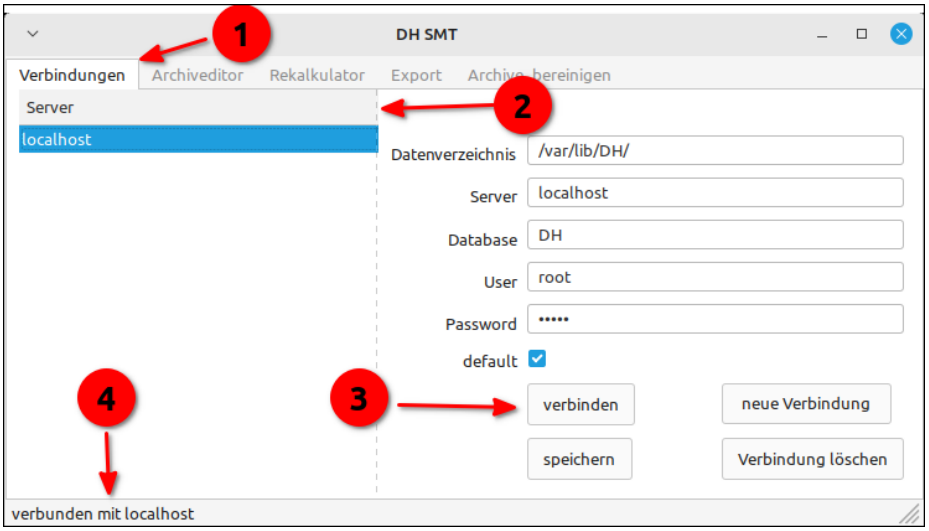


Abb. 37: Registerkarte Verbindungen

Die erste Karte, Verbindungen (1), listet auf der linken Seite alle verfügbaren Verbindungen zu Servern auf (2). Dabei spielt es keine Rolle, ob diese Server dem gleichen Kollektiv angehören oder nicht. Auf der rechten Seite befindet sich ein Schalter, über den man weitere Verbindungen anlegen kann. Wählen Sie auf der linken Seite einen Server aus, dann werden auf der rechten Seite die Verbindungsdaten angezeigt. Die Verbindung wird per Klick auf den Schalter **verbinden** (3) zu dem ausgewählten Server hergestellt. Mit welchem Server die Anwendung momentan verbunden ist, sehen Sie stets in der Statuszeile unten (4).

**wichtig:**
Falls Sie eine Verbindung zu einem entfernten Server herstellen, müssen Sie den kompletten Pfad, ausgehend vom lokalen Rechner, als Datenverzeichnis angeben. Es muss vor dem Herstellen einer Verbindung sichergestellt sein, dass das Datenverzeichnis auch erreichbar ist. Hierzu muss das Dateisystem, oder zumindest der benötigte Teil davon, per NFS gemountet sein.

## 4.2. Archiveditor

Manchmal kommt es vor, dass Werte ins Archiv geschrieben werden, die nicht plausibel sind. Das ist meistens nicht schlimm. Werden diese Werte aber irgendwo für Berechnungen verwendet, dann ist das nicht mehr trivial. Im Archiveditor auf der zweiten Registerkarte können die Werte korrigiert oder gelöscht werden. Der Archiveditor korrigiert dabei auch die Mittelwerte, vt und vt_interpol.

**Beispiel:**
Für den Luftdruck, welcher gewöhnlich zwischen 970 und 1040 mbar liegt, wird zweimal nacheinander eine 0 ins Archiv geschrieben. Da für diesen Point auch die Mittelwerte berechnet werden, hat das auch Auswirkungen auf die hMW-, dMW-, hMin- und dMin- Werte.

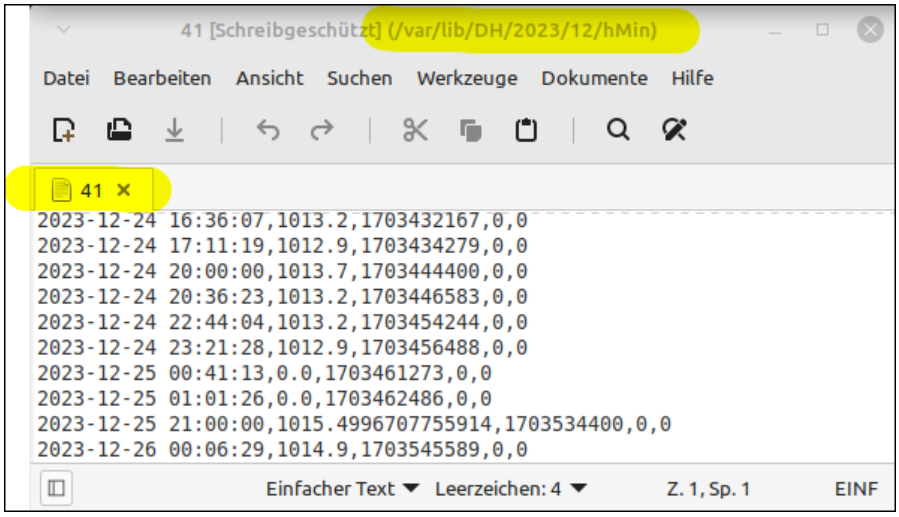


Abb. 38: Auch die hMin - Werte sind fehlerhaft.

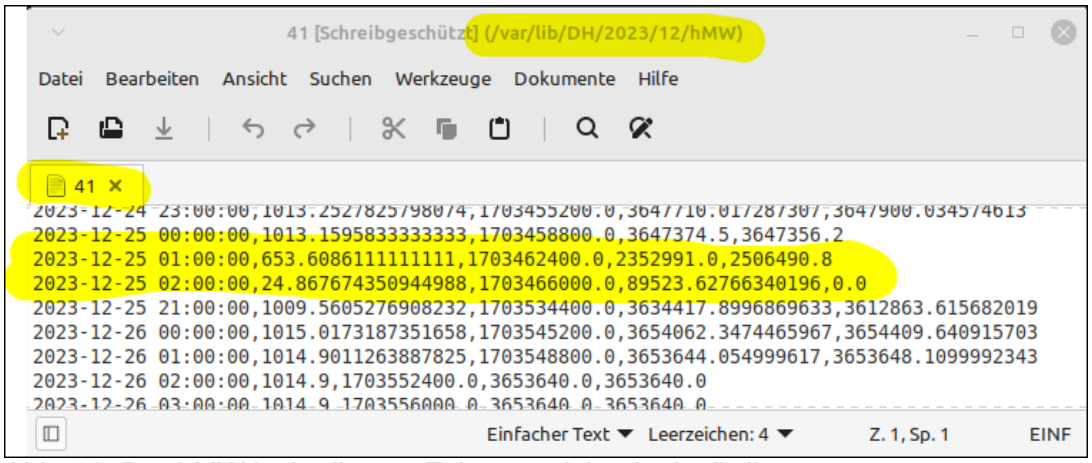


Abb. 39: Der hMW ist in diesem Zeitraum nicht glaubwürdig.

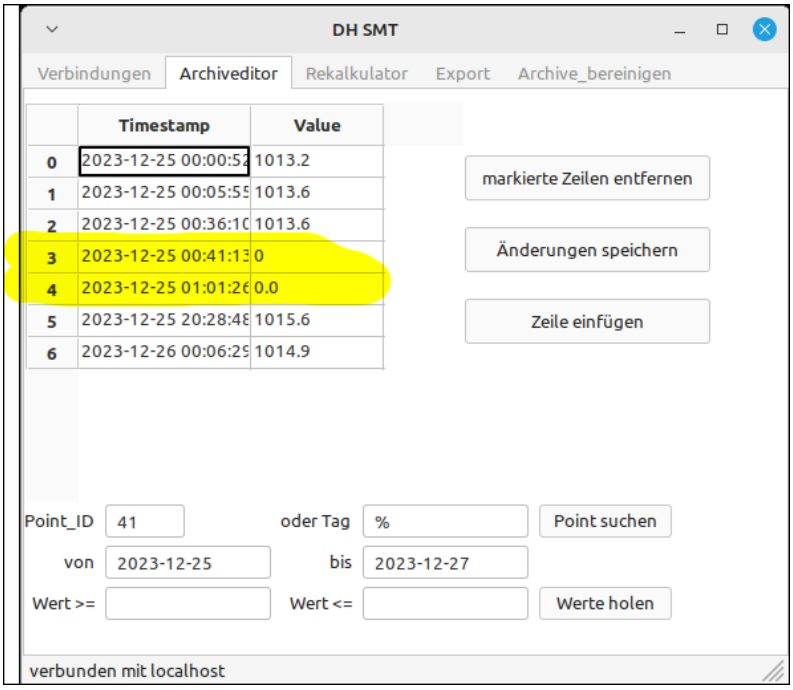


Abb. 40: Archiveditor mit fehlerhaften Werten (Point_ID 41).

Jetzt werden die unglaubwürdigen Werte durch plausible Werte ersetzt.

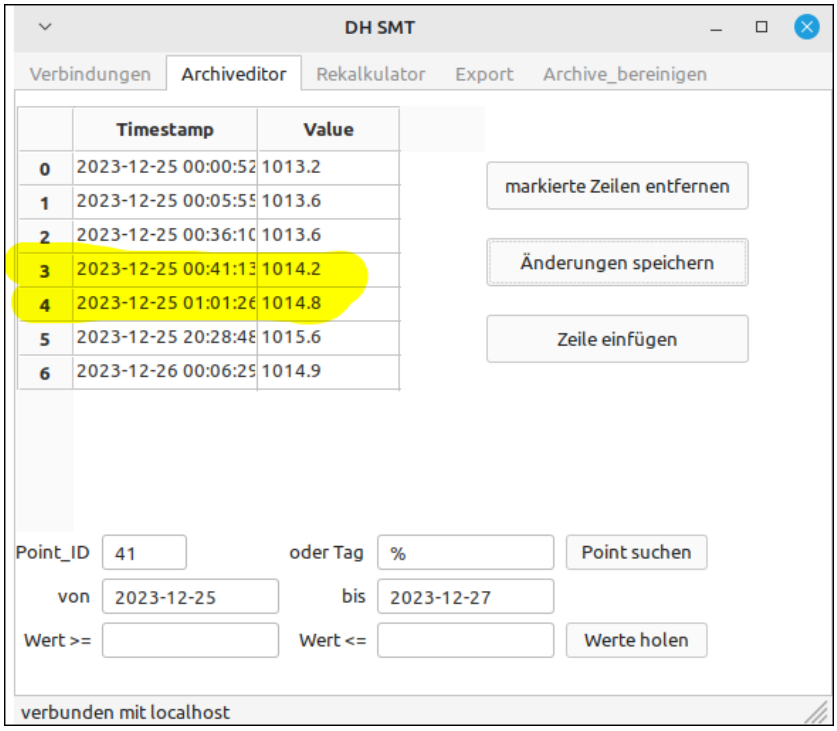


Abb. 41: Archiveditor mit korrigierten Werten (Point_ID 41).

Die Werte für den vt und den vt_interpol haben sich ebenfalls deutlich geändert. Rechnet man in einer Tabellenkalkulation mit dem vt oder dem vt_interpol, dann hat das deutliche Auswirkungen.

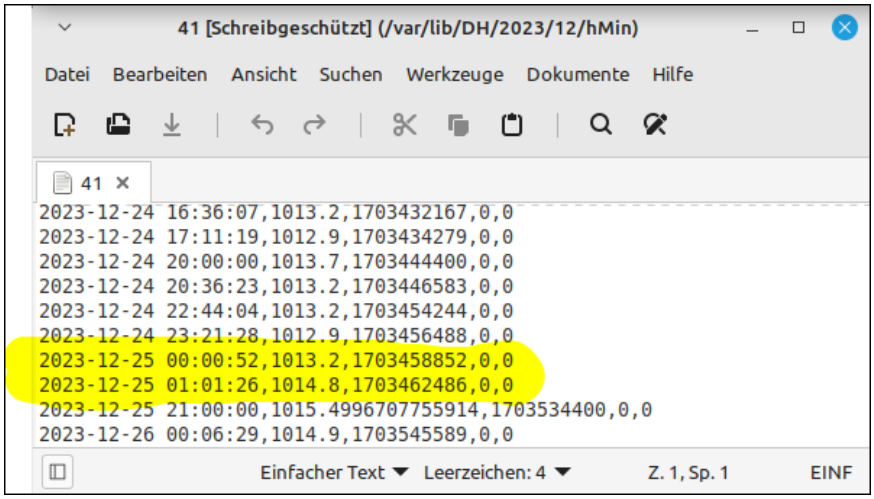


Abb. 42: neue hMin - Werte

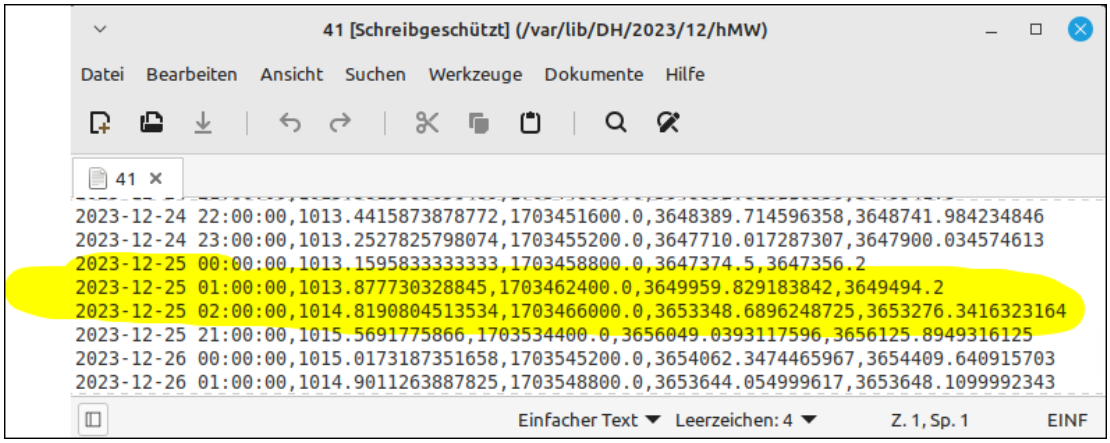


Abb. 43: neue Stundenmittelwerte mit geändertem vt und vt_interpol.

Besser als die manuelle Korrektur ist es jedoch, wenn die Schnittstelle eine Plausibilitätsprüfung vornimmt und somit erst gar keinen Datenmüll zum DH schickt.

## 4.3. Rekalkulator

Auf dieser Registerkarte können ein oder mehrere Points der Schnittstelle **calc** neu berechnet werden. Dabei werden zuerst alle Werte dieser Points für den angegeben Zeitraum aus dem Archiv gelöscht und dann neu berechnet. Je mehr Points Sie neu berechnen wollen und je länger der gewählte Zeitraum ist, umso länger müssen Sie warten.

**Achtung!**
Schauen Sie sich bitte vor der Neuberechnung die Formel eines jeden Points an. Sie finden den zu berechnenden Ausdruck in der Pointkonfiguration im Feld *Info*. Steht dort irgendwo *time.time()*, dann wird an dieser Stelle stets der aktuelle Unix - Zeitstempel zur Berechnung verwendet. Wenn Sie einen solchen Point neu berechnen, dann erhalten Sie grundsätzlich ein unsinniges Ergebnis.

Anders sieht es bei Ausdrücken aus, die die DH - Funktion *akt()* verwenden. Bei der Rekalkulation wird die Funktion durch *AW()* ersetzt. Dabei verwendet *AW()* den Zeitstempel der der aktuellen Zeit zu dem Berechnungszeitpunkt entspricht und simuliert somit die Funktion *akt()* zu diesem Zeitpunkt.

## 4.4. Export

Es kommt öfters mal vor, dass man als Admin von den Anwendern gefragt wird, ob sie die Werte über einen langen Zeitraum eines Points in einer Tabelle bekommen können. Grundsätzlich steht jedem Anwender eine Exportfunktion zur Verfügung, welche die Werte als csv - Datei ausgibt. Der Export kann allerdings etwas Zeit in Anspruch nehmen. Bequemer ist es dann, die Werte über die Registerkarte **Export** in eine Datei zu schreiben. Auch wenn man einen Point komplett aus dem DH entfernen möchte, ist der Weg über diese Registerkarte eine gute Option, um sämtliche Werte eines Points irgendwohin zu schreiben, bevor man die Werte endgültig aus dem Archiv löscht. Beim Export der Werte in ein anderes System ist dieser Export ebenfalls sehr nützlich.

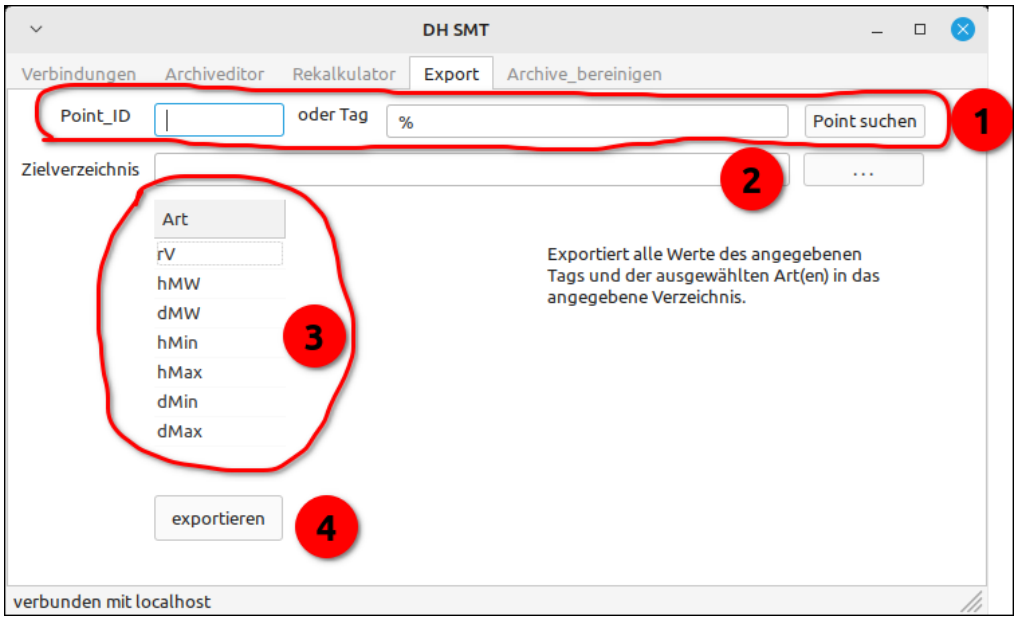


Abb. 44: Registerkarte Export

Geben Sie entweder die gewünschte Point_ID direkt an oder suchen Sie sie über das Suchfeld (1). Tragen Sie das Verzeichnis ein, in das Sie die Werte schreiben möchten (2). Wählen Sie nun noch aus, ob Sie die Rohwerte (rV) und / oder die statistischen Werte exportieren möchten (3). Mit einem Klick auf den Schalter **exportieren** erhalten Sie sehr schnell alle Werte des Points in einer csv - Datei (4).

## 4.5. Archive bereinigen.

Leider kommt es hin und wieder vor, dass Archivdateien nicht sauber geschrieben wurden. Es können am Anfang oder am Ende der Dateien leere Zeilen stehen. Es ist auch schon vorgekommen, dass sich Werte aus dem Vormonat zwischen die Werte des Folgemonats eingeschlichen haben.
Wie auch immer, es schadet nie, wenn man mal zwischendurch die Archive bereinigt. Die Daten werden im angegebenen Zeitraum sortiert. Leere Zeilen werden entfernt und Einträge aus anderen Monaten werden gelöscht. Einzig die erste Zeile jeder Datei enthält den letzten Wert des Vormonats und das soll

auch so sein.

Hinweis:
Wenn ein Trend etwas seltsam aussieht, weil er eigentlich die Werte eines Tages anzeigen soll, aber stattdessen die Zeitachse sich über mehrere Wochen erstreckt und die Trendlinie zwischendurch einen Ausflug an den Anfang der Zeitachse macht, dann haben sich höchstwahrscheinlich Werte aus dem Vormonat eingeschlichen. Bereinigen Sie einfach die Archivdateien des laufenden Monats und alles ist wieder schön.