IIoT Building Blocks

Collect

iT Engineering Software Innovations GmbH

Table of contents

1. Einführung

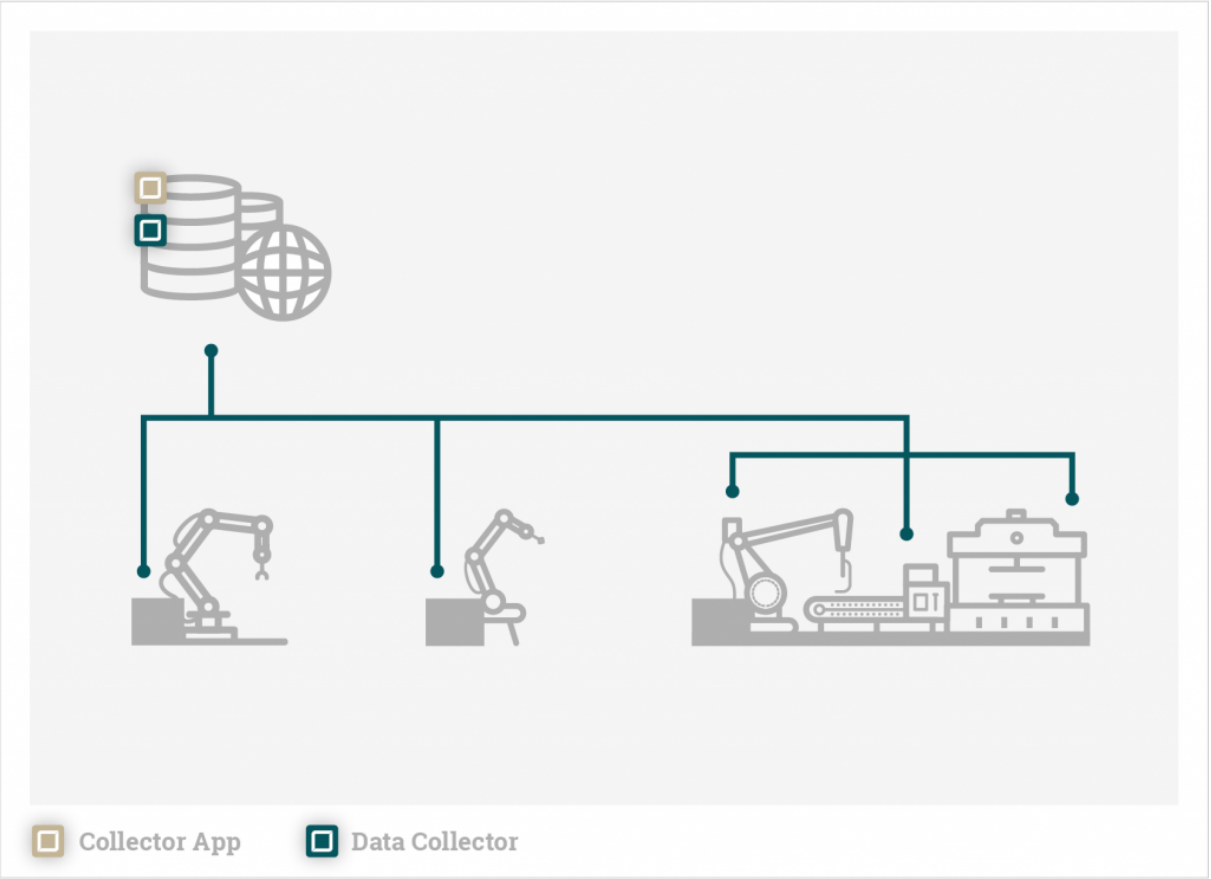
1.1 Deployment Varianten

Die Komponenten des IIoT Building Block Stacks sind sehr modular und an eine Microservice-Architektur angelehnt. Daher ist das Deployment so flexibel wie Möglich und fügt sich sehr gut in bestehende Infrastrukturen.

1.1.1 Ein Server Installation

Der gesamte Stack wird auf einem Maschinen Rechner/Server installiert:

- Collector
- Collector App
- Datenbank

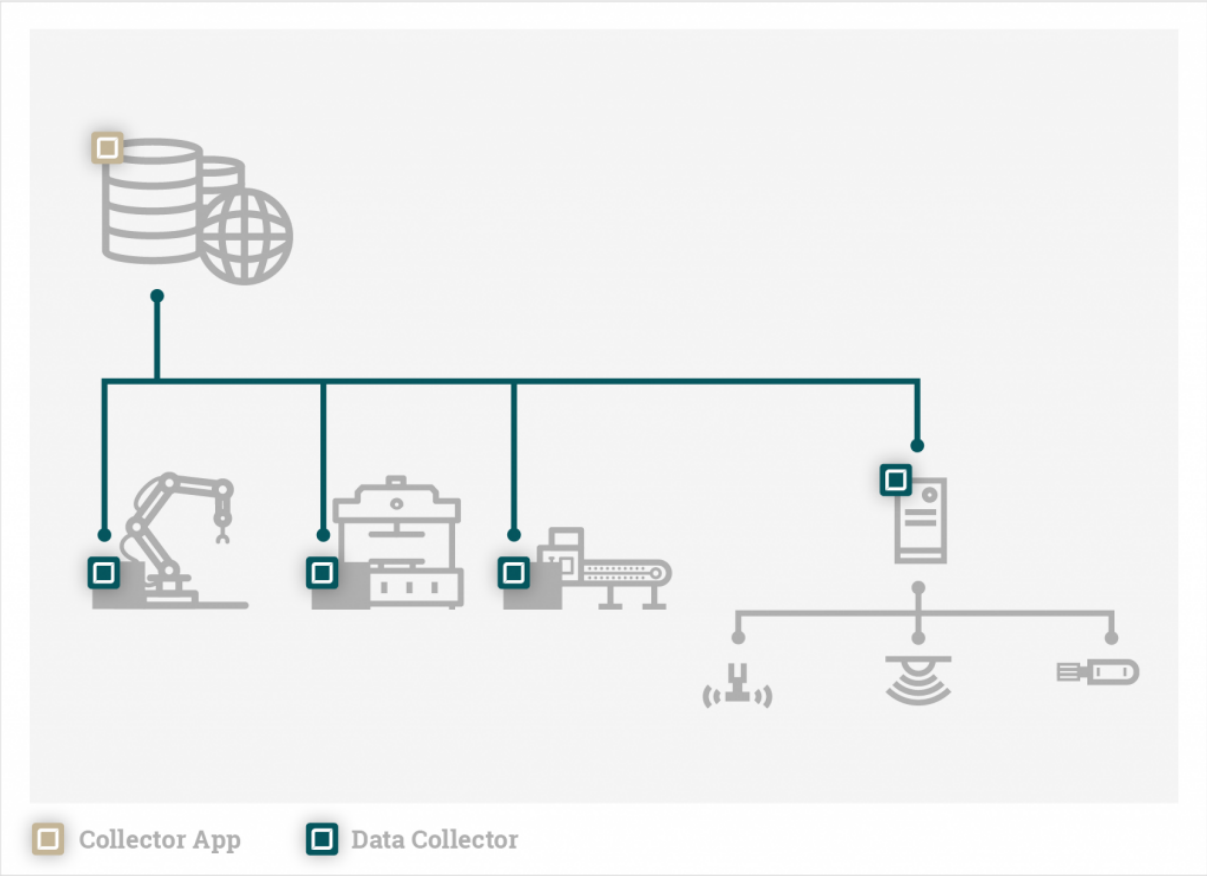


Der Collector sammelt die Daten von verschiedenen Maschinen und Anlagen über eine Netzwerkschnittstelle und speichert sie auf der zentralen Datenbank die neben dem Collector auf dem gleichen Rechner installiert ist. Über die Collector App, ebenfalls auf dem gleichen Server muss noch ein Collector konfiguriert werden.

Ist das Netzwerk gestört können auch keine Daten übertragen werden. Daten können unter Umständen verloren gehen.

1.1.2 Verteiltes System

Es können mehrere Collector Instanzen im Netzwerk installiert werden. Auf PC basierten Steuerungen kann der Collector sogar direkt mit laufen. Die Collector App sowie die Datenbank befinden sich auf einem Zentralen Maschinen Rechner/Server.



In der Collector App können alle Collectoren im Netzwerk über eine Benutzeroberfläche konfiguriert werden. Die Collectoren können im Falle einer Netzwerksstörung die Daten puffern und nach Behebung der Störung suggestive übers Netzwerk an die Datenbank senden.

1.2 Getting Started

Für einen schnellen Einstig empfiehlt es sich zunächst den gesamten Stack auf einer Maschine zu installieren.

1.2.1 Schnellstart mit docker-compose

Der einfachste und zuverlässigste Weg den Stack zu deployen ist über Docker, da die Applikationen in definierten Container-Umgebungen laufen und somit keine Abhängigkeiten zum Host-System haben. Zudem können die Container nach dem Beenden restlos vom System wieder entfernt werden.

Installation von Docker (Linux, Mac und Windows): Get-Docker

Für die Windows- und Mac-Docker-Installation wird eine Lizenz benötigt. Als Alternative kann auch Rancher-Desktop als Container-Runtime installiert werden. Falls Rancher-Desktop mit containerd verwendet wird muss im folgenden die docker cli durch nerdctl ersetzt werden.

Um mit einem Befehl direkt mehrere Container starten zu können wird Docker Compose verwendet. In einer docker-compose Datei können mehrere Container mit Konfiguration und Verknüpfung definiert werde. In diesem Fall sind es folgende Container:

- Collector
- Collector App
- OPC UA Browser
- InfluxDB
- Umati OPC UA Server
- Grafana

Stellen Sie daher sicher, dass Sie die Images für Collector, Collector-App und OPC UA Browser gemäß der Installationsanleitung geladen haben.

Das InfluxDB und Grafana Image wird automatisch aus dem offiziellen online Repository geladen. Dabei dient die InfluxDB als demo Datensenke und Grafana zum darstellen von vom Data Collector gespeicherten Daten.

1. Erstellen eines Ordners

```
mkdir iiot-bb-stack
cd iiot-bb-stack
```

2. Erstellen des Shard Volume Ordner

Da Container keinen persistenten Speicher besitzen, müssen sämtliche persistente Dateien und Datenbanken auf dem Host-System gespeichert werden, damit nach dem Container neustart der Zustand nicht verloren geht.

```
mkdir docker
```

Der Ordner wird in den Container-Volumes verwendet.

3. Erstellen der compose Datei

Linux

```
touch docker-compose.yaml
```

Windows

```
New-Item -Path . -Name "docker-compose.yaml" -ItemType "file"
```

Die Datei mit einem beliebigen Editor öffnen und folgenden Inhalt einfügen:

docker-compose.yaml

```
version: "3.4"
services:
  collector:
    image: ite-si/collector:vx.x.x
    volumes:
      - ./docker/collector:/opt/ite-si/collector/tmp
    links:
      - influxdb
    environment:
      - COLLECTOR_WORK_DIR=/opt/ite-si/collector
      - COLLECTOR_CONFIG_DIR=/opt/ite-si/collector/tmp
      - COLLECTOR_LOG_DIR=/opt/ite-si/collector/tmp/logs
      - COLLECTOR_CONSOLE_LEVEL=warn

  opcua-browser:
    image: ite-si/opcua-browser:vx.x.x
    volumes:
      - ./docker/collector/certificates:/opt/ite-si/opcua-browser/certificates

  collector-app:
    image: ite-si/collector-app:vx.x.x
    ports:
      - "4000:4000"
    environment:
      - COLLECTOR_APP_HOST=0.0.0.0
      - COLLECTOR_APP_PORT=4000
      - COLLECTOR_APP_TOKEN_SECRET=asdfasdfjklasdcpwerasdfvuewardciadioweroqwer
      - COLLECTOR_APP_ENCRYPTION_KEY=e44c966f21b9e1577802464f8924e6a37e3e9751fa01304213b2f845d8841d61
      - COLLECTOR_APP_CORS_ORIGIN=http://localhost:4000
      - COLLECTOR_APP_PUBLIC_GRAPHQL_URL=http://localhost:4000/api/graphql
      - COLLECTOR_APP_PUBLIC_WS_URL=ws://localhost:4000/api/subscriptions
    volumes:
      - ./docker/collector-app:/var/lib/collector-app

  grafana:
    image: grafana/grafana:latest
    ports:
      - "3000:3000"
    links:
      - influxdb
    volumes:
      - ./docker/grafana:/var/lib/grafana

  opcua-server:
    image: ghcr.io/umati/sample-server:main
    links:
      - collector
      - opcua-browser
```

```
  influxdb:
    image: influxdb:1.8
    volumes:
      - ./docker/influxdb:/var/lib/influxdb
```

4. Starten des Stacks

```
docker-compose up
```

oder

```
nerdctl compose -f docker-compose.yaml up
```

Zum starten im Hintergrund:

```
docker-compose up -d
```

oder

```
nerdctl compose -f docker-compose.yaml up -d
```

5. Collector App aufrufen

Nach dem Start ist die App im Browser unter http://localhost:4000/collector-app erreichbar.

Standard Login:

Benutzername	Passwort
admin	admin

Einen Einstieg in die Collector App finden Sie hier.

6. Stoppen/Entfernen des Stacks

```
docker-compose down
docker-compose rm
```

oder

```
nerdctl compose -f docker-compose.yaml down
nerdctl compose -f docker-compose.yaml rm
```

1.2.2 Schnellstart unter Windows (nativ)

1. Installer

Zunächst müssen die Komponenten:

- Collector
- Collector-App
- OPC UA Browser

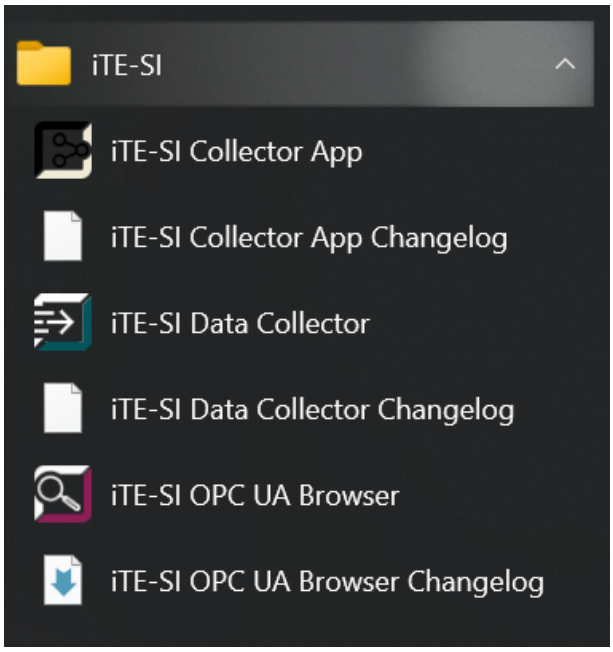
nach der Windows Installationsanleitungen über den Installer installiert werden.

Als Demo Output kann die InfluxDB installiert werden.

Als Demo Input wird der OPC UA Sample Server von Unified Automation empfohlen.

2. Starten der Komponenten

Die Komponenten lassen sich über das Windews Startmenü starten:



Nachdem alle drei Komponenten gestartet wurden, erscheint ein Windows Tray Icon in der Taskleiste. Hier kann jeweils der Log angezeigt werden und die Applikation wieder beendet werden.

3. Collector App aufrufen

Nach dem Start ist die App im Browser unter http://localhost:4000/collector-app erreichbar.

Standard Login:

Benutzername	Passwort
admin	admin

Einen Einstieg in die Collector App finden Sie hier.

2. Lizensierung

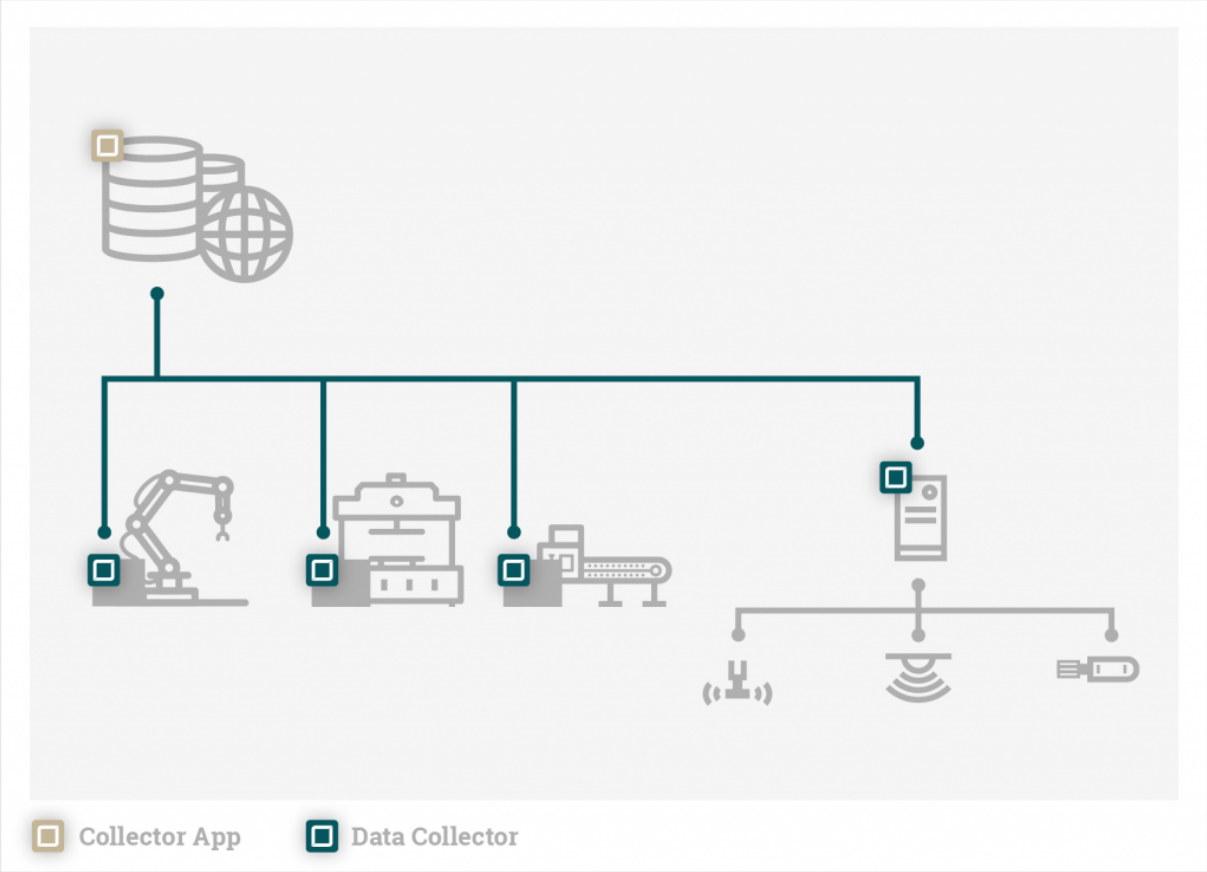
Die einzelnen Verbindungen von einem Collector werden lizensiert. Schreibt ein Collector Daten von einem OPC UA Server in eine Influx-Datenbank, dann braucht er eine Lizenz für den OPC UA Server und eine Lizenz für die Influx Datenbank.

Hierbei gibt es zwei unterschiedliche Formen der Lizensierung: Kauf-Version und Abo-Version. Details zu den einzelnen Versionen sind unter Preismodelle zu finden. Bei der Kauf-Version werden verbindungstyp-spezifische Lizenzen gekauft, die vom Collector zeitlich unbegrenzt nutzbar sind. Bei der Abo-Version werden verbindungstyp-unabhängige Blöcke gekauft, welche in unterschiedlichen Verbindungsarten flexibel verwendet werden können. Die Blöcke gelten nur im Abo und können nicht mehr verwendet werden, sobald die Subscription ausgelaufen ist.

Jeder Collector kann entweder im Abo-Modus oder im Kauf-Modus betrieben werden. Ein Mischbetrieb, d.h. ein Collector im Abo-Modus und ein anderem im Kaufmodus, ist im Netzwerk problemlos möglich.

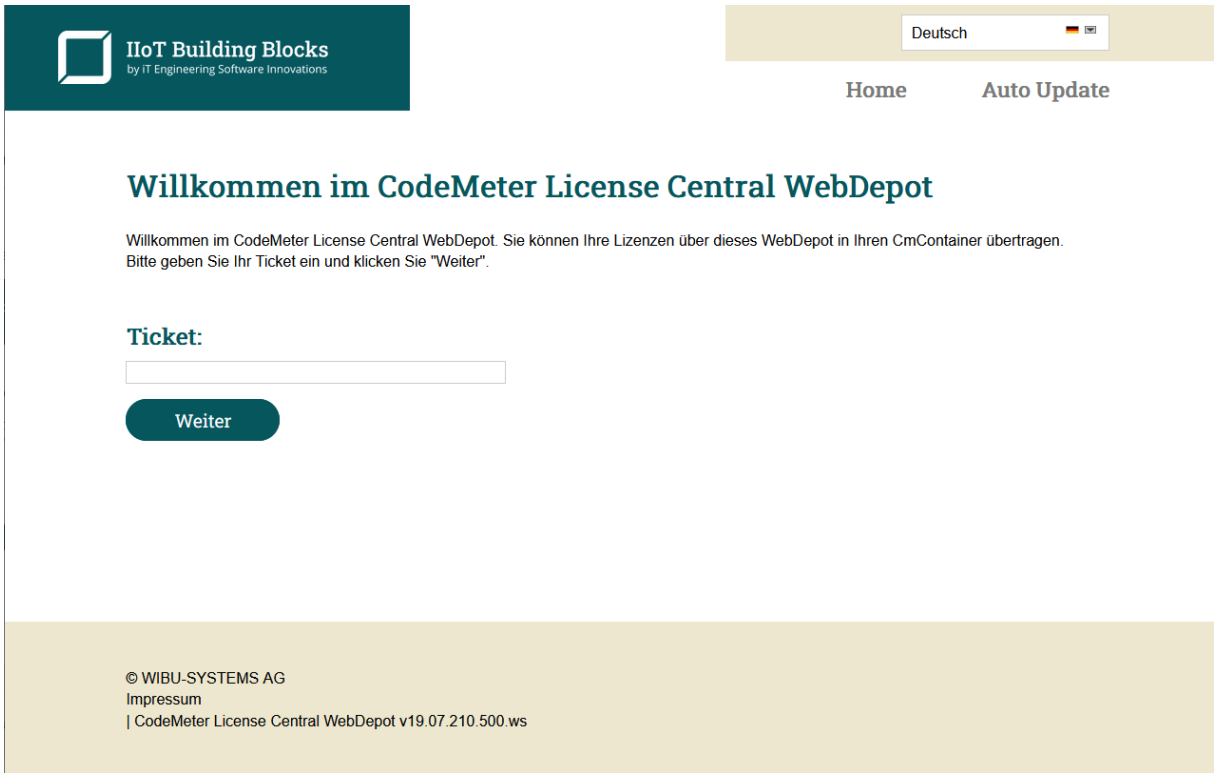
2.1 Installation von Lizenzen

Die Collector-Lizenzen (Kauf- und Abo-Version) werden als "Floating Network Licenses" installiert. Ein Collector sucht im lokalem Netzwerk nach entsprechenden Lizenzen. Dies ermöglicht es, viele Collectoren mit einem Lizenzserver betreiben zu können.

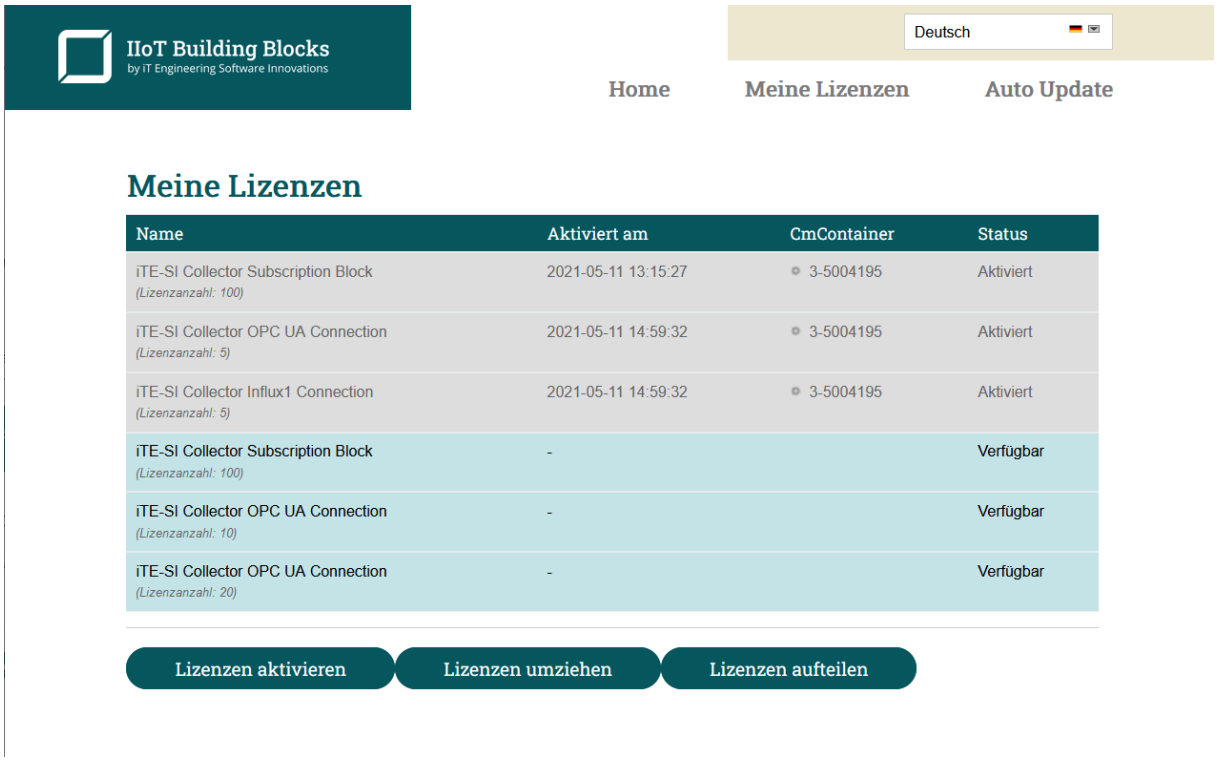


Im obigen Bild können alle Lizenzen für die Collectoren auf dem Server installiert werden, auf dem die Collector App läuft. Alternativ kann auf jedem Endgerät die passenden Lizenzen installiert werden.

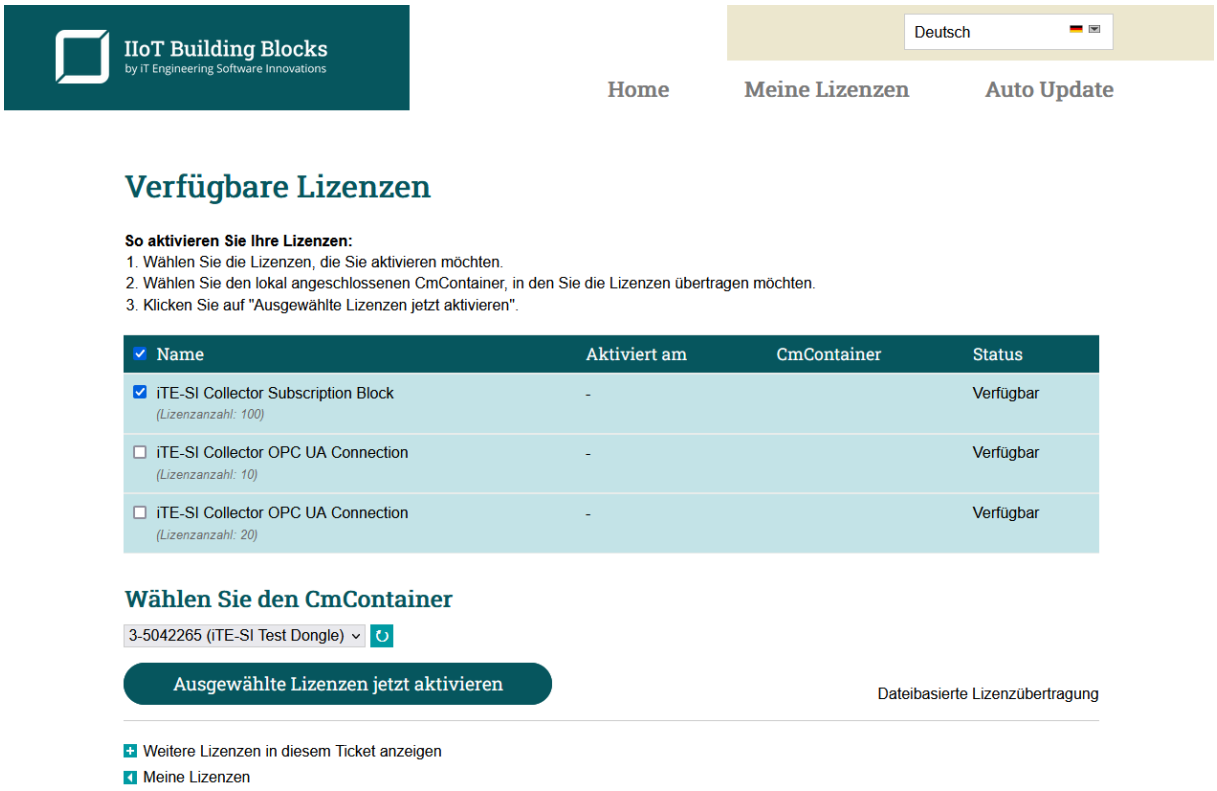
Die Lizenzen können über das Lizenzportal abgerufen werden. Dieses von dem PC aufgerufen werden, auf dem die Lizenzen installiert werden sollen.



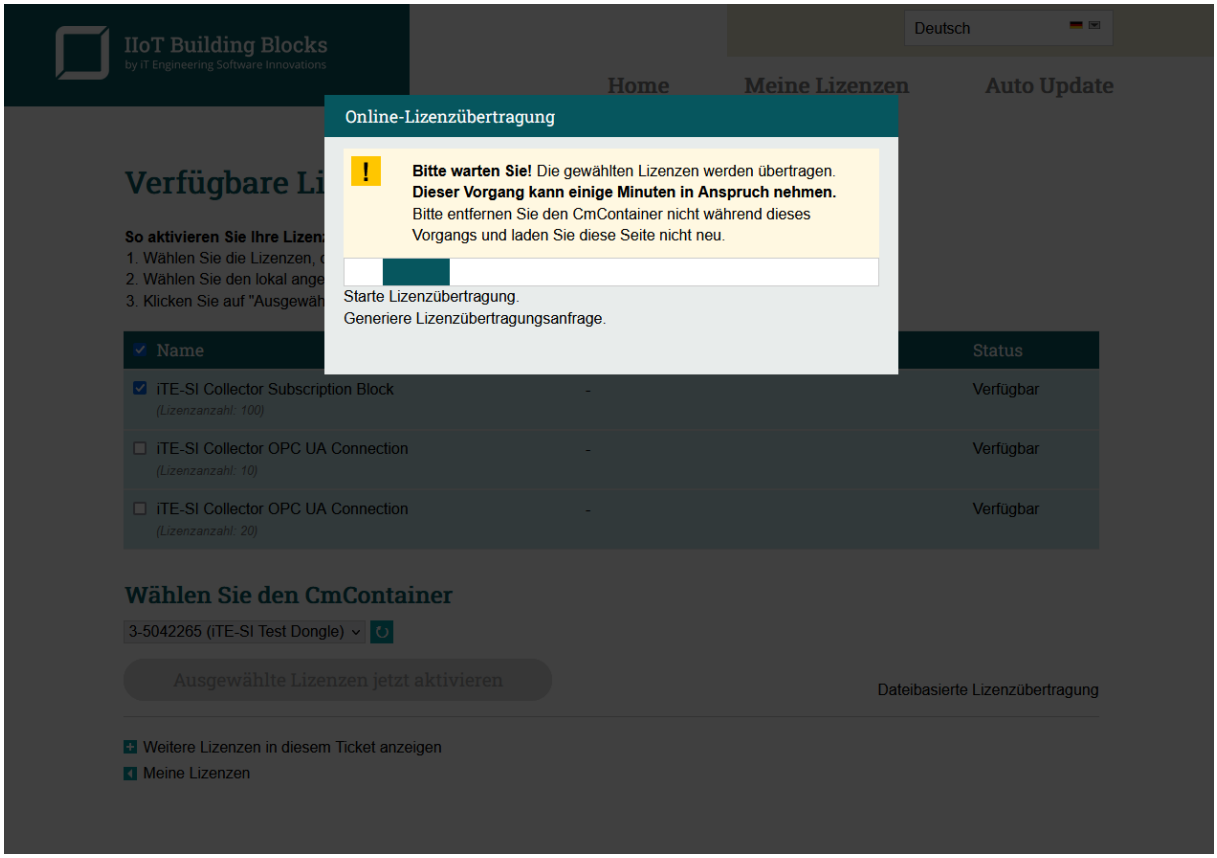
Über die Ticket-Nummer (wird bei der Bestellung mitgeliefert) kommt man zur Übersicht der Lizenzen



Lizenzen können aktiviert, umgezogen oder aufgeteilt werden.



Es können gleichzeitig mehrere Lizenzen installiert werden.

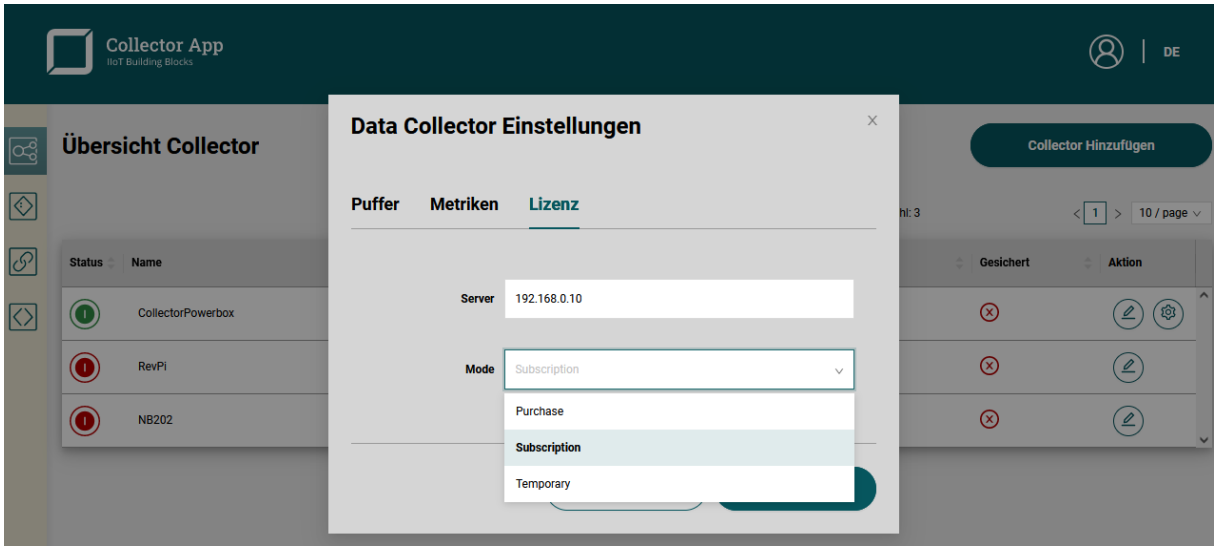


2.2 Funktionsweise bei fehlenden Lizenzen

Jeder Collector funktioniert zeitlich begrenzt auch ohne Lizenz für 6 Stunden. Nach Ablauf der Zeit werden die Verbindungen terminiert, für die es keine gültigen Lizenzen gibt.

2.3 Lizenz-Konfiguration in der Collector App

Für jeden Collector können über die Collector App der Lizenzmodus und der Lizenz-Server eingestellt werden.



Normalerweise muss der Lizenz-Server nicht konfiguriert werden, da der Collector den Lizenz-Server automatisch findet. Es kann unter gewissen Umständen (Virtualisierung, VPN, ausgeschaltetes UDP-Broadcast im lokalen Netz, etc.) dazu kommen dass der Server nicht gefunden wird. Hier kann der Hostname oder die IP-Adresse des Server eingetragen werden.

Der Collector kann entweder im Kauf-Model, Abo-Model, oder Test-Betrieb eingestellt werden. Im Test-Betrieb wird der Collector keine Lizenzen abrufen, selbst wenn im Netzwerk Lizenzen vorhanden sind. Im Kauf- oder Abo-Modell werden die entsprechenden Lizenzen gesucht.

3. iTE Collector App

3.1 Allgemeines

Die Collector App ermöglicht mit ihrer Bedienoberfläche eine einfache Einrichtung des Systems, Konfiguration von Dateninputs und outputs. Die Daten verschiedener Produktionsmaschinen und -anlagen können somit übersichtlich zusammengeführt werden.

3.2 Installation

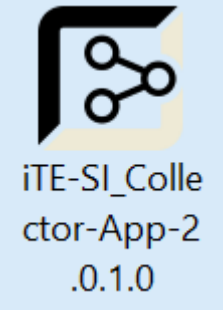
3.2.1 Windows

Installer herunterladen

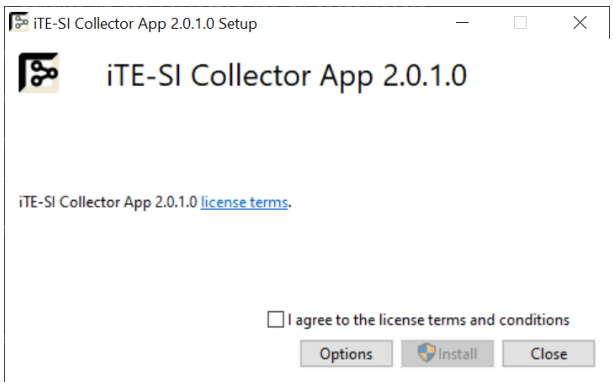
Der Windows Installer kann hier heruntergeladen werden: IIoT Building Blocks Downloads

Installieren

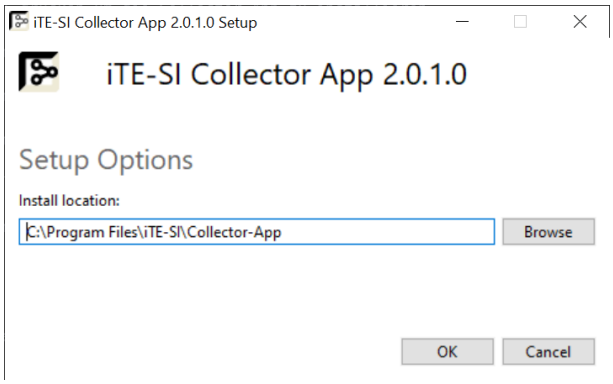
1. Den Installer ausführen.



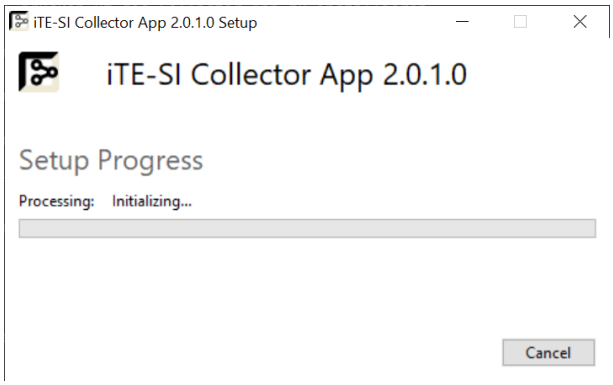
2. Die Endbenutzer-Lizenzvereinbarung lesen, gegebenenfalls akzeptieren und auf `Install` drücken. Ansonsten auf Cancel drücken um die Installation abzubrechen.



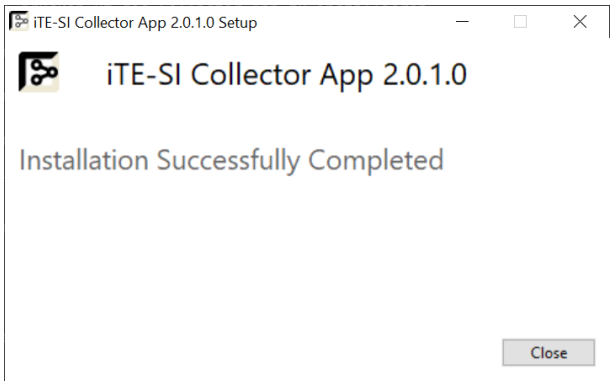
Optional: Auf Options drücken um den Speicherort zu wählen.



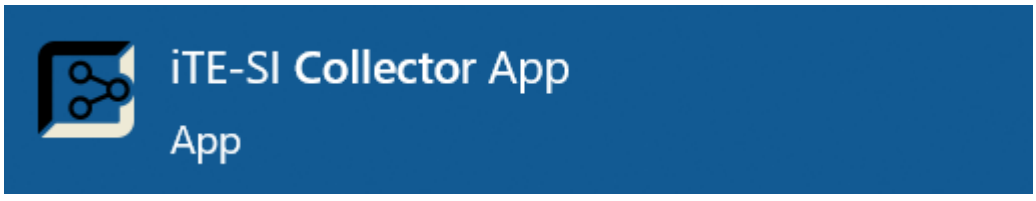
3. Kurz warten.



4. Die Installation wurde erfolgreich abgeschlossen. Auf Close drücken.



5. Die Collector App kann man jetzt über das Startmenü starten.



3.2.2 Docker

Das Docker Image kann hier herunterladen werden: IIoT Building Blocks Downloads

1. Laden des Images

> **Info**
>
> x.x.x.x muss durch die heruntergeladene Version ersetzt werden!

> **Info**
>
> Falls rancher-desktop mit containerd vernwedet wird, muss docker durch nerdctl ersetzt werden.

```
docker load -i iTE-SI_Collector-App-x.x.x.x.tar.gz
```

2. Container starten:

```
docker run -p 4000:4000 ite-si/collector-app:vx.x.x
```

Die Collector App kann über Environment-Variablen konfiguriert werden. Zum Beispiel:

```
docker run -p 4000:4000 -e COLLECTOR_APP_CORS_ORIGIN="https://hostname" -e COLLECTOR_APP_PUBLIC_GRAPHQL_URL="https:/
```

Mehr zur Konfiguration siehe: Collector App Konfiguration

Um die Collector-App-Einstellungen persistent zu Speichern, muss der Ordner mit der SQLite-Datenbank im Container auf einen Ordner im Hostsystem gemappt werden.

```
docker run -p 4000:4000 -v .config/collector-app:/var/lib/collector-app ite-si/collector-app:vx.x.x
```

3.3 Konfiguration

3.3.1 Parameter

Folgende Parameter können für die Collector App konfiguriert werden:

Parameter	Beschreibung	Default
host	Die IP Adresse des Interface über das der Webserver der App zur Verfügung gestellt werden soll. Wird `localhost` gewählt, ist die App nur von dem Server, auf dem sie installiert wurde, erreichbar. Standardmäßig bindet die App an alle Interfaces und ist somit von außen erreichbar.	0.0.0.0
port	Der Port auf welchem der Webserver der App laufen soll	4000
tokenSecret	Der Schlüssel, mit dem der User Cookie signiert wird. Die mindestlänge beträgt 32 Zeichen.	-
encryptionKey	Der Schlüssel, mit dem Collector-Api-Tokens verschlüsselt und Benutzer-Passwörter gehashed in der Datenbank gespeichert werden. Die mindestlänge beträgt 32 Zeichen.	-
corsOrigin	Die IP Adressen oder Domains unter denen die Collector App über das Netzwerk erreichbar ist. Dies dient der Sicherheit. Anfragen von Web-Seiten mit einer andren IP oder Domain werden geblockt. Mehrere IP Adressen oder Daomains werden über ein `;` getrennt	http://localhost:4000
apiURL	Die Url zum Collector App Graphql-Server. Diese Konfiguration ist für das das Frontend. Das Backend startet die API stets unter der Route `/api/graphql`	http://localhost:4000/api/graphql
wsUrl	Die Url zum Collector App Web-Socket. Diese Konfiguration ist für das das Frontend. Das Backend startet den Web-Socket stets unter der Route `/api/subscriptions`	ws://localhost:4000/api/subscriptions
database	Es können verschiedene SQL Datenbanken konfiguriert werden wie hier beschrieben	typeorm SQLite config

Die Collector App benötigt eine Datenbank, um Nutzer und Einstellungen zu speichern. Hierfür können verschiedene SQL Datenbanken-Management-Systeme konfiguriert werden. Standardmäßig verwendet die Collector App eine SQLite Datenbank in Form einer Datei. Es wird also kein Datenbank Server benötigt.

Es gibt verschiedene Möglichkeiten die Collector App zu konfigurieren die im Folgenden erläutert werden.

3.3.2 Sichere Web-Kommunikation (TLS)

Der Web/API Server der Collector App läuft unverschlüsselt (http). Es lässt sich auch keine Verschlüsselung (https) konfigurieren. Das hat den Hintergrund, da sonst der Konfigurationsaufwand, vor allem durch das Zertifikatsmanagement, deutlich komplizierter wäre. In einem vertrauenswürdigen, internen Netzwerk kann der unverschlüsselte Betrieb unter Umständen toleriert werden.

Warning

Sobald die Collector App in einem Netzwerk betrieben wird, auf das auch nicht vertrauenswürdige Personen Zugriff haben könnten oder gar im Internet, ist eine verschlüsselt sehr wichtig, da sonst übertragene Passwörter mitgelesen werden können!

Um die Collector App via TLS (https) zu betreiben, wird der Web-Server nginx empfohlen. Dieser kann als so genannter Reverse Proxy fungieren. Dabei konfiguriert man die Collector App so, dass sie nur unter localhost oder einem sicheren VPN läuft. Nginx wird auf dem selben Server oder auf einem im VPN installiert und verbindet sich mit der Collector App über das unverschlüsselt http Protokoll. Nach außen hin zu dem unsicheren Netzwerk stellt nginx einen sicheren https Server bereit. Dadurch ist nginx in der Lage die Anfragen die über das sichere Protokoll kommen an die Collector weiterzuleiten.

Beispiel Nginx Konfiguration als Reverse Proxy für die Collector App:

```
events {}
http {

  upstream collector_app {
      server {http_collector_url};
  }

  map $http_upgrade $connection_upgrade {
          default upgrade;
          ''      close;
  }

# Anfragen an http werden auf https umgeleitet
  server {
      listen 0.0.0.0:80;
      server_name {server_name};
      server_tokens off;
      return 301 https://$host$request_uri;
  }

# Der sichere https server
  server {
      listen 443 ssl;
      ssl_certificate {path_to_cert};
      ssl_certificate_key {path_to_key};

      ssl_protocols TLSv1.2 TLSv1.1 TLSv1;

      server_name {server_name};
      access_log /var/log/nginx/myapp.log;
      error_log /var/log/nginx/myapp_error.log;

      location /collector-app {
         proxy_set_header X-Real-IP $remote_addr;
         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
         proxy_set_header Host $http_host;
         proxy_set_header X-NginX-Proxy true;
         proxy_set_header X-Ssl on;

         proxy_pass http://collector_app/collector-app;
      }

      location /api/ {
         proxy_set_header X-Real-IP $remote_addr;
         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
         proxy_set_header Host $http_host;
         proxy_set_header X-NginX-Proxy true;
         proxy_set_header X-Ssl on;
```

```
        proxy_pass http://collector_app/api/;
      }

      location /api/subscriptions {
         proxy_set_header X-Real-IP $remote_addr;
         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
         proxy_set_header Host $host;
         proxy_http_version 1.1;
         proxy_set_header Upgrade $http_upgrade;
         proxy_set_header Connection $connection_upgrade;
         proxy_connect_timeout 7d;
         proxy_send_timeout 7d;
         proxy_read_timeout 7d;


         proxy_pass http://collector_app/api/subscriptions;
      }

      location / {
          return 301 /collector_app/;
      }
  }
}
```

Der Wert für die Collector App Url {http_collector_url} und den nginx host name {server_name} müssen entsprechend konfiguriert werden. Außerdem benötigt man ein signiertes Zertifikat {path_to_cert} und Schlüssel {path_to_key}. Wird die Collector App übers Internet betrieben, kann der Zertifikatsdienst Let's Encrypt verwendet werden.

Zu Testzwecken kann auch ein selbst signiertes Zertifikat erstellt werden:

```
openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365
```

3.3.3 JSON Datei

Die Default-Konfigurationsdatei befindet sich

- unter Windows im App Data Verzeichnis: `%appdata%/iTE-SI/Collector-App/config/default.json`
- und unter Docker/Linux `/etc/collector-app/default.json`

Defaultmäßig hat die Datei folgenden Inhalt:

```
{
  "host": "0.0.0.0",
  "port": 4000,
  "tokenSecret": "this-is-a-secret-value-with-at-least-32-characters",
  "encryptionKey": "e44c966f21b9e1577802464f8924e6a37e3e9751fa01304213b2f845d8841d61",
  "corsOrigin": "http://localhost:4000",
  "apiUrl": "http://localhost:4000/api/graphql",
  "wsUrl": "ws://localhost:4000/api/subscriptions",
  "database": {
    "type": "sqlite",
    "database": "/var/lib/collector-app/db.sqlite3"
  }
}
```

Im Abschnitt `database` lassen sich unterschiedlich SQL Datenbanken konfigurieren. Hierfür wird auf die Doku von typeorm verwiesen: https://github.com/typeorm/typeorm/blob/0.2.45/docs/connection-options.md

Standardmäßig wird eine sqlite3 Datenbankdatei verwendet, damit die Collector App, ohne andere Abhängigkeiten, eigenständig betrieben werden kann. Für die Performance der App ist diese Datenbank in der Regel vollkommen ausreichend.

Custom Konfiguration

Möchte man die Standard-Einstellungen anpassen, dann sollte man dies nicht nicht der `default.json` Datei tun, sondern man erstellt eine Kopie `production.json` im selben Verzeichnis. Diese Datei kann nun nach belieben angepasst werden.

Info

Es wird empfolen, nach der Installation die beiden Secrets `tokenSecret` und `encryptionKey` zu ändern.

3.3.4 Environment Variablen

Für das Deployment via Docker Container aber auch zum setzen der Secret-Keys eigenen sich vor allem Environment Variablen. Ist eine Variable gesetzt wird sie gegenüber dem Wert in der Konfigurationsdatei bevorzugt verwendet.

Allgemeine Environment Variablen

- COLLECTOR_APP_HOST
- COLLECTOR_APP_PORT
- COLLECTOR_APP_TOKEN_SECRET
- COLLECTOR_APP_ENCRYPTION_KEY
- COLLECTOR_APP_CORS_ORIGIN
- COLLECTOR_APP_PUBLIC_GRAPHQL_URL
- COLLECTOR_APP_PUBLIC_WS_URL

Datenbank Environment Variablen

Zur Konfiguration der Datenbank werden die Typeorm Environment Variablen gesetzt wie hier beschrieben.

- TYPEORM_CACHE
- TYPEORM_CACHE_ALWAYS_ENABLED
- TYPEORM_CACHE_DURATION
- TYPEORM_CACHE_OPTIONS
- TYPEORM_CONNECTION
- TYPEORM_DATABASE
- TYPEORM_DEBUG
- TYPEORM_DRIVER_EXTRA
- TYPEORM_HOST
- TYPEORM_LOGGER
- TYPEORM_LOGGING
- TYPEORM_MAX_QUERY_EXECUTION_TIME
- TYPEORM_PASSWORD
- TYPEORM_PORT
- TYPEORM_SCHEMA
- TYPEORM_SID
- TYPEORM_SUBSCRIBERS
- TYPEORM_SUBSCRIBERS_DIR
- TYPEORM_SYNCHRONIZE
- TYPEORM_URL
- TYPEORM_USERNAME
- TYPEORM_UUID_EXTENSION

3.4 Getting Started

Nachdem das Basis-Setup gemäß Basis Getting Started installiert wurde, kann die Collector App eingerichtet und der Collector konfiguriert werden.

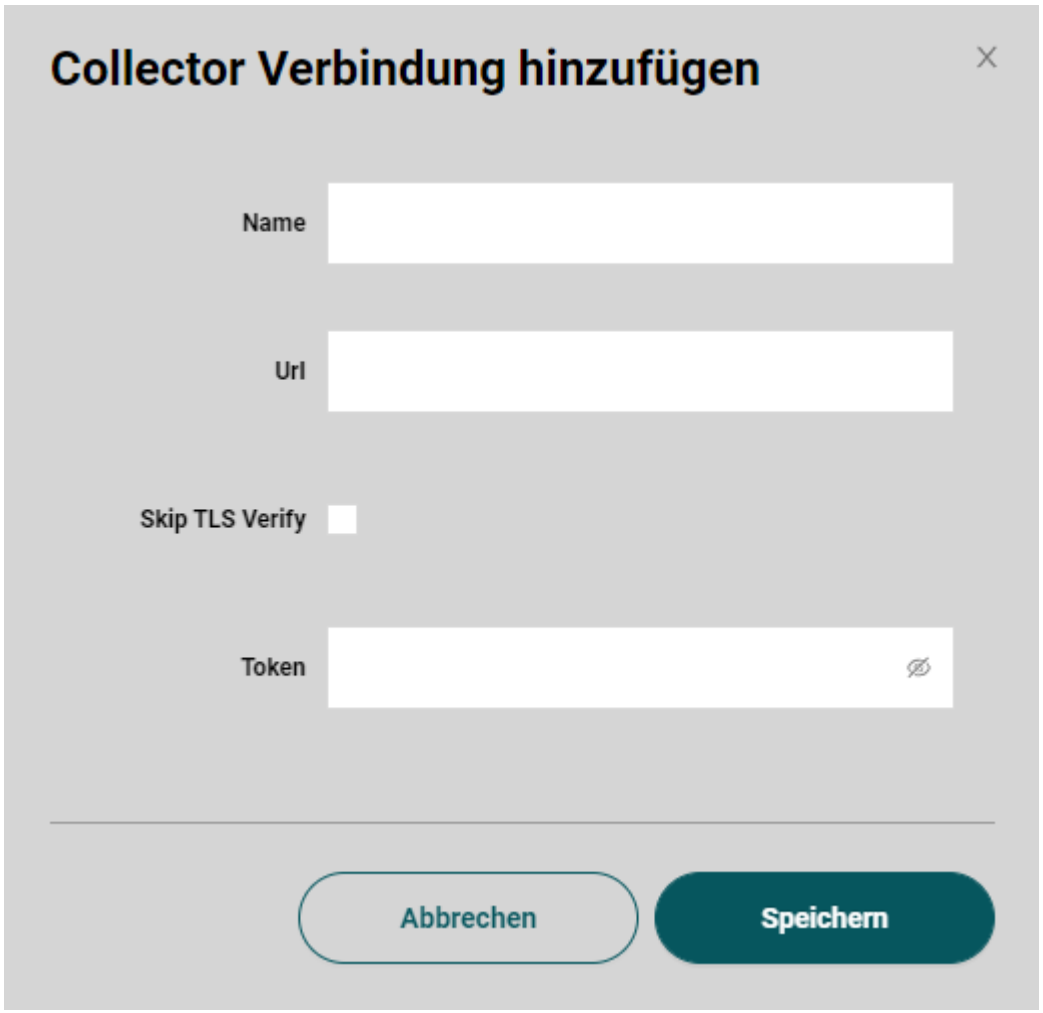
3.4.1 Collector(n) einbinden

Damit die Collector App Data Collectoren konfigurieren kann braucht Sie deren Netzwerk-Endpunkt.

1. Über die Navigationsleiste zur Data Collectors Seite: http://localhost:4000/collector-app/de/collectors wechseln.

2. Auf den `Collector Hinzufügen` Button klicken.

3. Es kann ein beliebiger Name vergeben werden. Die Url beginnt mit `http://` für eine unverschlüsselte Verbindung und mit `https://` für eine mit TLS verschlüsselte, gefolgt von der IP-Adresse/Hostname und dem Port (default: `http://localhost:8001`). Falls eine verschlüsselte Verbindung mit selbst signierten Zertifikaten verwendet wird, muss die TLS Verifizierung ausgehschalten werden. Die Konfigurations-Schnittstelle des Data Collectors kann mit einem Token abgesichert werten. Standardmäßig ist kein Token konfiguriert.



Name	Url
Collector	**Docker:** http://collector:8001 **Windows:** http://localhost:8001

4. Auf Speichern klicken. Der Collector erscheint in der Collector Übersichtstabelle.

Collector Einstellungen

1. Durch klicken auf das Zahnrad-Symbol in der Aktions-Spalte der Übersichtstabelle einer Collector Spalte, öffnen sich die Collector-Einstellungen.

2. Die Puffer Einstellung erlaubt es Daten auf dem Computer auf dem der Collector läuft lokal zwischen zu speichern, falls es zu einer Netzwerkunterbrechung kommt. Angegeben werden muss ein Dateipfad und die maximale Puffergröße. Der Lokale Fehlerspeicher speichert Informationen, wenn Daten nicht korrekt übertragen werden konnten.

3. Durch die Metriken Einstellung schreibt der Collector zyklische Informationen über seinen eigenen Zustand in eine Influx Datenbank.

4. Falls der Lizenz-Server nicht automatisch über die Broadcast Suche gefunden wird, kann dessen IP oder Hostname konfiguriert werden. Außerdem wird hier angegeben in welchem Lizenzmodus der Collector betrieben werden soll.

3.4.2 OPC UA Browser einbinden

Um OPC UA Server durchsuchen zu können wird ein OPC UA Browser benötigt. Die Collector App braucht auch hier den Netzwerk-Endpunkt. Die Konfiguration erfolgt equivalent zum Collector auf der OPC UA Browser Seite.

Name	Url
OPC UA Browser	**Docker:** http://opcua-browser:8080 **Windows:** http://localhost:8080

3.4.3 Verbindungen anlegen

Als erstes müssen die Verbindungsinformationen zu den Daten-Quellen und -Senken konfiguriert werden. Das kann auf der Verbindungs-Seite getan werden.

OPC UA Server Verbindung

Es wird die OPC UA Server Verbindung als Daten-Quellen angelegt.

1. Klicken auf den `Verbindung erstellen` Button. Es öffnet sich ein Dialog mit 3 Schritten.

2. Eine Verbindung kann gleichzeitig auf mehreren Collectoren angelegt werden. Zum Beispiel wenn sie alle in eine zentrale Datenbank schreiben sollen. Es werden also jene Collectoren im ersten Schritt ausgewählt.

3. Im zweiten Schritt werden allgemeine Verbindungsinformationen abgefragt. Die Ausfallzeitabschaltung gibt die Zeit an, die bei einem Verbindungsabbruch gewartet wird, bis ein erneuter Wiederaufbauversuch erfolgt.

Für die Schnellstartkonfiguration werden folgende Werte konfiguriert:

Name	Url
Sample OPC UA Sample Server	**Docker:** opc.tcp://opcua-server:4840 **Windows:** opc.tcp://localhost:48010

Die Import-Funktion erlaubt es zuvor exportierte Verbindungen in Form einer json-Datei zu laden.

4. Zuletzt muss noch der OPC UA Verbindungstyp ausgewählt werden. Dann können noch OPC UA spezifische Sicherheitseinstellungen vorgenommen werden. Im default Fall werden keine Sicherheitsmechanismen angewandt.

5. Nach dem klicken auf Speichern erscheint die Verbindung in der Verbindungsübersichtstabelle.

Influx Datenbank Verbindung

Die Verbindung zur Influx Datenbank dient als Daten-Senke.

1. Der Erstellen Dialog wird ein zweites mal durchlaufen. Dieses mal mit den folgenden Werten für den Demo Aufbau:

Name	Url	Datenbank
InfluxDB	**Docker:** http://influxdb:8086 **Windows:** http://localhost:8086	demo-db

Im 3. Schritt wird die Influx Verbindung ausgewählt und der Datenbankname konfiguriert. Existiert die Datenbank noch nicht, wird diese automatisch angelegt. Die restlichen Parameter werden default mäßig gesetzt.

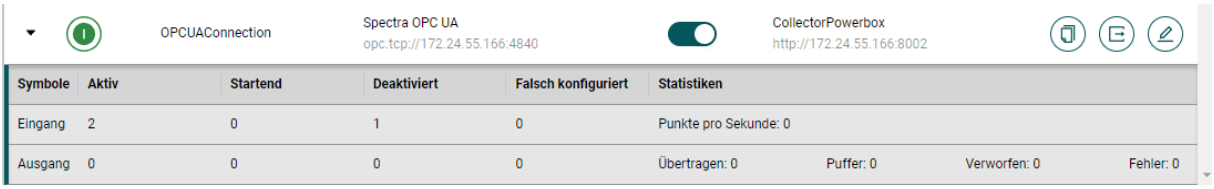
2. Nach dem Speichern erscheint auch die Influx Datenbank Verbindung in der Tabelle.

3.4.4 Symbole anlegen

Jede Verbindung hat untergeordnete Symbole. Dabei gibt es Ausgangs- und Eingangs-Symbole, welche untereinander verlinkt werden. Ein Eingangssymbol kann dabei mit mehreren Ausgangssymbolen verbunden werden. In den Folgenden Schritten wir erklärt, wie ein OPC UA Subscription Symbol angelegt und mit einem Influx Measurement verlinkt wird.

OPC UA Subscription anlegen

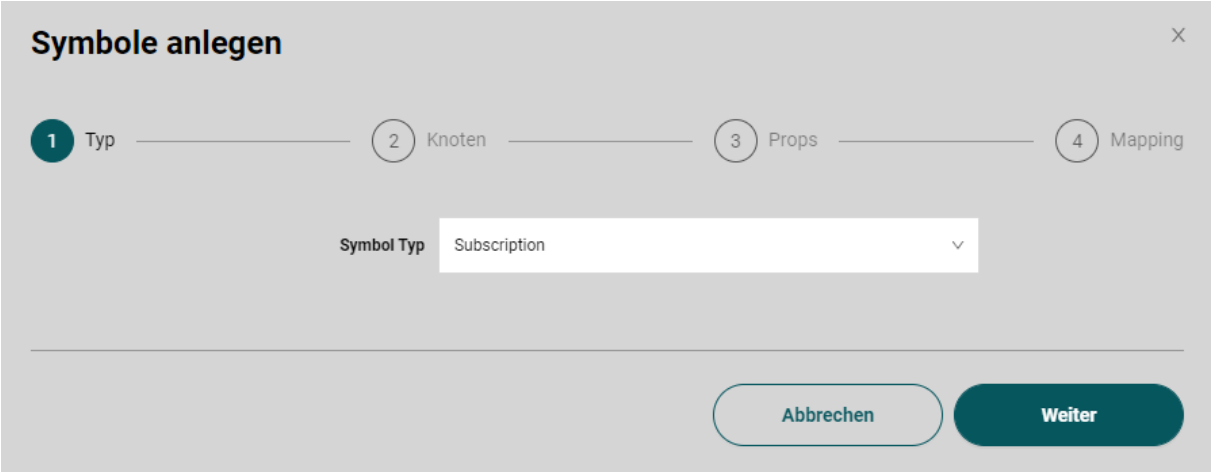
1. Grundsätzlich kann man auf zwei verschiedene Weg zur Symbolübersicht gelangen: Über die Navigationsleiste oder wenn man in der Verbindungstabelle die OPC UA Verbindung anklickt, wird die Zeile expandiert. Es erscheint eine Zusammenfassung der Eingangs- und Ausgangs-Symbolen. Durch klicken auf die Eingangs-Zeile wird man automatisch auf die Symbol Seite weitergeleitet, so dass die Verbindung direkt vor gefiltert wird.



2. Auf der Symbol Seite muss im übergeordneten Filter die OPC UA Sample Server Verbindung ausgewählt werden. Um aus allen Verbindungen auswählen zu können wird auf das Lupe Symbol geklickt.

3. Nachdem eine Verbindung ausgewählt wurde erscheint der `Symbol erstellen` Button auf dem man klickt.

4. Es öffnet sich ein Dialog mit mehreren Schritten. Als Typ wird der `Subscription` gewählt.



5. Im nächsten Schritt kann der OPC UA Server mit Hilfe des Browsers durchsucht werden. Durch klicken auf die Checkbox können OPC UA Variablen ausgewählt werden. Es muss mindestens eine Variable ausgewählt werden. Für jede markierte Variable wird ein Symbol erstellt.

6. Danach können optional verschiedene Eigenschaften konfiguriert werden.

7. Im letzten Schritt `Automapping` können verschiedene Verbindungen ausgewählt werden, für die jeweils Ausgangssymbole angelegt und mit den OPC UA Subscription Symbolen automatisch verknüpft werden. Hier wird die InfluxDB ausgewählt.

8. Nach dem Klicken auf Speichern, werden OPC UA Subscription - und Influx Measurement Symbole angelegt und automatisch verknüpft. Falls die Symbole eingeschalten sind beginnt der Data Collector mit der Arbeit. Die Symbole erscheinen in der Tabelle. Durch klicken auf eine Symbol-Zeile werden die Verbundenen Symbole angezeigt.

Influx Measurement anpassen

1. Wechseln des Übergeordneten Filters auf die InfluxDB Verbindung.

2. Expandieren eines Influx Measurements in der Tabelle.

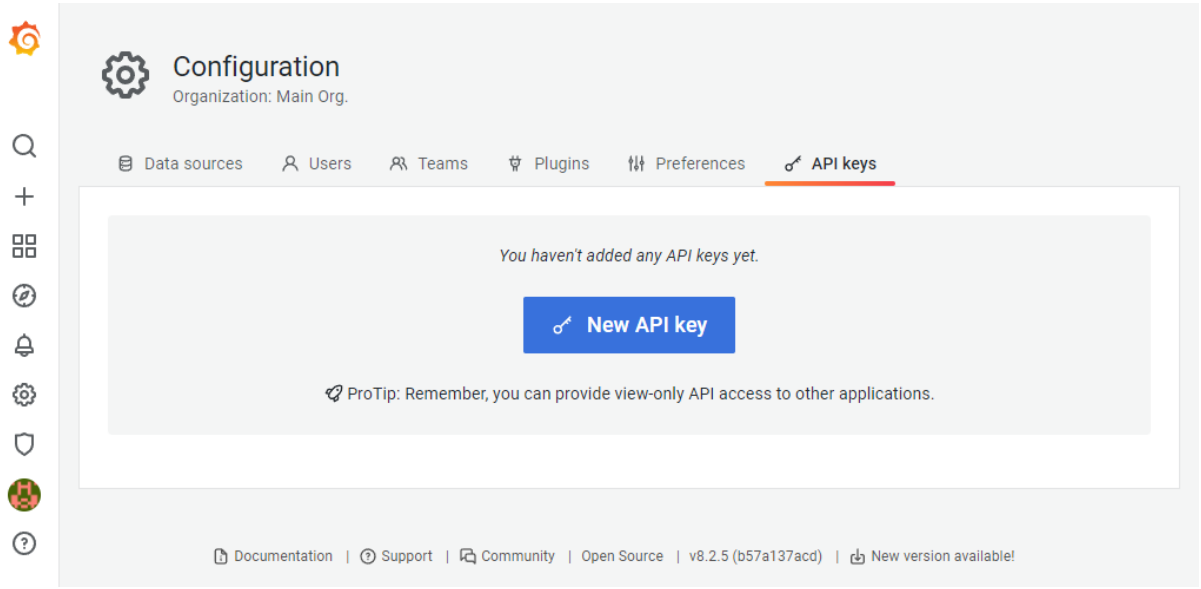
3. Im Expandierten Bereich wird die Struktur des Measurements dargestellt. Hier können Fields und Tags konfiguriert werden (Influx Doku). Standardmäßig wird der Variablen-Pfad im OPC UA Tree als Tag und der Wert des Verknüpften Symbols als Value Field angelegt. Hier kann die Struktur noch weiter angepasst werden.

3.4.5 Grafana Integration zu Visualisierung

Grafana ist ein web-basiertes Tool um vor allem Zeitreihendaten zu visualisieren. In der Collector App kann eine Anbindung ans Grafana erflogen, damit die vom Collector erfassten Daten direkt Angezeigt werden können.

API Key erstellen

In den Grafana Einstellungen kann ein API Key angelegt werden. Diese benötigt die Admin Rolle. Weitere Infos in der Grafana doku.



API Key eingragen

1. Zunächst wird über die Navigationsleiste die Einstellungsseite ausgewählt.

2. Im `Baustein Explore` wird die Grafana integration konfiguriert. Hier wird die URL und der zuvor erstellte API Key eingetragen. Nach klicken auf Speichern wird gefragt, ob Grafana Datasources automatisch auf basis der Verbindungen in der Collector App erstellt werden sollen.

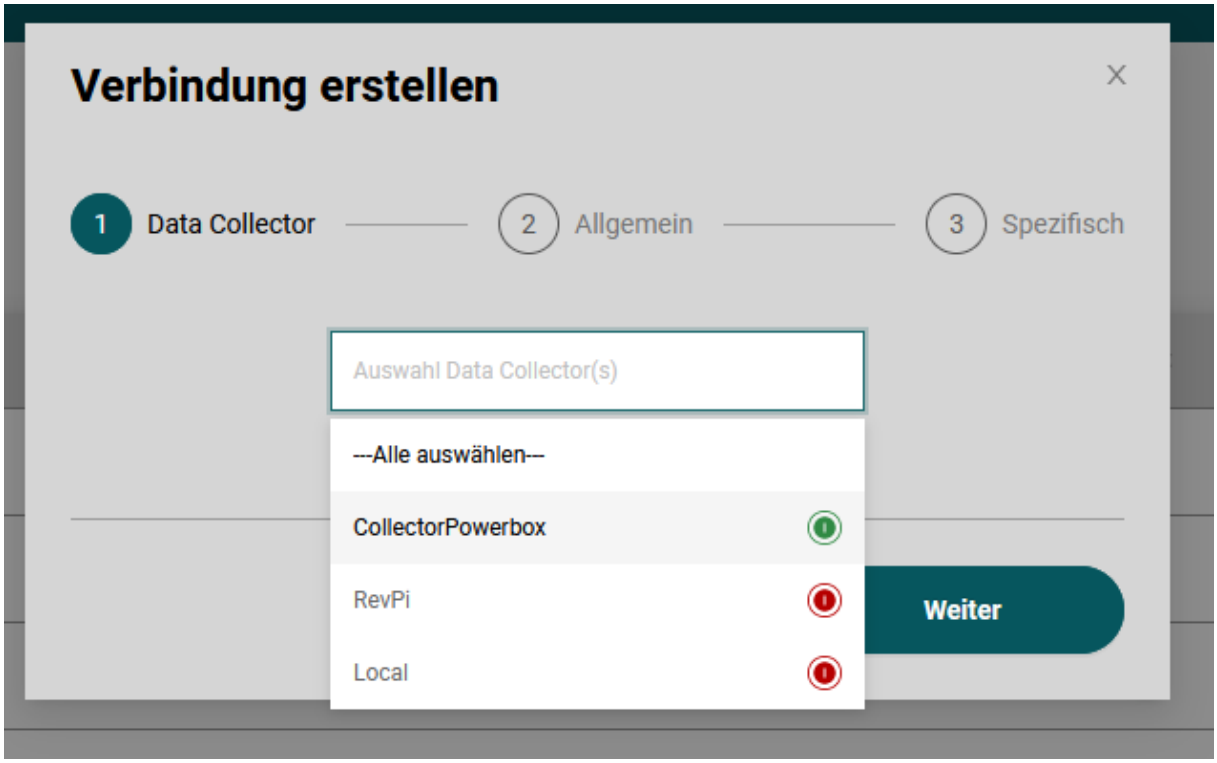
3. Nun kann man in der Symboltabelle bei Influx Measurements auf den Link Button klicken. Dann öffnet sich ein neuer Browser Tab mit einem Grafana Panel in dem das Measurement angezeigt wird.

3.5 Konfiguration von Verbindungen

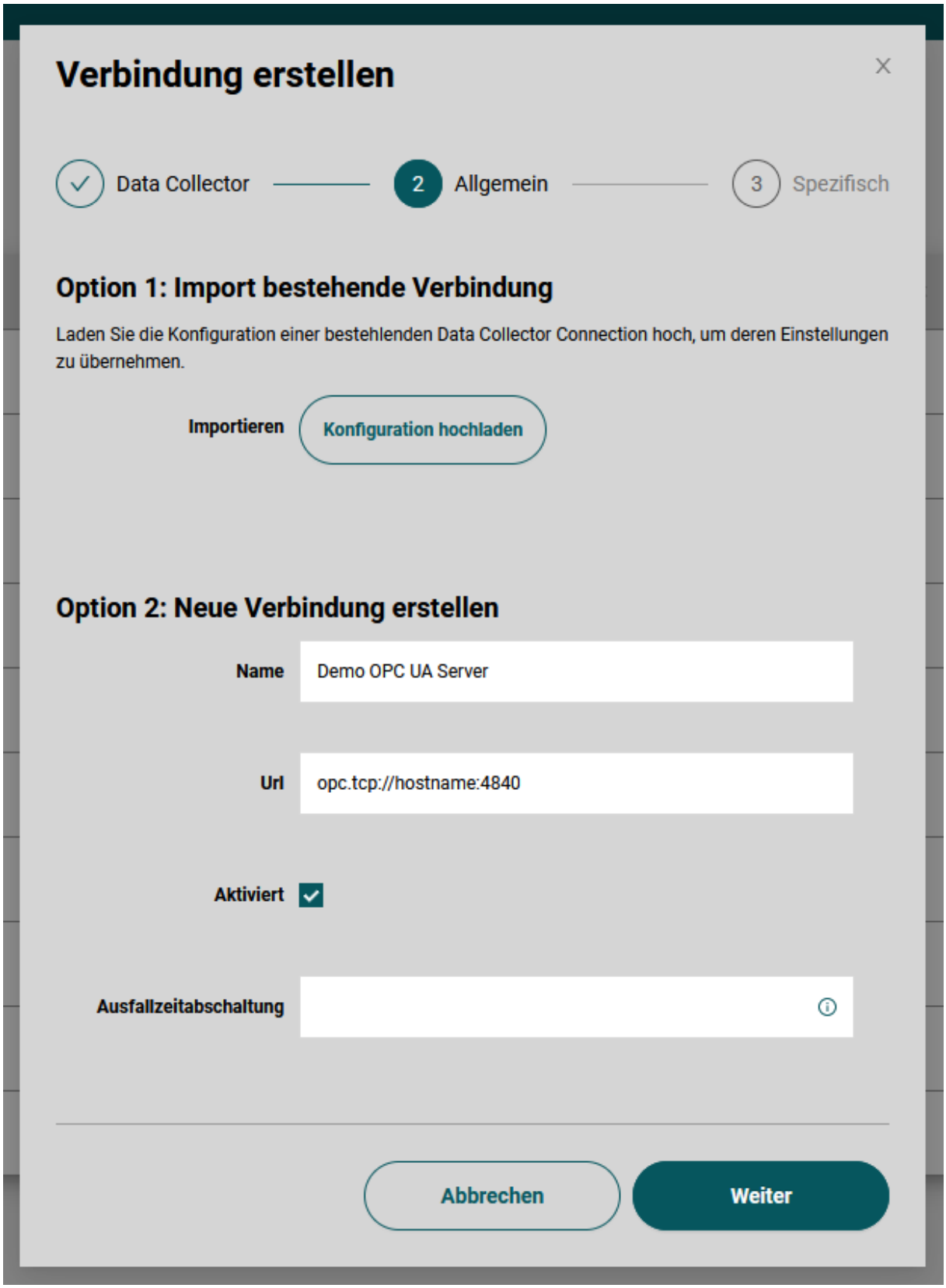
Verbindungen können über den "Verbindungen Seite", welche in der Navigationsleiste in der linken Seite ausgewählt werden kann, angelegt werden. Über den "Verbindung erstellen" Button öffnet sich der entsprechende Wizard-Dialog.

3.5.1 Verbindung erstellen über Wizard-Dialog

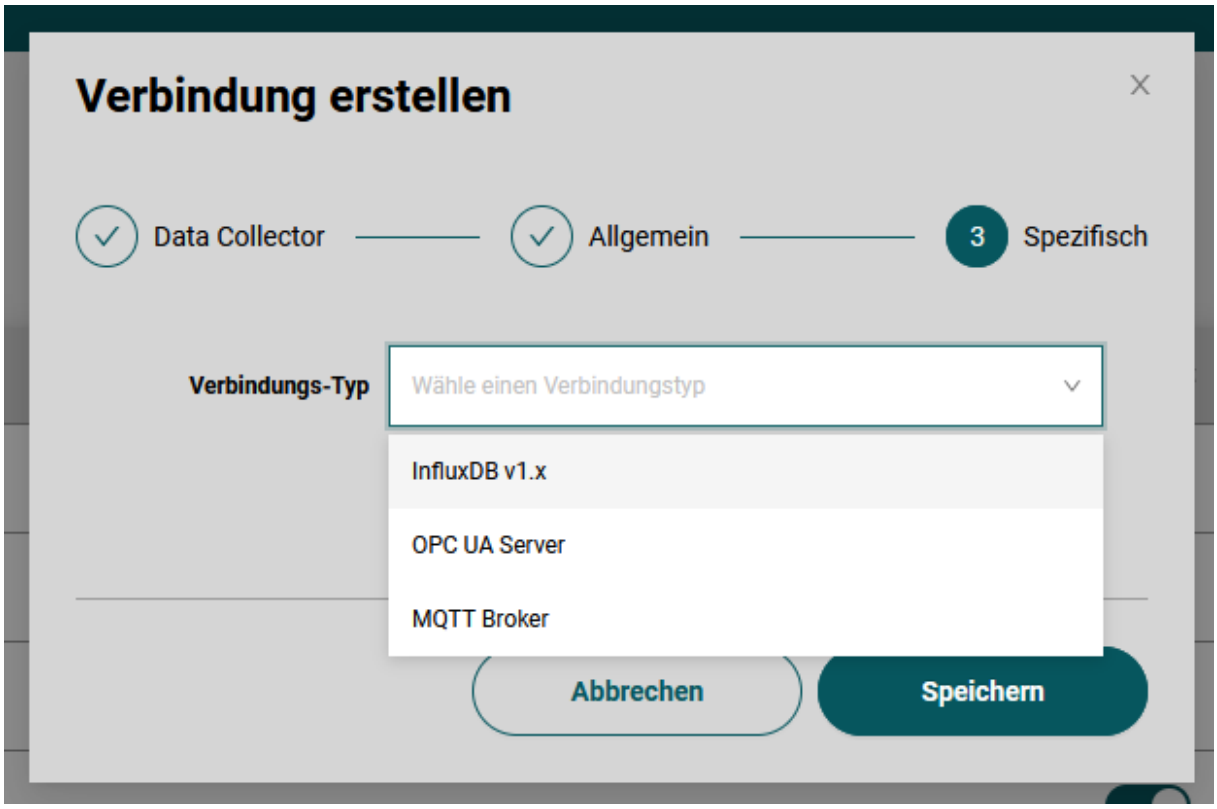
Im ersten Schritt muss der Data Collector ausgewählt werden, für den die Verbindung erstellt werden soll.



Auf der zweiten Seite muss der **Name** und die **Url** der Verbindung, z.B. opc.tcp://hostname oder mqtt://hostname, eingegeben werden.

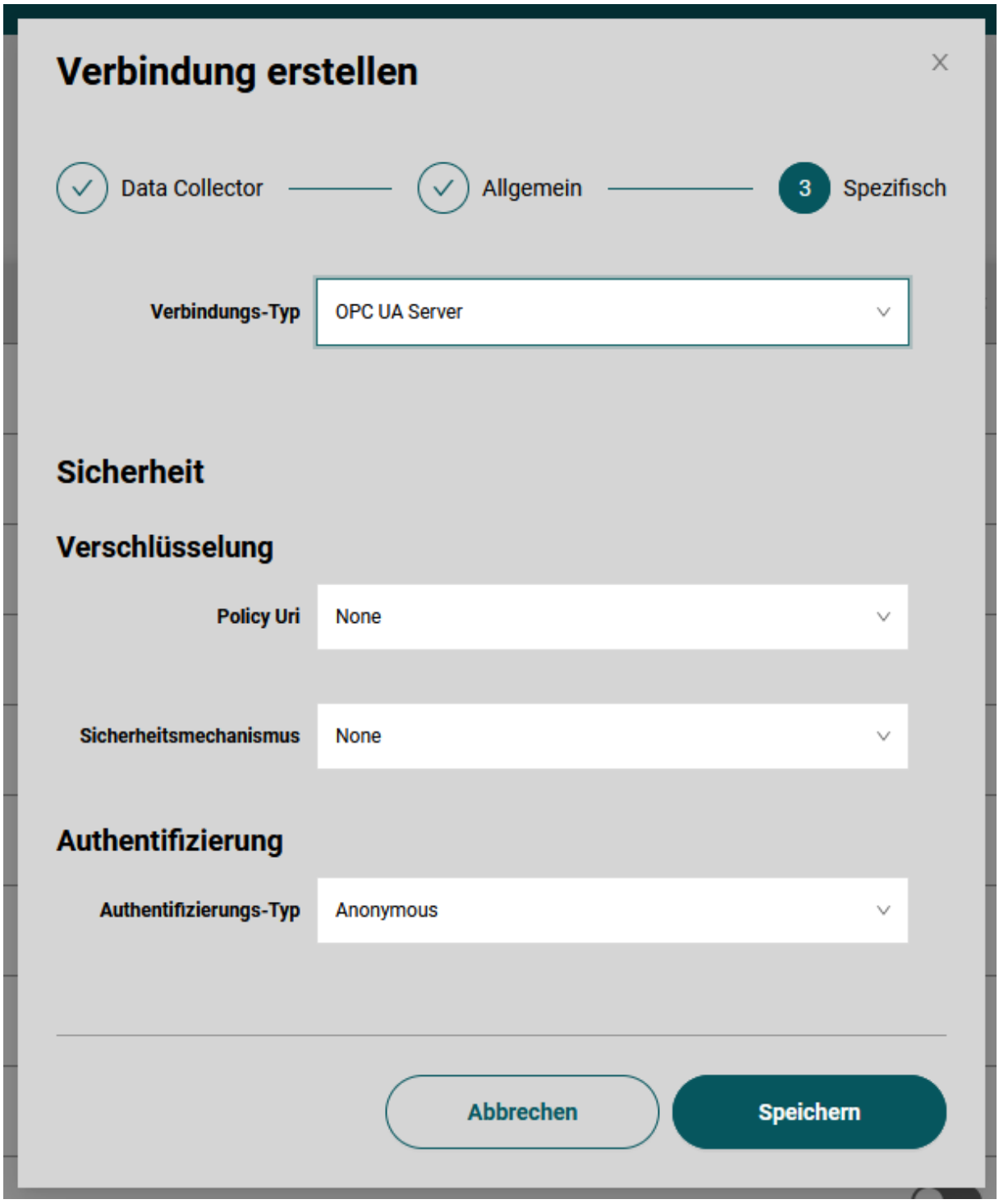


Auf der dritten Seite wird der Verbindungstyp ausgewählt und die entsprechenden verbindungsspezifischen Einstellungen angezeigt.



OPC UA Verbindung

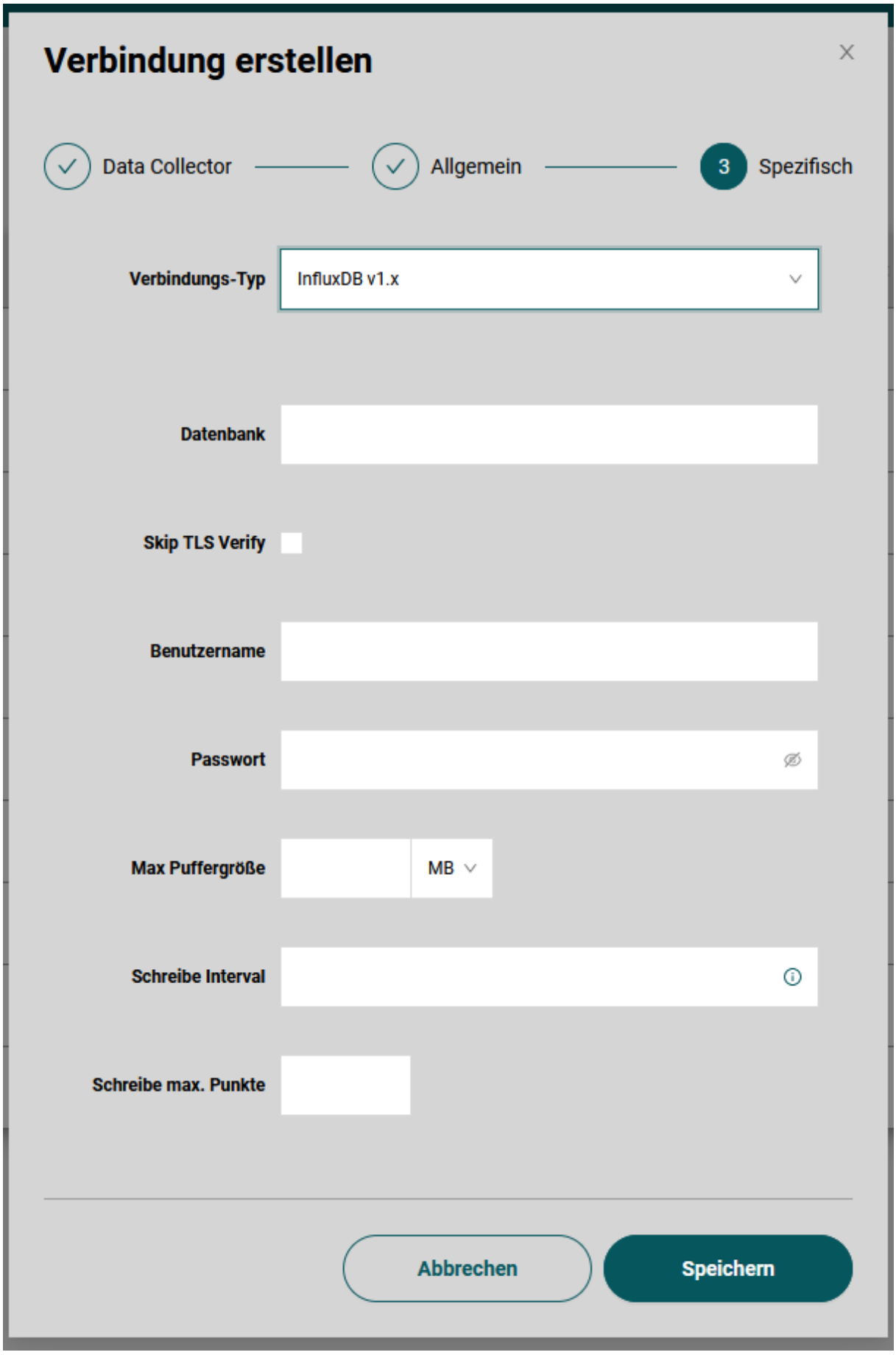
Die OPC UA spezifischen Einstellungen werden verwendet um die Sicherheit der Verbindung einzustellen.



Mit den Default-Einstellungen verbindet sich der Data Collector unverschlüsselt und ohne Benutzername-Passwort.

Influx Verbindung

Für die Influx Verbindung ist es notwendig den Datenbank-Namen zu konfigurieren. Jede Influx-Datenbank benötigt eine eigene Verbindung im Data Collector.

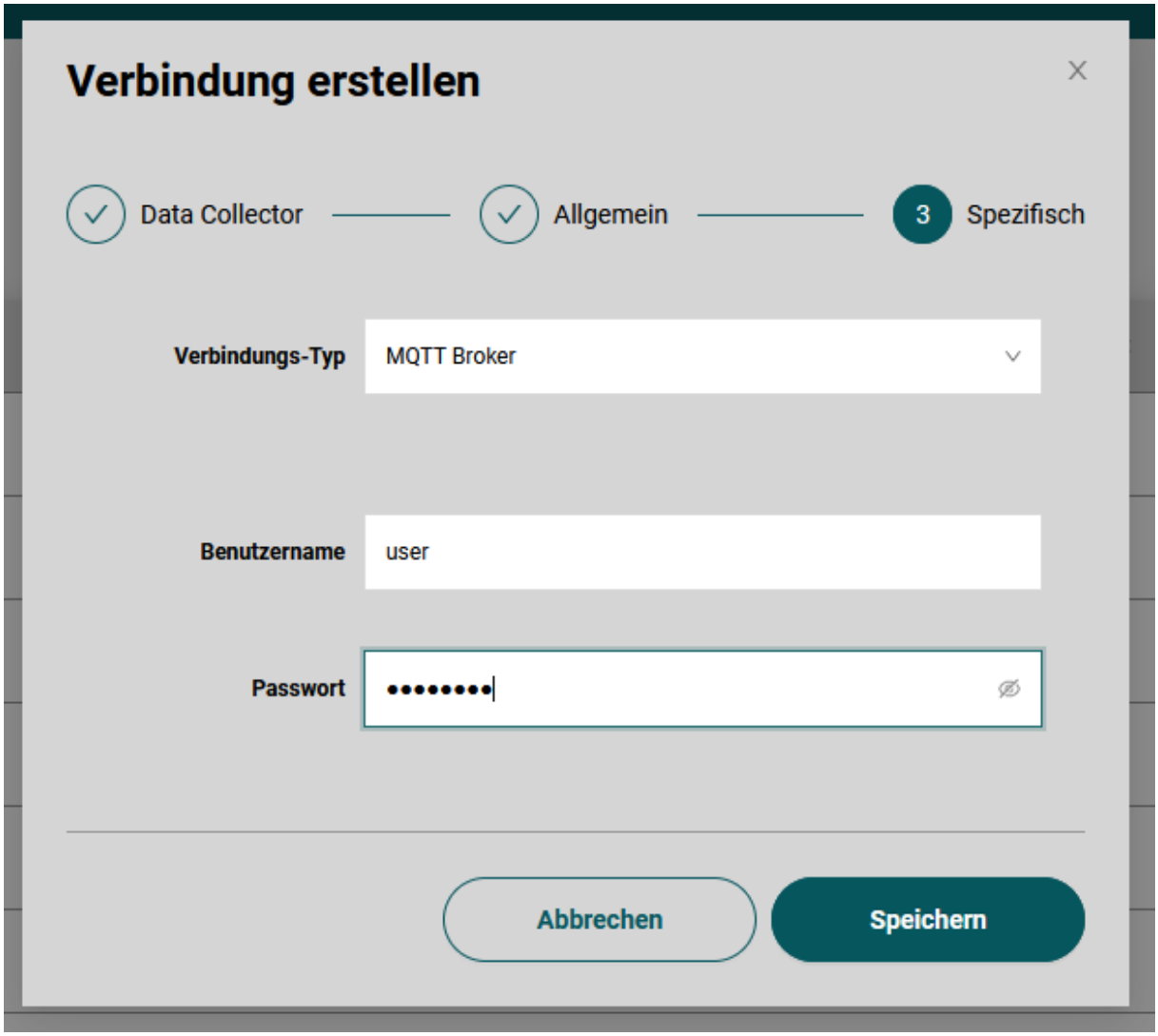


Die restlichen Einstellungen können unausgefüllt bleiben, so dass sie mit passenden Default-Werten ausgefüllt werden.

- **Skip TLS Verify** sollte ausgewählt werden, falls die Influx Datenbank ein selbst-signiertest Zertifikat und einen HTTPs-Endpunkt verwendet.
- **Benutzername** und **Passwort** sollten ausgefüllt werden wenn die Influx Datenbank eine Autorisierung erfordert.
- **Max Puffergröße**, **Schreibe Interval** und **Schreibe max. Punkte** können verwendet werden um den Data Collector intern zu optimieren. In der Regel sollten diese Felder freigelassen werden und nur bei Bedarf angepasst werden.

MQTT Verbindung

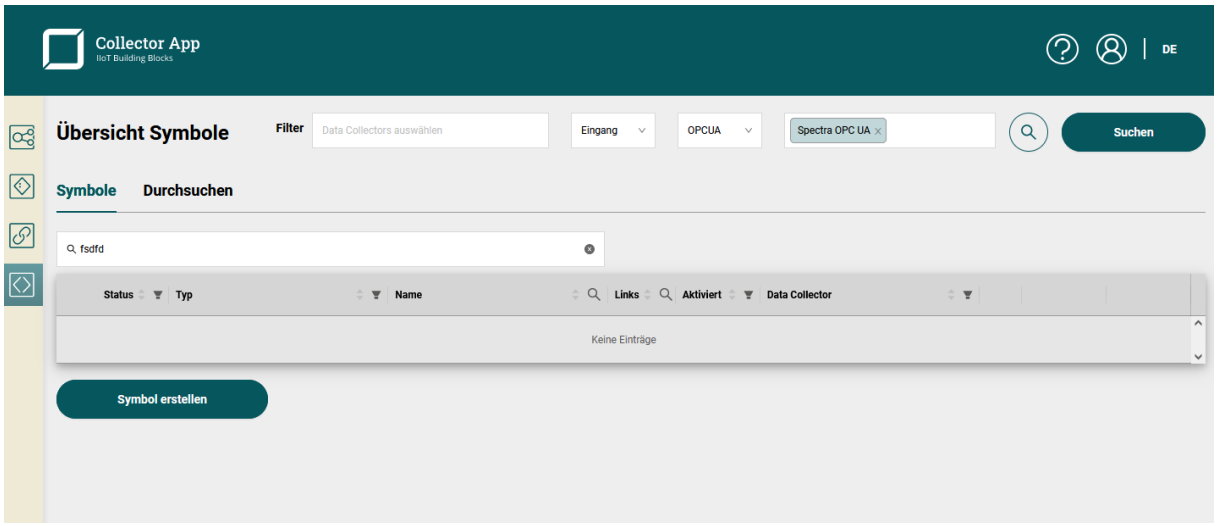
Für die MQTT-Verbindung kann optional der Benutzername und das Passwort angegeben werden, falls der MQTT Broker Autorisierung implementiert.



3.6 Konfiguration von Symbolen

Symbole können über die "Symbole Seite", welche in der Navigationsleiste in der linken Seite ausgewählt werden kann, angelegt werden.

Um Symbole anlegen zu können ist es notwendig, dass eine Verbindung ausgewählt wird. Hier im Beispiel ist die Verbindung "Spectra OPC UA" in der Filter-Leiste ausgewählt.

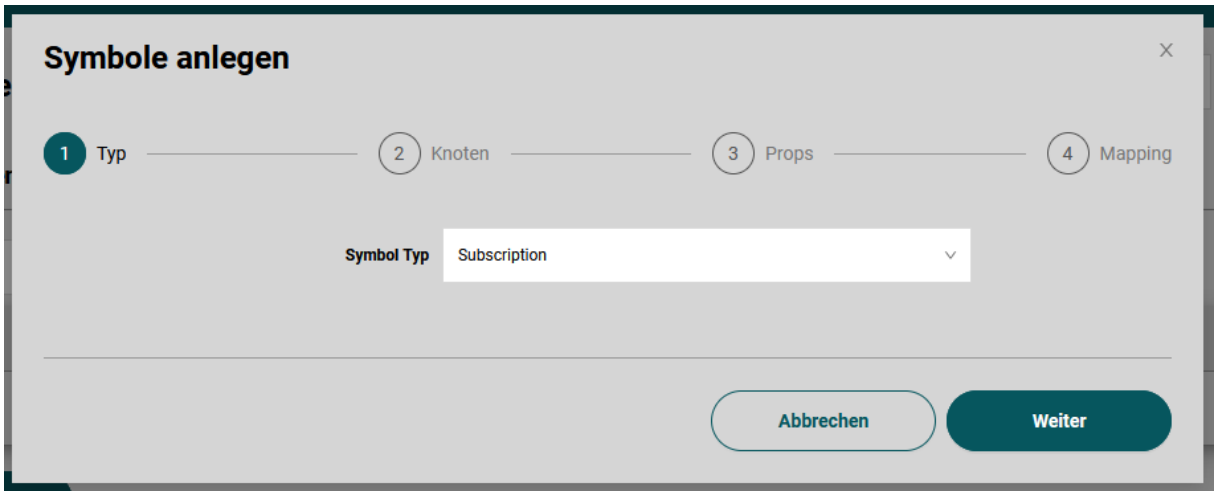


Über den "Symbol erstellen" Button wird der verbindungsspezifische Dialog geöffnet.

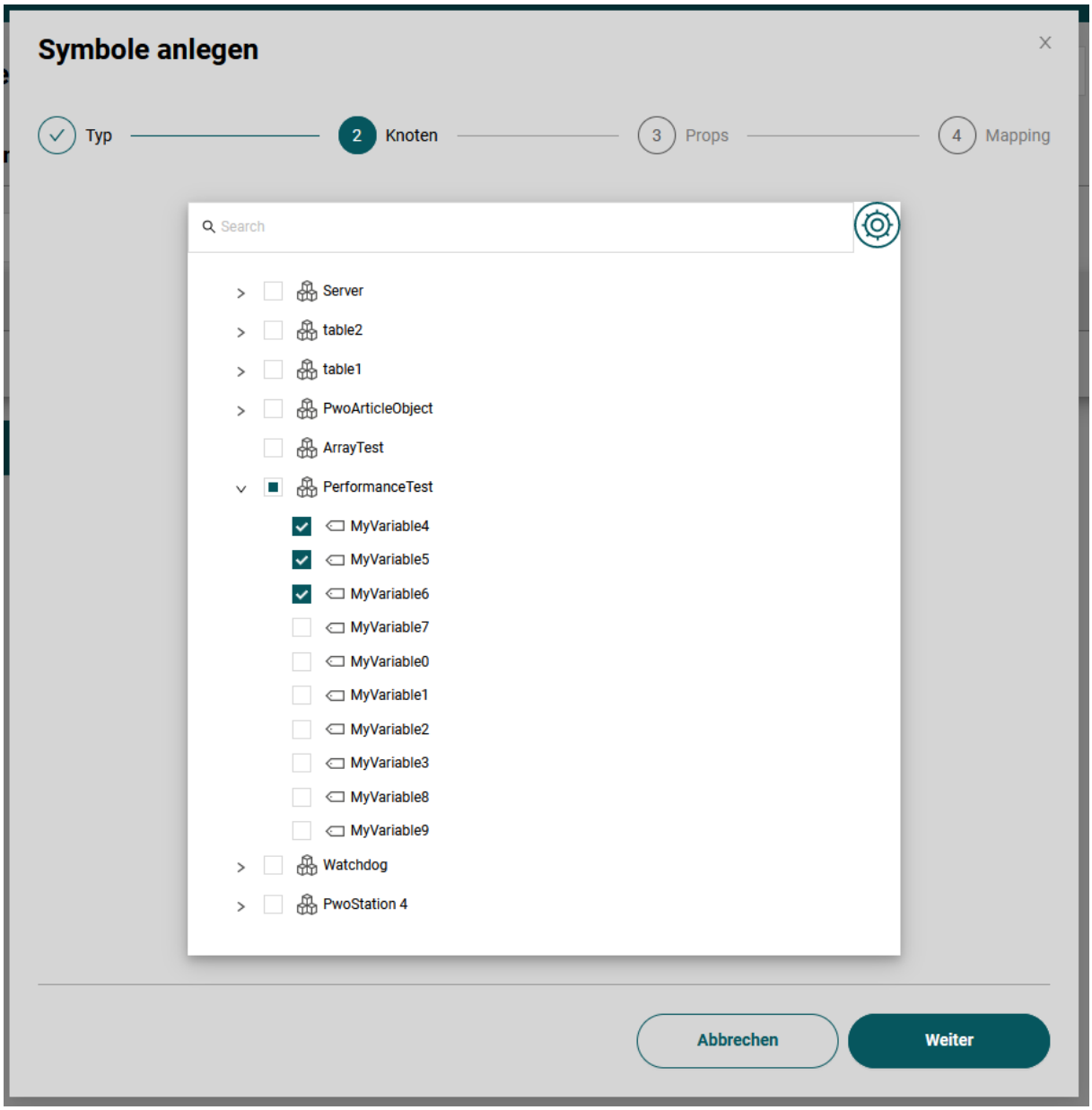
3.6.1 OPC UA Subscription Symbol anlegen

Eine OPC UA Subscription ermöglicht es von einem einzelnen OPC UA Knoten abzutasten (sampling). Hierbei werden OPC UA MonitoredItems verwendet, sodass nur Änderungen übertragen werden. Die Datenpunkte können an ein oder mehrere Output Symbole weitergeleitet werden, welches es dann z.B. in eine Influx Datenbank schreibt.

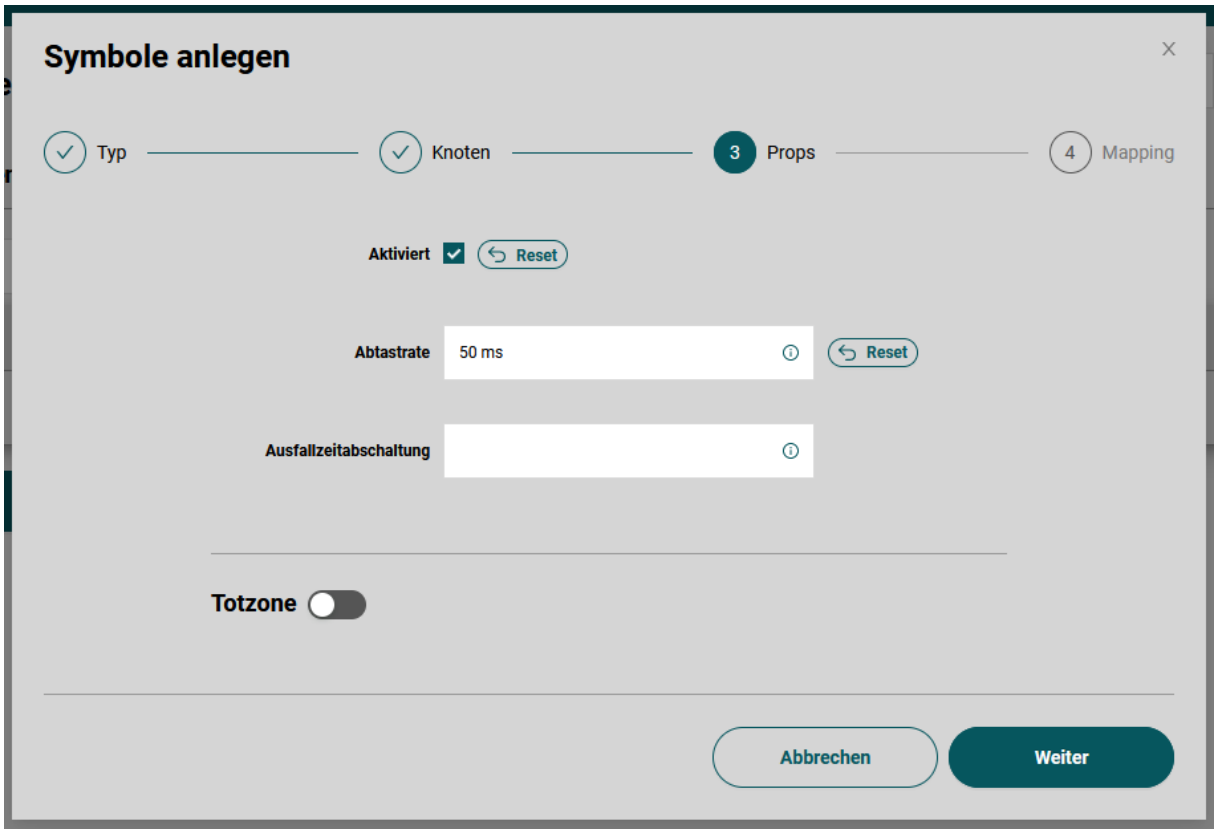
Auf der ersten Seite muss der Typ *Subscription* ausgewählt werden.



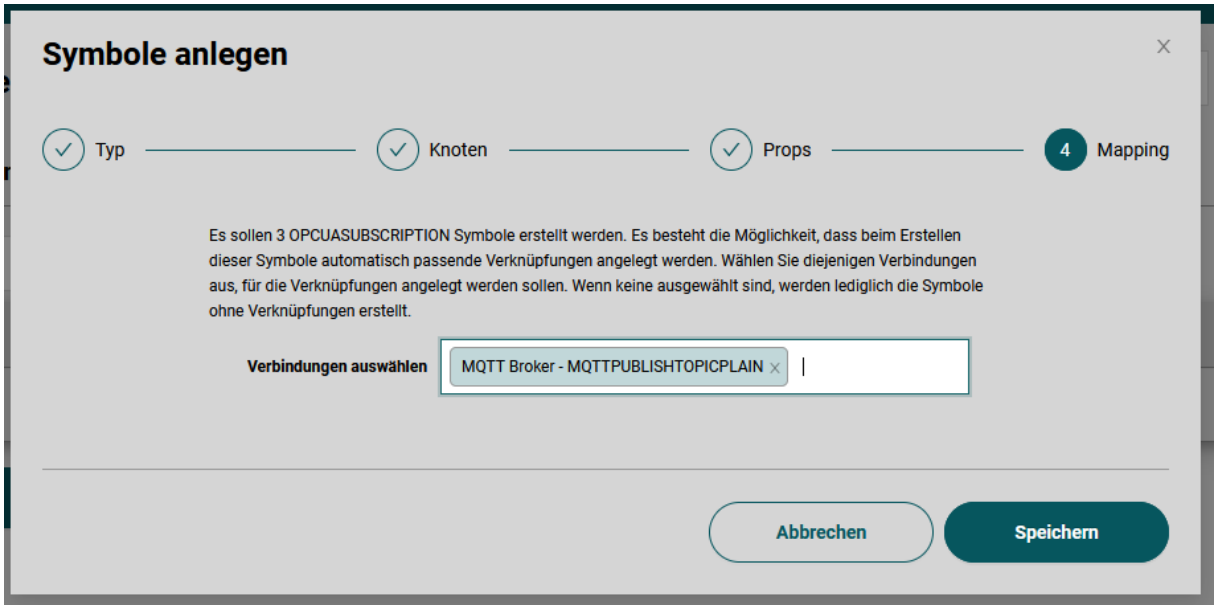
Auf der zweiten Seite wird der OPC UA Adressraum angezeigt. Hierzu muss ein OPC UA Browser konfiguriert sein. Einen oder mehrere Knoten können ausgewählt werden. Für jeden ausgewählten Knoten wird eine OPC UA Subscription erzeugt.



Auf der dritten Seite können spezifische Einstellungen vorgenommen werden.



Auf der vierten Seite kann das Automapping konfiguriert werden. Für jedes OPC UA Subscription Symbol wird in der ausgewählten Verbindung ein Output Symbol angelegt und mit dem Input Symbol verknüpft. Im Beispiel wird MQTTPublishTopicPlain Symbol für die Verbindung 'MQTT Broker' erzeugt..



3.6.2 OPC UA BulkRead Symbol anlegen

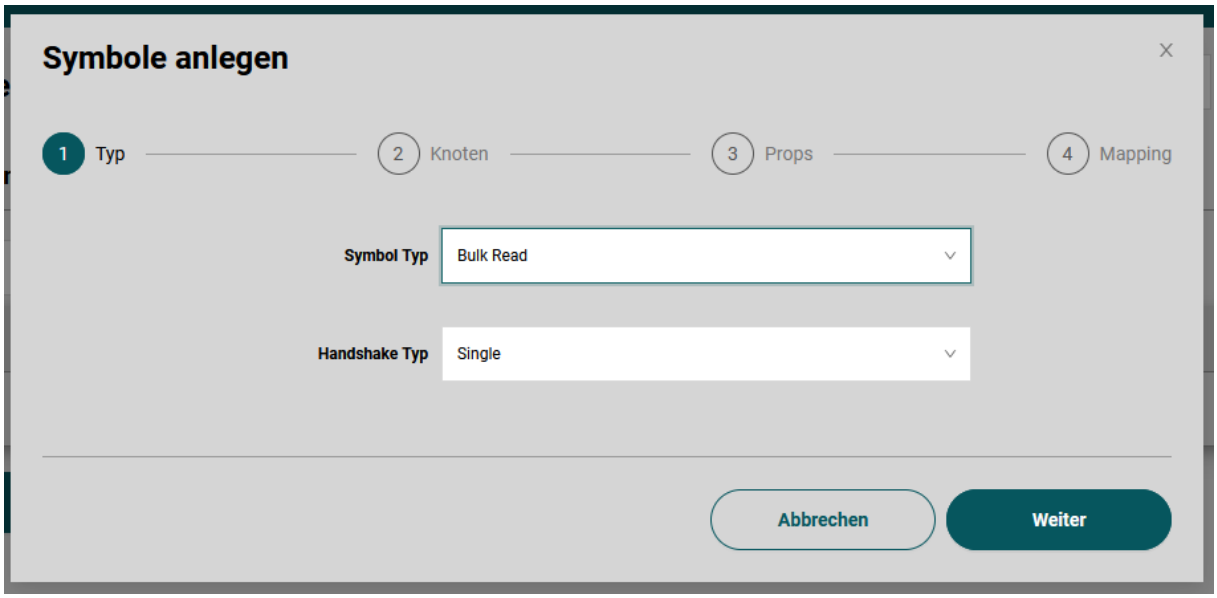
Das OPC UA BulkRead Symbol erlaubt es mehrere OPC UA Knoten gleichzeitig in eine Struktur auszulesen. Dafür kommuniziert der Data Collector mit dem OPC UA Server über einen Handshake. Der OPC UA Server teilt dem Data Collector mit, wann die Daten ausgelesen werden sollen, und der Data Collector benachrichtigt den OPC UA Server, wenn der Datenpunkt abgearbeitet wurde. Dieser Prozess wiederholt sich.

Auf der ersten Seite muss der Typ **BulkRead** ausgewählt werden. Es gibt die Möglichkeit die Art des Handshakes auszuwählen.

Der Handshake-Typ **Single** setzt den Handshake auf das folgende Verhalten. Der OPC UA Server bietet einen OPC UA Knoten (Trigger-Knoten) vom Typ Boolean an, den der Data Collector ausliest. Soll der Datenpunkt ausgelesen werden, setzt der OPC UA Server den Wert auf *true*. Die konfigurierten Knoten werden in eine Struktur ausgelesen und weitergeleitet. Um den Handshake zu beenden, setzt der Data Collector den Trigger-Knoten auf *false* zurück.

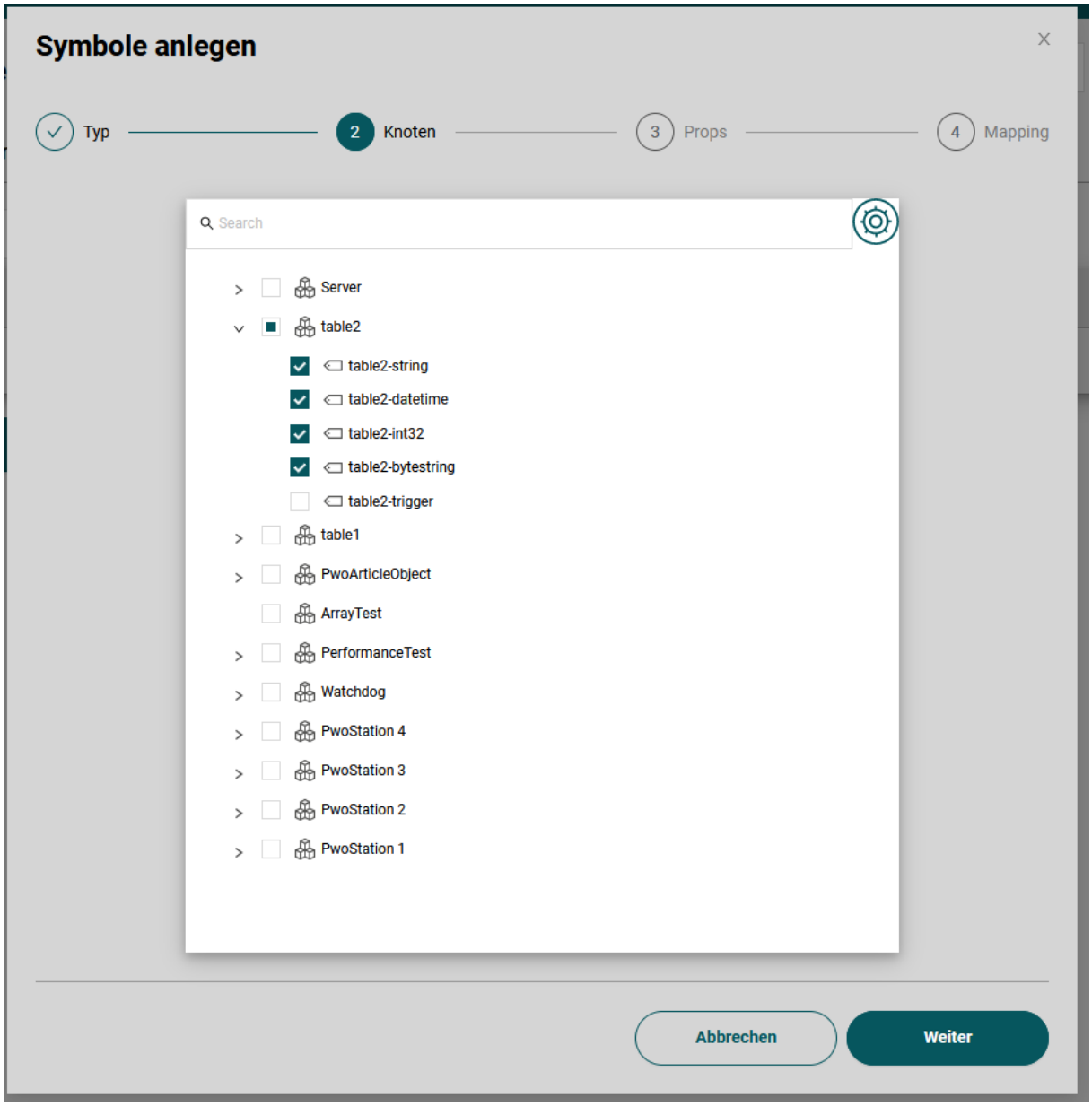
Der Handshake-Typ **Input/Output** verwendet zwei OPC UA Knoten vom Typ Boolean. Der OPC UA Server setzt den Input Knoten auf *true*, wenn der Datenpunkt ausgelesen werden soll. Hat der Collector den Datenpunkt verarbeitet setzt er den Output Knoten auf *true*. Anschließend setzt der OPC UA Server den Input Knoten auf *false*. Um den Handshake zu beendet setzt der Data Collector den Output Knoten wieder auf *false*.

Beide Handshake-Typen lassen sich über den **Confirm After** Type einstellen. Mit **Read** beendet der Data Collector den Handshake nach erfolgreichen Lesen. Hier gibt es keine Garantie, dass der Datenpunkt auch wirklich geschrieben wurde. Mit **WriteAny** beendet der Data Collector den Handshake erst, wenn mindestens ein Output Symbol den Punkt erfolgreich geschrieben hat. Mit **WriteAll** müssen alle konfigurierten Output Symbole den Datenpunkt erfolgreich geschrieben haben.

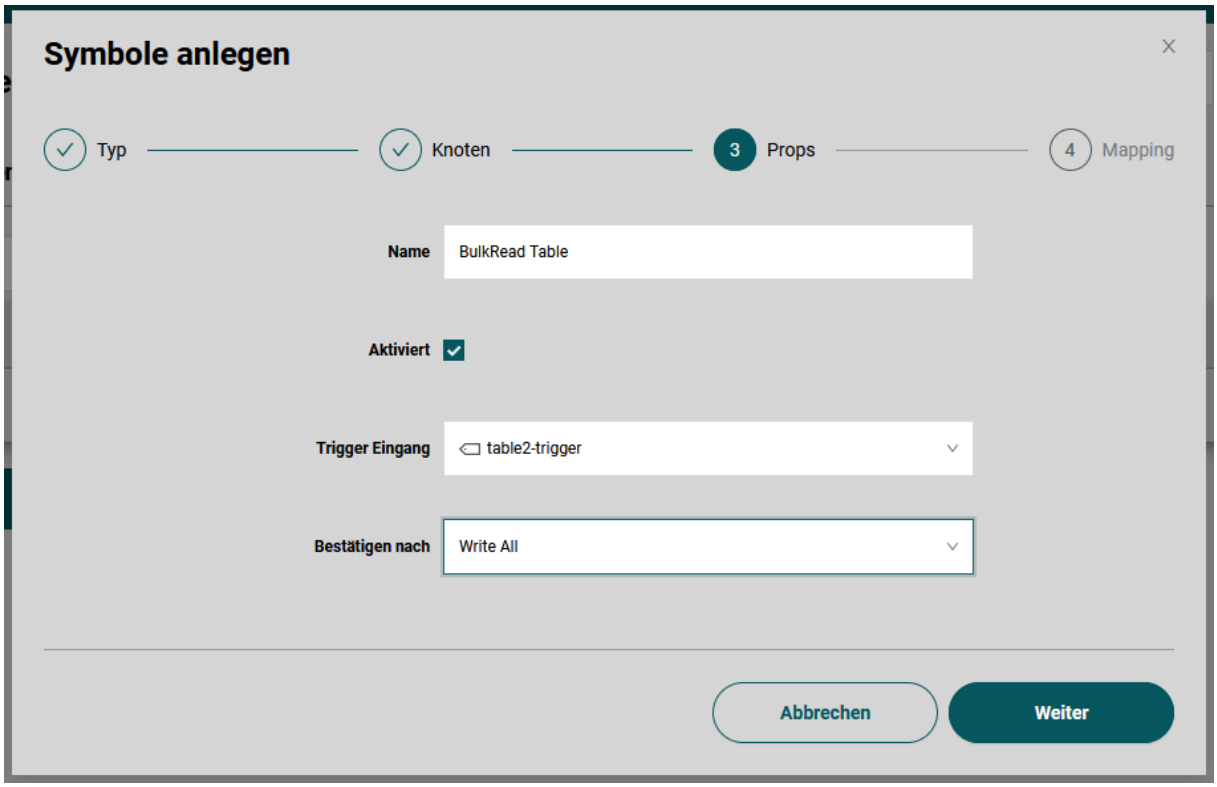


Single Handshake Konfiguration

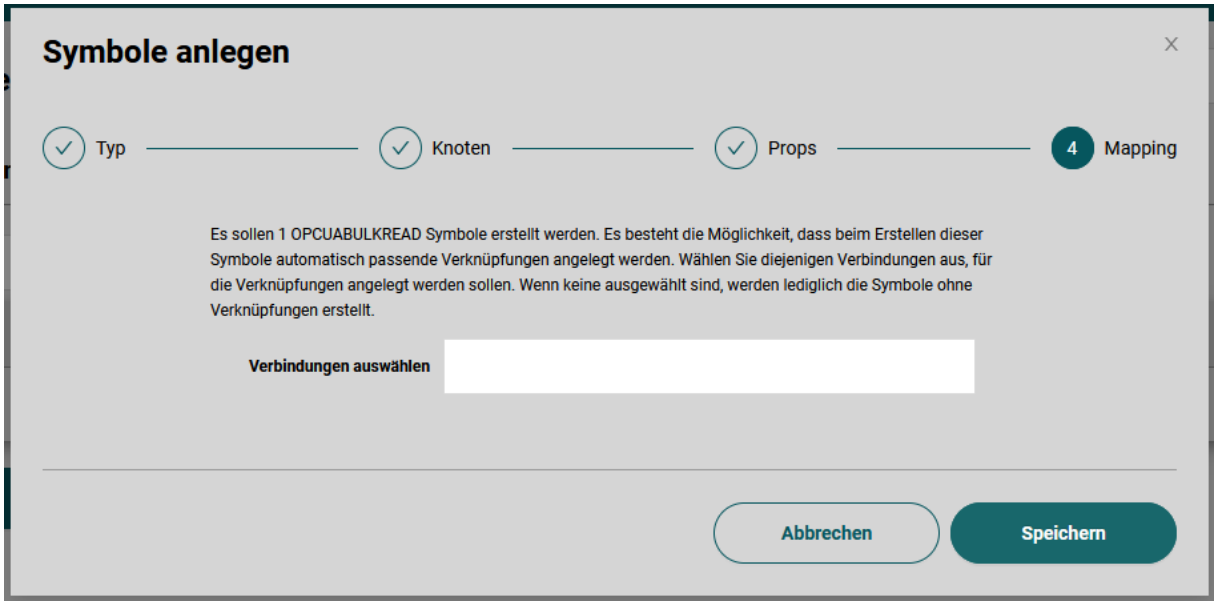
Auf der zweiten Seite wird der OPC UA Adressraum angezeigt. Alle Knoten die für den Datenpunkt ausgelesen werden sollen müssen ausgewählt werden. Der Trigger-Knoten soll **nicht** ausgewählt werden.



Auf der dritten Seite muss der **Trigger-Eingang**-Knoten und der **Bestätigen nach** Typ ausgewählt werden.

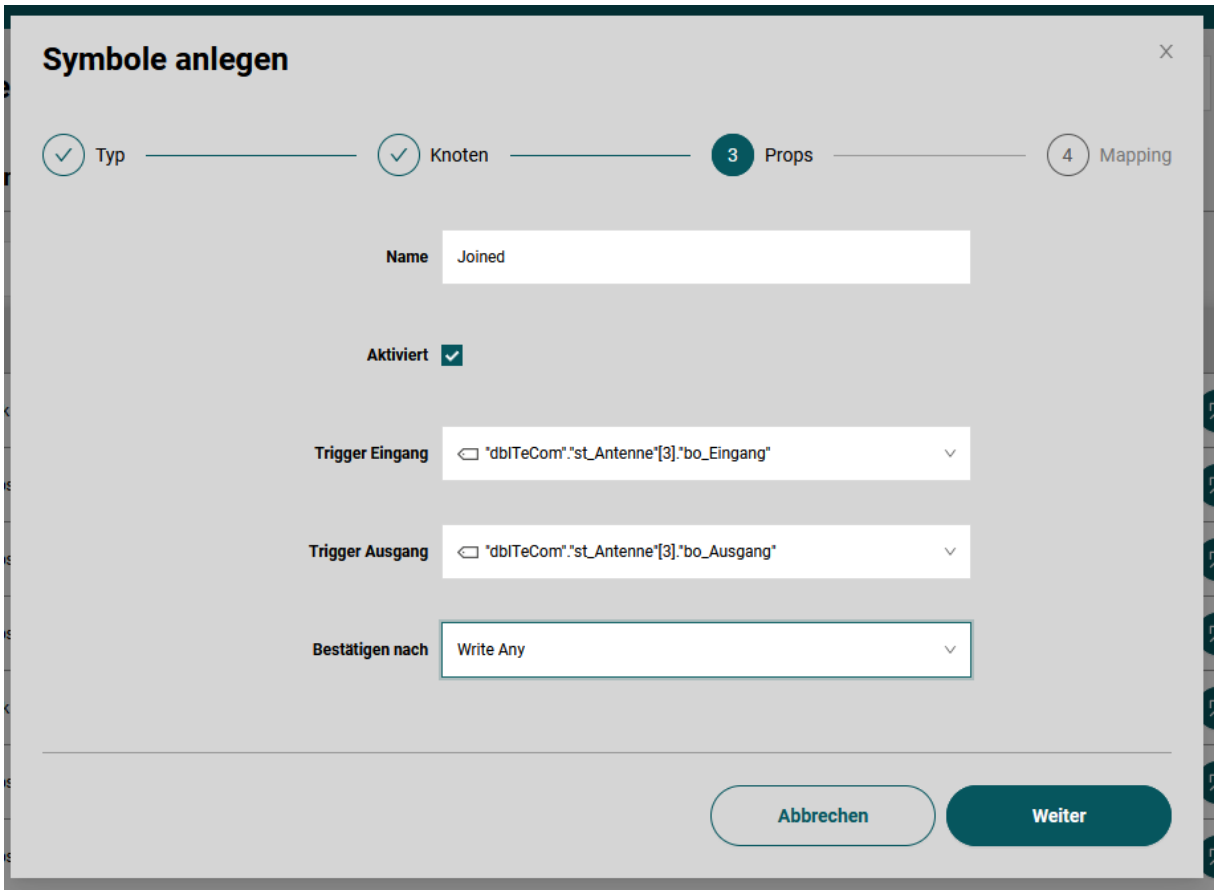


Auf der letzten Seite kann das Automapping konfiguriert werden. Dies ist analog zur Beschreibung für OPC UA Subscriptions.



Input/Output Handshake Konfiguration

Der Input/Output Handshake-Type wird analog zum Single Handshake-Type konfiguriert. Auf der dritte Seite des Dialogs müssen aber die *Trigger Eingang*- und *Trigger Ausgang*-Knoten konfiguriert werden.

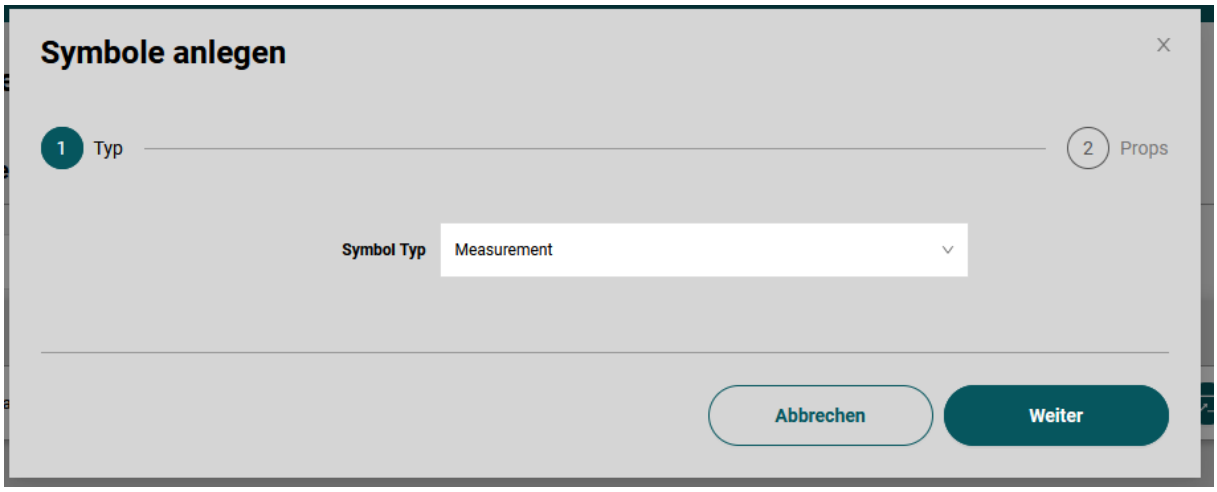


Achtung: Die Trigger-Knoten dürfen nicht auf der zweiten Seite ausgewählt sein, denn sonst können sie nicht auf der dritten Seite ausgewählt werden.

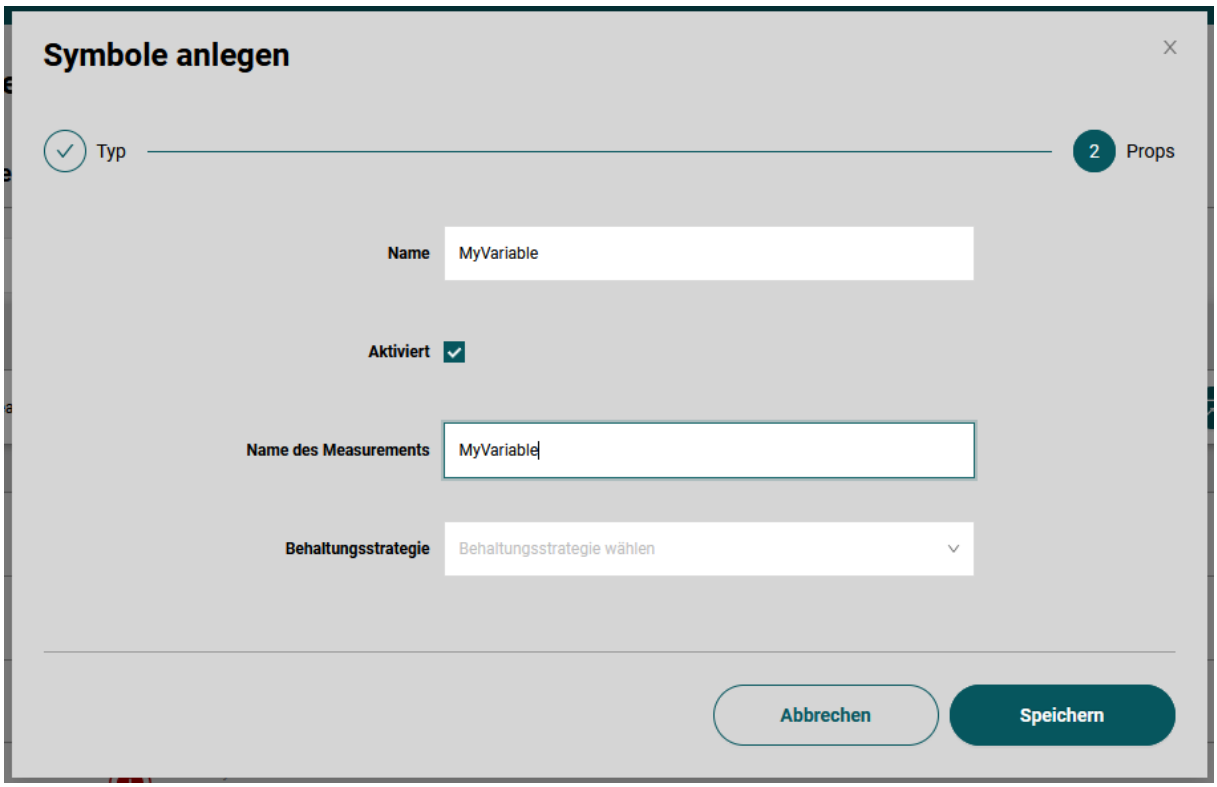
3.6.3 Influx Measurement anlegen

In der Regel werden Influx Measurements über das Automapping aus einen Input Symbol erstellt. Manchmal ist es auch hilfreich ein Measurement manuell anzulegen oder umzukonfigurieren.

Zunächst muss die Verbindung ausgewählt werden, für die das Measurement erzeugt werden soll. Im "Symbol erstellen" Dialog muss der Symbol Typ **Measurement** ausgewählt werden.

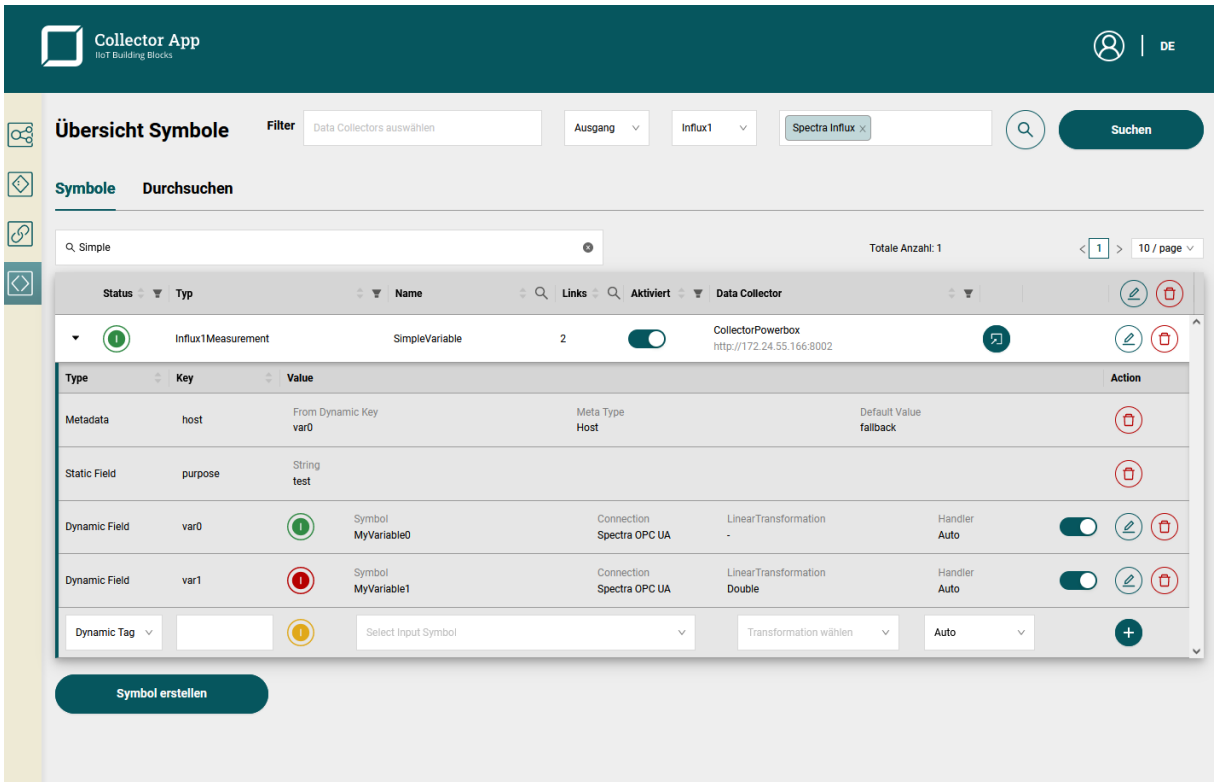


Auf der zweiten Seite können weitere Einstellungen vorgenommen werden.



Der **Name** wird innerhalb der App verwendet. Der **Name des Measurements** entspricht dem Influx Measurement Namen. Der Default-Wert der **Behaltungsstrategie** entspricht der Default Retention Policy der Datenbank.

Abschließend können die Tags und Fields des Measurements in der Symboltabelle konfiguriert werden. Dazu muss das erstellte Measurement per Mouse-Click expandiert werden. Neue Felder können in der letzten Reihe konfiguriert werden.



Es gibt 5 Konfigurationstypen:

1. **Metadata**: verwendet für den Tag die Metadata Informationen des Input Symbols. Der Default Value wird verwendet um im Fall von fehlenden Metadaten einen Wert vorzugeben. kann der Wert nicht ermittelt werden (fehlt und kein Defaultwert), werden die Datenpunkte verworfen. Im Beispiel wird **Host** konfiguriert um den Hostname der Verbindung im Tag 'host' zu annotieren. Metadaten können nur von konfigurierten dynamischen Tags oder Fields verwendet werden.
2. **Static field**: Text der als Feld im Measurement gespeichert wird. *Achtung*: In dieser Version wird immer der Datentyp String verwendet, selbst wenn der Text einer Zahl entspricht.
3. **Static tag**: Text der als Tag im Measurement gespeichert wird.
4. **Dynamic field**: Die Datenpunkte des ausgewählten Input Symbols werden in das Feld gespeichert. Lineare Transformation und fester Datentyp können ausgewählt werden. In der Regel ist es sinnvoll den Datentyp auf *Auto* zu belassen.
5. **Dynamic tag**: Die Datenpunkte des ausgewählten Input Symbols werden in den Tag mit dem konfigurierten Key gespeichert. Der Datentyp des Datenpunkts muss einer Ganzzahl, einem Boolean oder einer einem String entsprechen. Andere Datentypen werden nicht unterstützt. **WARNUNG**: Dynamische Tags können die Performance der Datenbank verschlechter. Für jeden unterschiedlichen Tag-Wert wird innerhalb der Influx Datenbank eine neue Zeitreihe angelegt. Dynamische Tags können hilfreich sein, wenn es wenig Änderungen im Wertebereich gibt, da man über die InfluxQL *WHERE* Queries auf Tags aber nicht auf Fields durchführen kann.

Achtung werden mehr als ein dynamisches Tag oder Field verwendet, dann gelten die Beschränkungen der Join-Funktionalität.

Array in Influx Measurements

Besteht ein Datenpunkt aus einem Array, dann wird jedes Array Element als Datenpunkt in die Influx Datenbank geschrieben. Dazu annotiert der Data Collector jeden Punkt mit einem Tag 'index', welcher dem Index des Werts im Array entspricht. Ist ein statisches oder dynamisches Tag mit dem Key 'index' konfiguriert, wird dieses einfach überschrieben.**Achtung** für jeden Index wird innerhalb der Datenbank eine eigene Zeitreihe erzeugt.

Datenstrukturen in Influx Measurements

Ist der Datenpunkt eine Struktur, z.B. über OPC UA BulkRead oder dem Datentyp einer OPC UA Subscription erzeugt, dann kann das Datum ebenfalls in die Datenbank geschrieben werden. Dazu wird die Struktur flach in einen Datenpunkt konvertiert. Datenstrukturen können nur als Field aber nicht als Tag gespeichert werden.

Beispielstruktur im JSON-Format:

```
{
  "anInt": 3,
  "anBool": false,
  "nested": {
    "aString": "world",
  }
}
```

Die entsprechende Zeile im Influx LineProtocol mit einem dynamischen Feld mit Key 'data' zum Zeitstempel 132909231230223 sieht wie folgt aus:

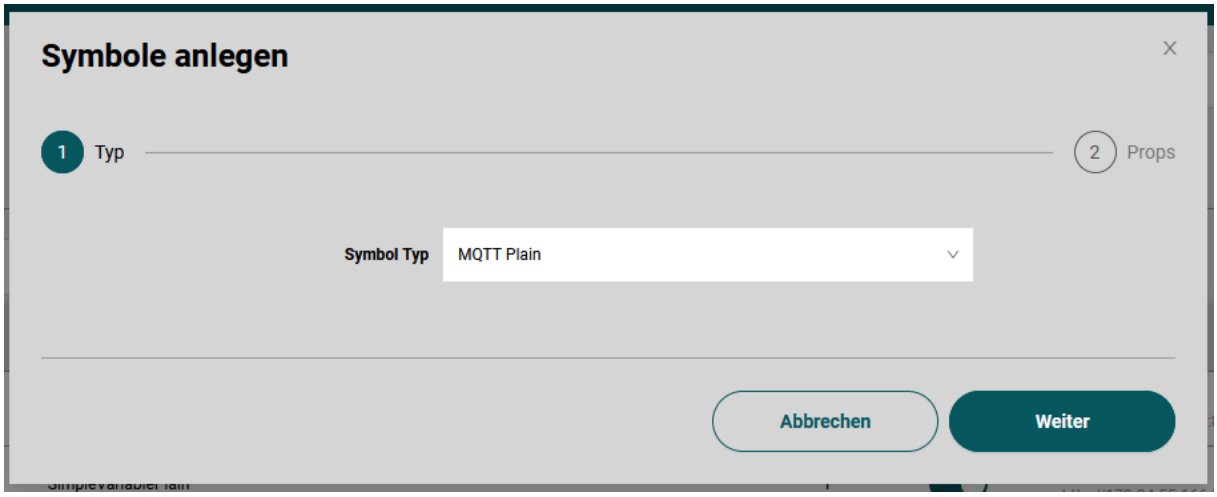
```
name data_anBool=f,data_anInt=3,data_nested_aString="world" 132909231230223
```

Achtung: Arrays innerhalb der Datenstruktur werden nicht unterstützt. Ein Array das Datenstrukturen enthält, kann hingegen konvertiert werden.

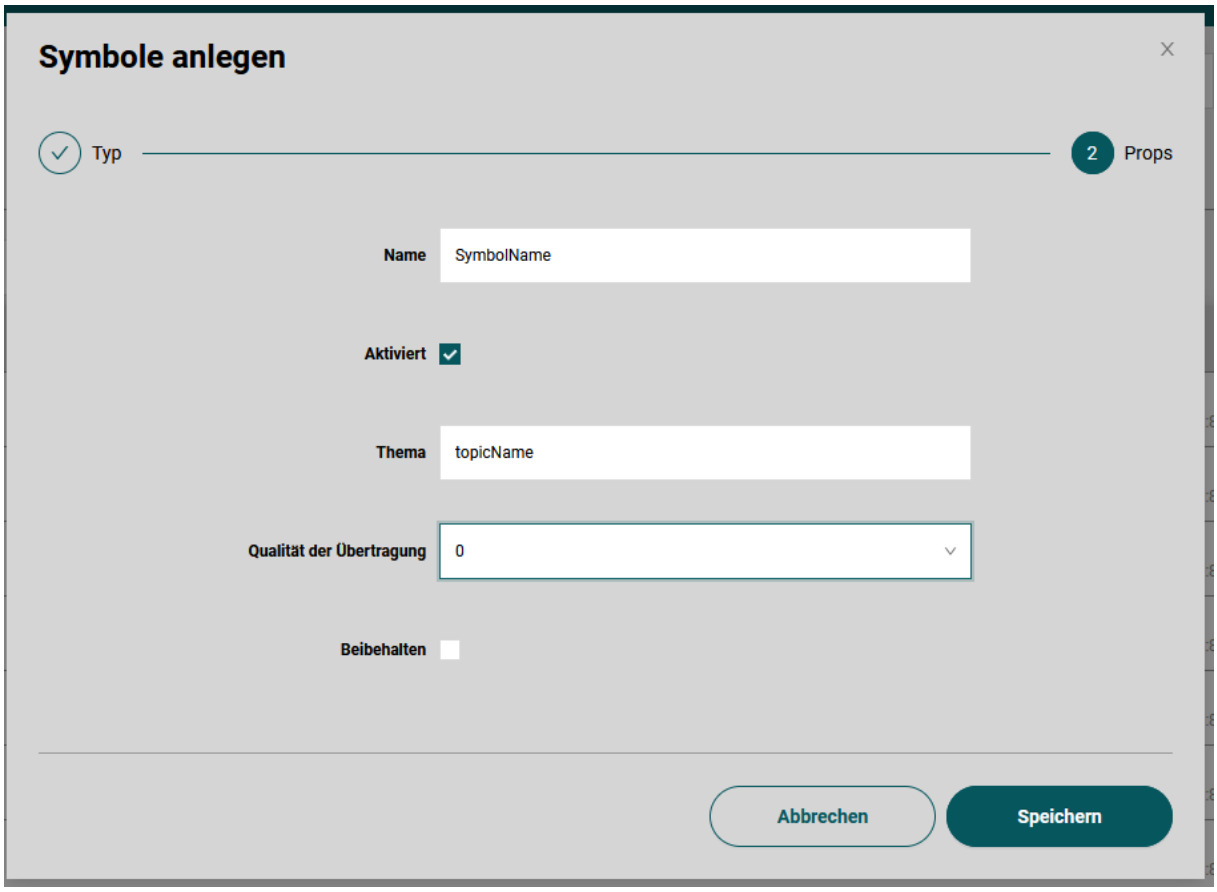
3.6.4 MQTT Publish Topic Plain erstellen

In der Regel werden MQTT Publish Topic Plain Symbole über das Automapping aus einen Input Symbol erstellt. Manchmal ist es auch hilfreich ein Symbol manuell anzulegen oder umzukonfigurieren.

Zunächst muss die Verbindung ausgewählt werden, für die das Symbol erzeugt werden soll. Im "Symbol erstellen" Dialog muss der Symbol Typ **MQTT Plain** ausgewählt werden.

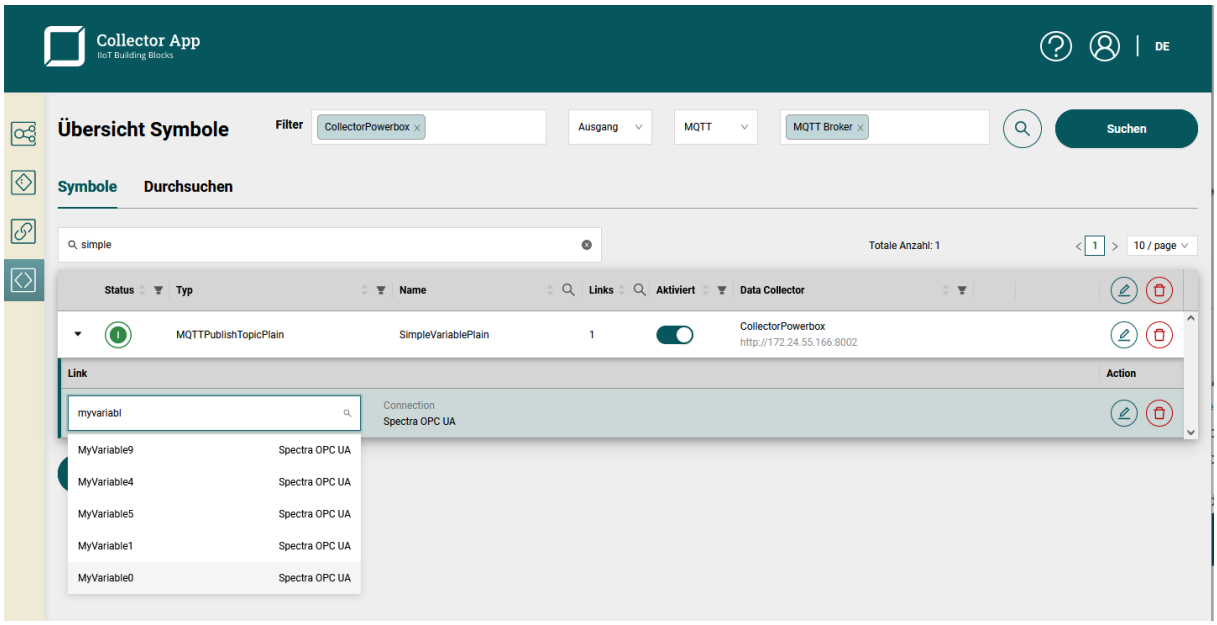


Auf der zweiten Seite können die grundlegenden Einstellungen vorgenommen werden.



Der **Name** wird in der App angezeigt. Das **Thema** entspricht dem MQTT Topic auf dem die Datenpunkte gespeichert werden. Die Qualität der Übertragung kann ausgewählt werden. Mit Beihalten lässt sich das Verhalten im Broker steuern.

Abschließend kann das Input Symbol in der aufgeklappten Zeile ausgewählt werden.



Die Datenpunkte werden als JSON versendet. Im folgenden werden Beispiele für Datentypen und ihr JSON Ergebnis dargestellt. Zahlen:

```
413658032
```

Strings:

```
"hello world"
```

Arrays:

```
[1, 2, 3, 4, 5]
```

Datenstruktur:

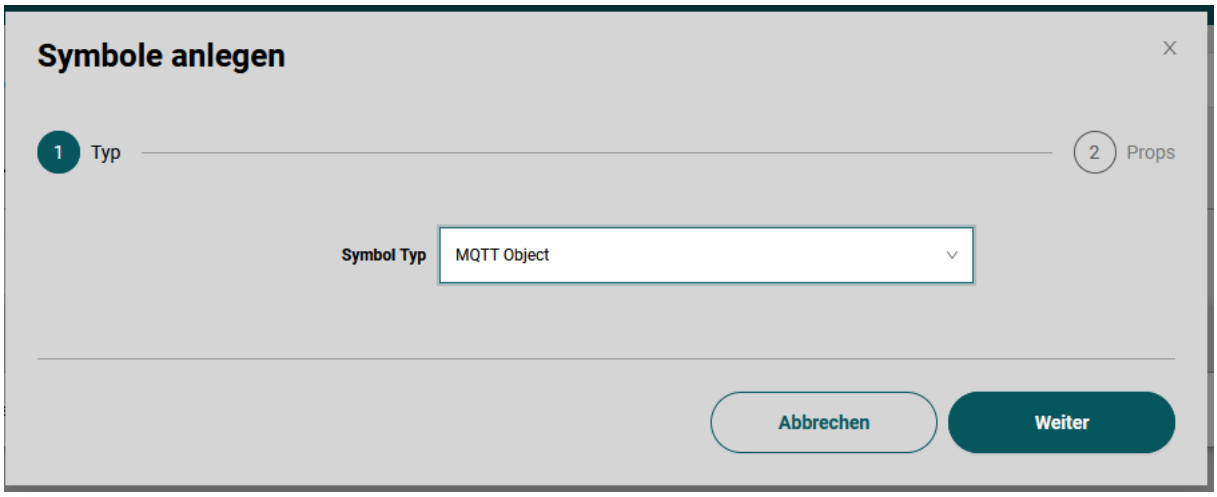
```
{
  "hello": "world",
  "anotherKey": false,
  "nested": {
    "structures": 2,
    "arePossible": -23.3
  },
  "anArray": [true, false, true, false]
}
```

3.6.5 MQTT Publish Topic Object erstellen

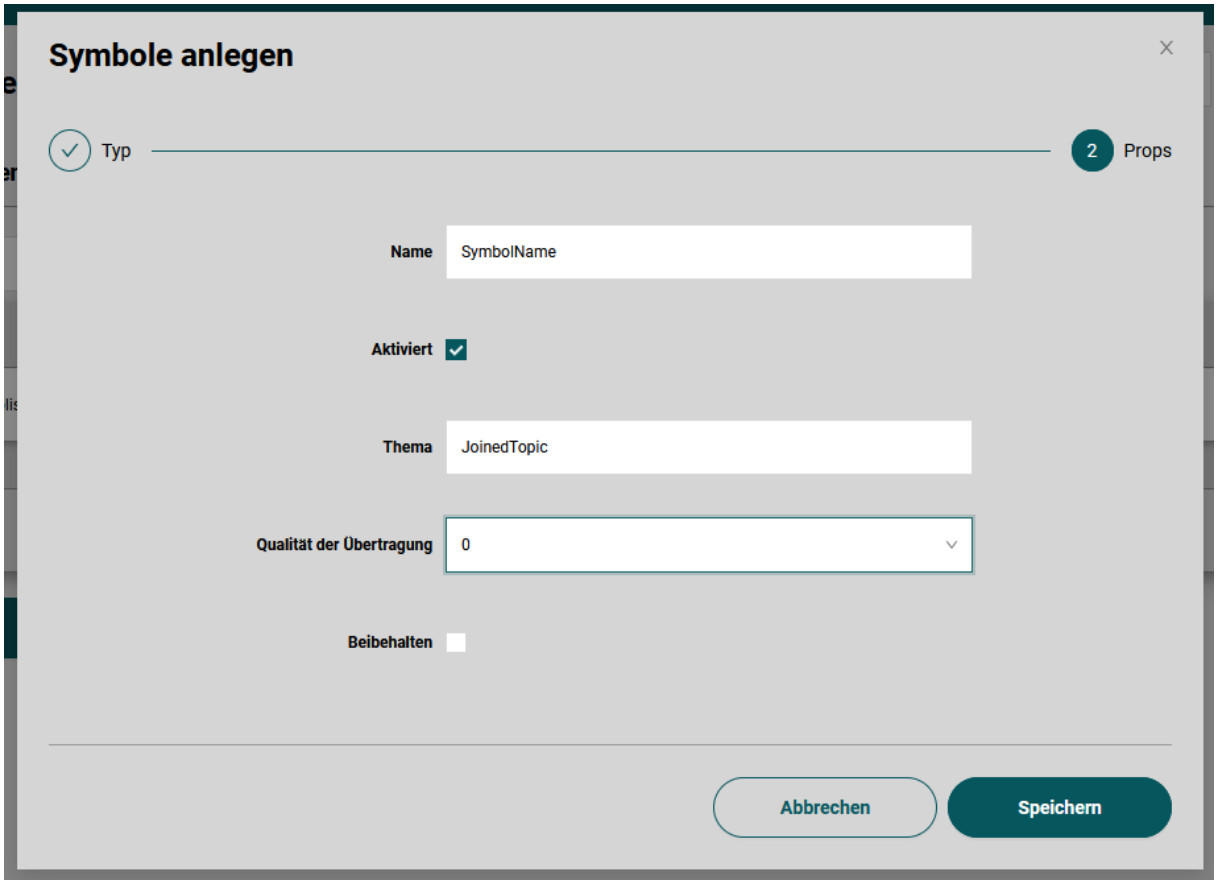
In der Regel werden MQTT Publish Topic Plain Symbole über das Automapping aus einen Input Symbol erstellt. Manchmal ist es auch hilfreich ein Symbol manuell anzulegen oder umzukonfigurieren.

Die Datenpunkte werden als JSON Objekt übertragen. Der Zeitstempel wird als ISO8061 mit Key 'timestamp' übermittelt. Am Ende des Abschnitt gibt es ein vollständiges Beispiel der Konfiguration und eines Datenpunktes.

Zunächst muss die Verbindung ausgewählt werden, für die das Symbol erzeugt werden soll. Im "Symbol erstellen" Dialog muss der Symbol Typ **MQTT Object** ausgewählt werden.

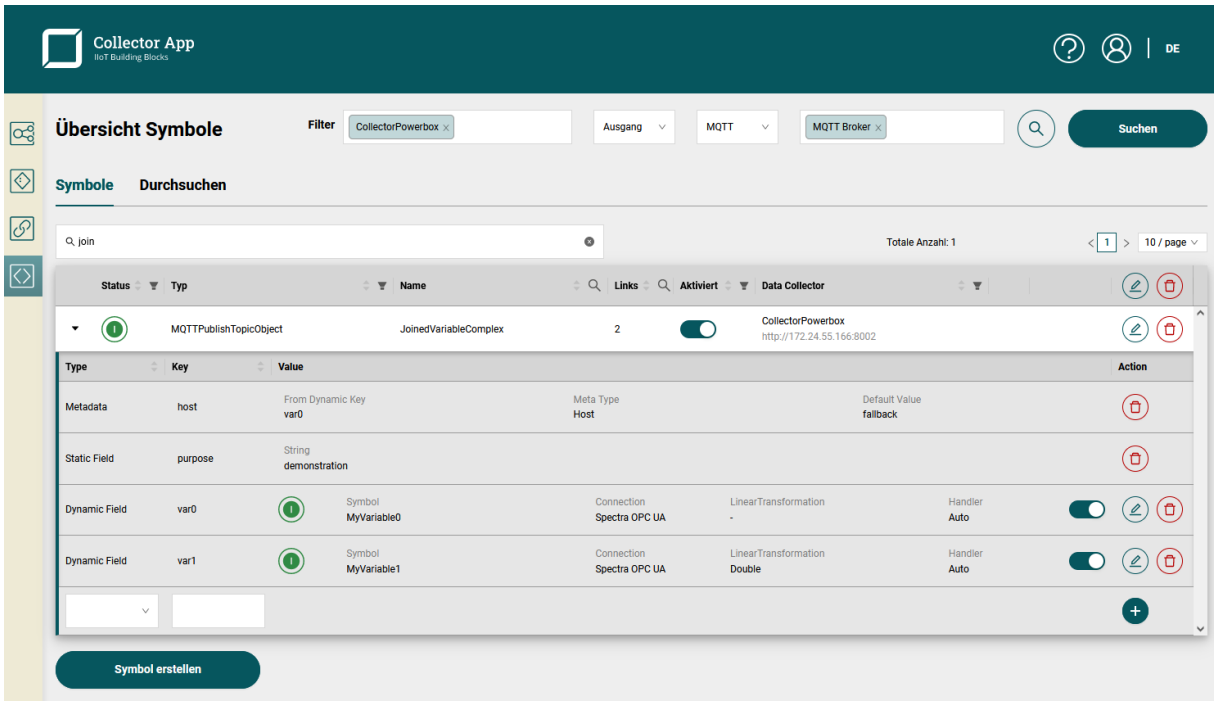


Auf der zweiten Seite können die grundlegenden Einstellungen vorgenommen werden.



Der **Name** wird in der App angezeigt. Das **Thema** entspricht dem MQTT Topic auf dem die Datenpunkte gespeichert werden. Die Qualität der Übertragung kann ausgewählt werden. Mit Beihalten lässt sich das Verhalten im Broker steuern.

Abschließend kann das Symbol und das zu konvertierende JSON Objekt in der aufgeklappten Zeile konfiguriert werden werden.



Es gibt drei Konfigurationsarten: 1. **Metadata**: verwendet für den Tag die Metadata Informationen des Input Symbols. Der Default Value wird verwendet um im Fall von fehlenden Metadaten einen Wert vorzugeben. kann der Wert nicht ermittelt werden (fehlt und kein Defaultwert), werden die Datenpunkte verworfen. Im Beispiel wird **Host** konfiguriert um den Hostname der Verbindung im Tag 'host' zu annotieren. Metadaten können nur von konfigurierten dynamischen Tags oder Fields verwendet werden. 2. **Static field**: Statischer Text der als String im Objekt gespeichert wird. 4. **Dynamic field**: Die Datenpunkte des ausgewählten Input Symbols werden gespeichert. Lineare Transformation und fester Datentyp können ausgewählt werden. In der Regel ist es sinnvoll den Datentyp auf *Auto* zu belassen.

Das erstellte JSON Objekt hat die folgende Struktur:

```
{
  "host": "ite-si.de",
  "purpose": "demonstration",
  "timestamp": "2022-07-28T07:16:30.798761Z",
  "var0": 413658032,
  "var1": 827316064
}
```

Achtung werden mehr als ein dynamisches Tag oder Field verwendet, dann gelten die Beschränkungen der Join-Funktionalität.

3.6.6 Beschränkungen der Join-Funktionalität

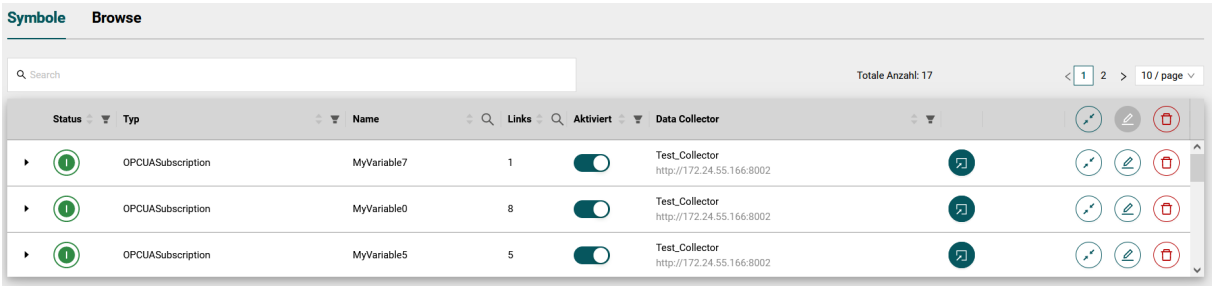
Der Data Collector ermöglicht es mehrere Input Symbole mit einem Output-Symbol zu verknüpfen. Die Datenpunkte werden über ein Best-Effort-Prinzip zusammengeführt. Konfiguriert werden kann dies beispielsweise in Influx Measurements oder in MQTT Publish Topic Object Symbolen über dynamische Fields.

Das Best-Effort-Prinzip beim Zusammenführen der Punkte bedeutet, dass der Data Collector keine Garantien zur Korrektheit der Zeitstempel beim Join durchführt. Die Datenpunkte werden zusammengeführt, wenn Sie am Data Collector ankommen. Kommen zwei Datenpunkte gleichzeitig (Netzwerkpaket) an, sollte der Zeitstempel exakt passen. Haben zwei Datenpunkte den gleichen Zeitstempel aber kommen aufgrund von Jitter 2 Sekunden zeitversetzt an, so ist in diesem Zeitraum ein "Fehler" in den Daten zu finden. Im Allgemeinen werden die Datenpunkte bei OPC UA Server bei OPC UA MonitoredItems sortiert nach Zeitstempel übertragen, so dass der Fehler selten auftreten sollte.

Werden Input Symbole aus unterschiedlichen Verbindungen verwendet, dann wird der Fehler mit den Zeitstempeln in der Regel vorhanden sein. Meistens werden mehrere Daten in Bündeln übertragen, was dazu führt, dass die Datenpunkte nicht zeitgleich eintreffen. Zu erwarten sind in der Regel mehrere 100 ms, aber der Zeitraum kann auch deutlich größer sein. Hat eine der beiden Datenquellen einen Verbindungsabbruch, kann die Ungenauigkeit noch höher sein.

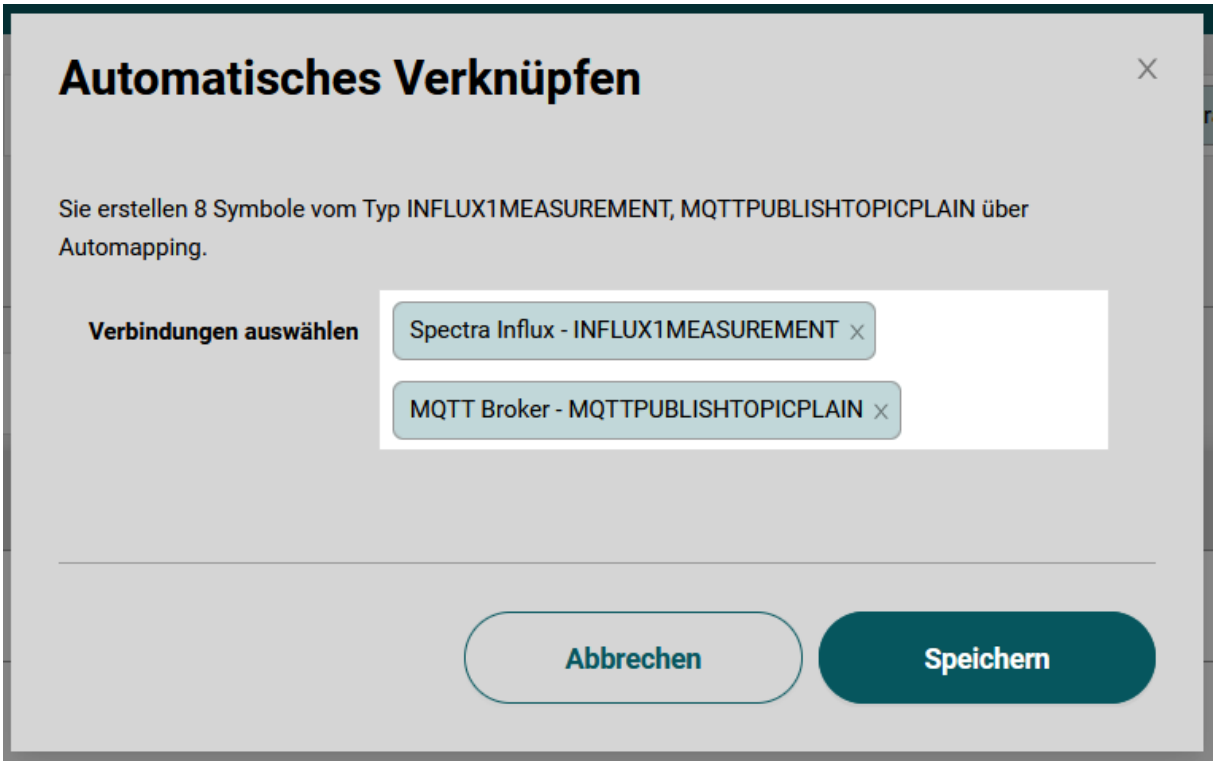
3.6.7 Mehrere Symbole gleichzeitig konfigurieren

In der Symboltabelle in der Kopfzeile befinden sich auf der rechten Buttons, die alle Symbole gleichzeitig in der Tabelle bearbeiten oder löschen können. Diese werden auch Mehrfach-Buttons genannt. Buttons am Ende einer Symbolzeile funktionieren nur auf diesem Symbol. Die Mehrfach-Buttons führen dieselbe Operation nur auf allen ausgewählten Symbolen auf der Tabelle auf. Möchte man nur einen Teil der Symbole gleichzeitig bearbeiten kann man die Auswahl der Symbole in der Tabelle mit den Filtern anpassen. In der folgenden Abbildung werden alle Symbole auf der Seite 1 und 2 der Tabelle beim Drücken eines Mehrfach-Buttons bearbeitet.



Automapping Button

Die Automapping Funktion ermöglicht schnell für ein oder mehrere Input-Symbole ein Output-Symbol pro Output Typ und Input-Symbol zu erstellen. Dazu müssen die Verbindungen der Output Typen ausgewählt werden.

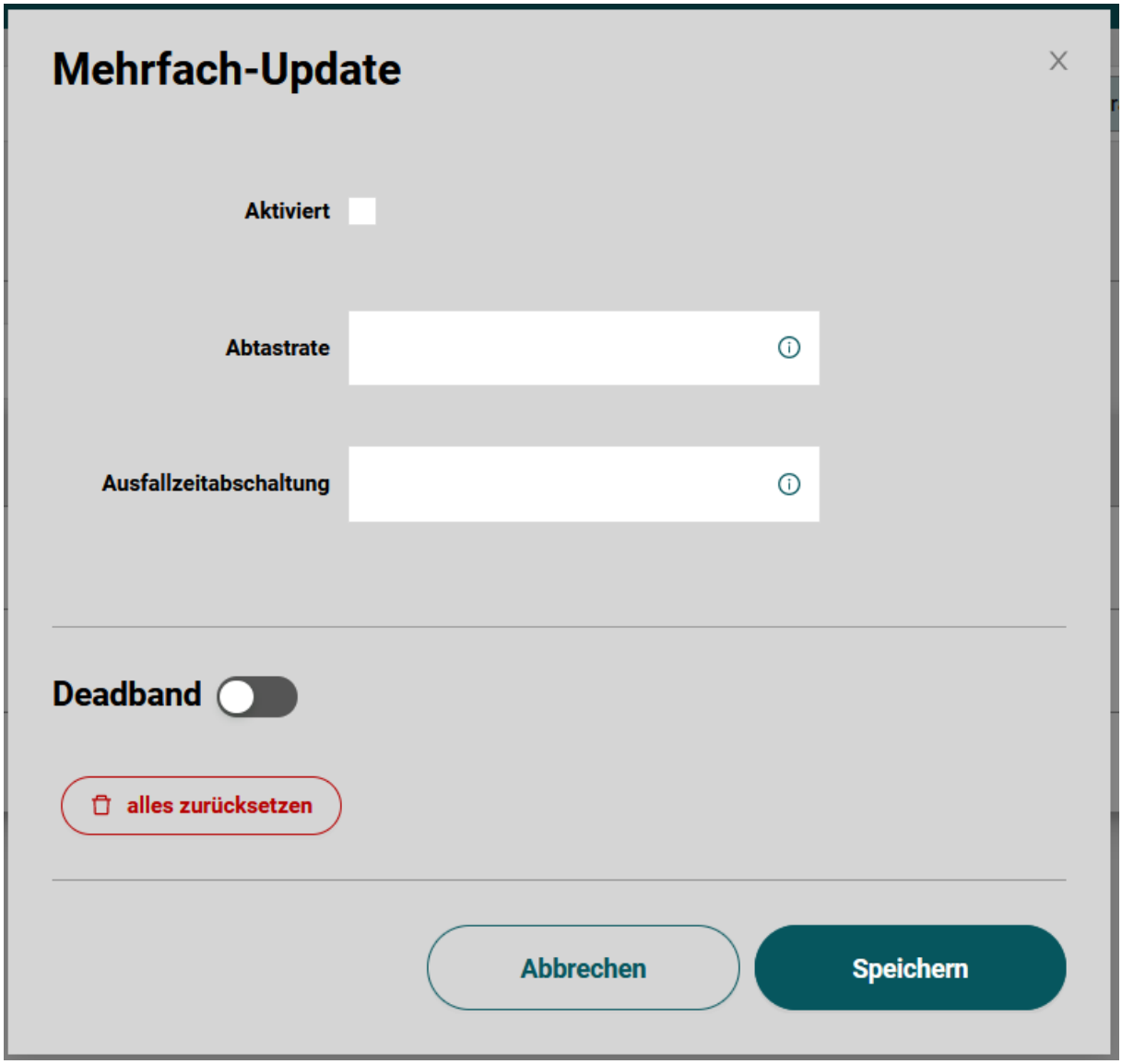


Beim normalen Automapping für ein einzelnes Input-Symbol wird ein Output-Symbol pro ausgewählter Verbindung erstellt. Wird das Automapping für mehrere Input-Symbole durchgeführt, wird fü jedes Input-Symbol ein Output-Symbol pro ausgewählter Verbindung erstellt.

Bearbeiten Button

Die Bearbeitung Button ermöglicht eine oder mehrere Symbole zu bearbeiten. Bei OPCUASubscription kann z.B. ein Interval, ein Timeout und ein Deadband eingestellt werden. Die Bearbeitungsfunktion funktioniert nur für Symbole des gleichen Symboltyps. Befinden sich unterschiedliche Typen, z.B. OPCUASubscription und OPCUABulkread in der Tabelle, muss man zunächst den gewünschten Typ über den Filter in der Tabellenkopfzeile auswählen.

Bei der normalen Bearbeitungsfunktion für einzelne Symbole kann der Deadband ausgeschaltet werden, in dem der Schalter deaktiviert wird und das Modal gespeichert wird.

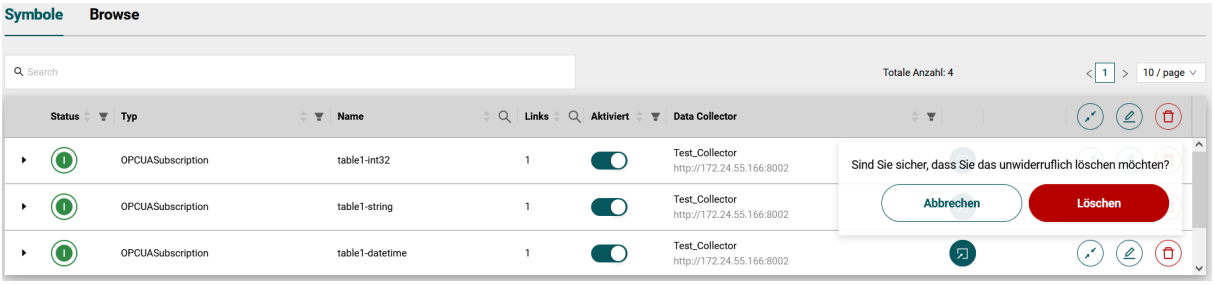


Bei der Mehrfach-Bearbeitungsfunktion gibt es einen zusätzlichen Button, mit der die Deadbands aller Symbole gleichzeitig zurückgesetzt werden kann. Hier muss ebenfalls das Modal gespeichert werden, damit das Zurücksetzen der Deadbands wirksam ist.

Löschen Button

Der Löschen Button ermöglicht es ein oder mehrere Symbole zu löschen. Einzelne Symbole werden durch den Löschen Button in der jeweiligen Symbolzeile gelöscht. Durch das Drücken des Mehrfach Löschen Button in der Kopfzeile werden alle in der Tabelle aufgelisteten Symbole gleichzeitig gelöscht.

Um unbeabsichtiges Löschen zu verhindern, wird vor jedem Löschvorgang noch eine Bestätigung des Nutzers abgefragt.



3.7 Benutzerverwaltung

Initial wird ein Admin Benutzer angelegt:

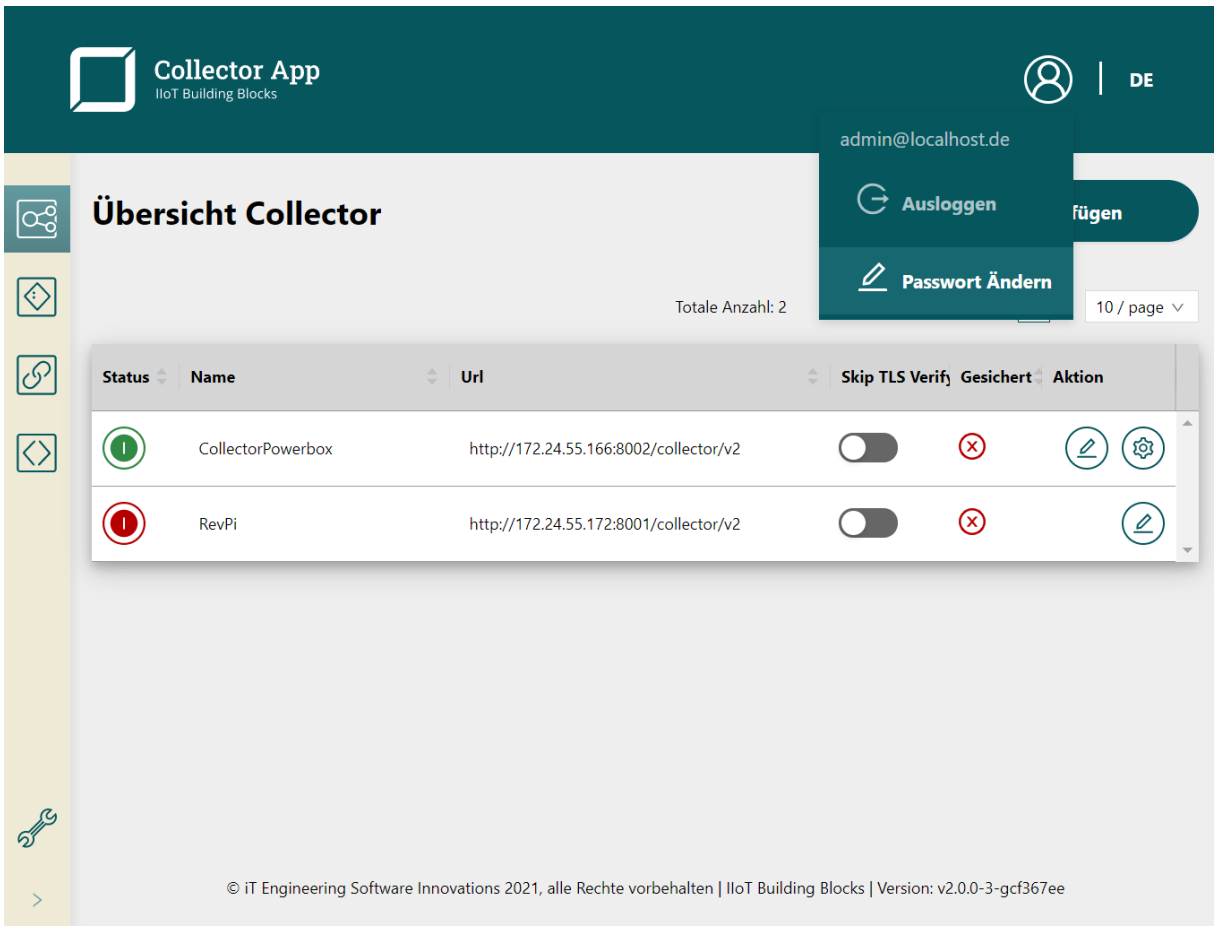
Benutzername	Passwort
admin	admin

Warning

Nach dem ersten Login mit dem Admin Benutzer sollte das Passwort geändert werden!

3.7.1 Passwort Ändern

Hovert man oben rechts über das Benutzermenü gibt es den Eintrag Passwort ändern:



3.7.2 Benutzerrollen

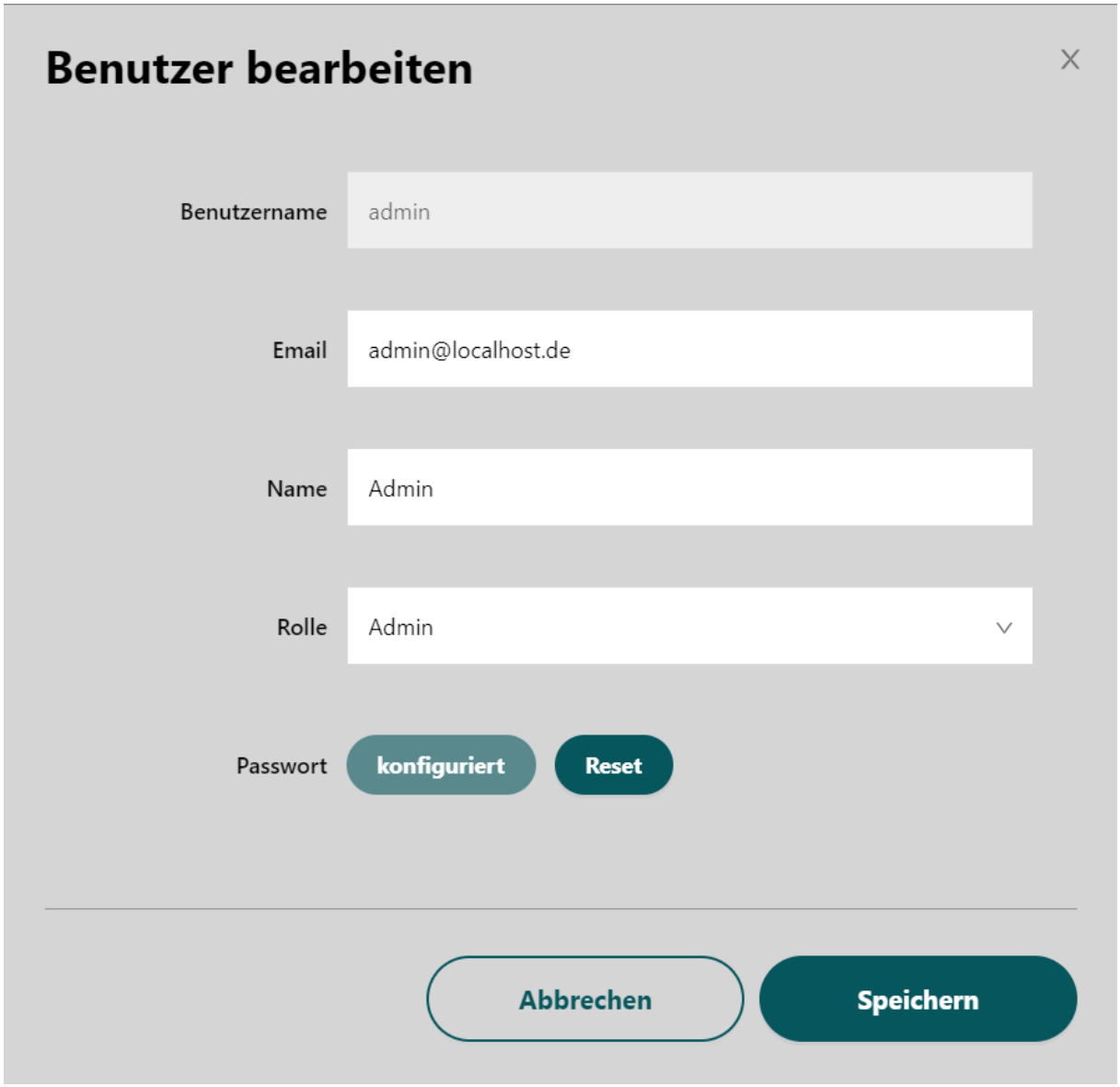
Rolle	Beschreibung
Admin	Die Rolle "Admin" hat die Berechtigung Benutzer zu verwalten. Das heißt Benutzer anzulegen, zu löschen, und Passwörter zurück zu setzen.
Developer	Developers können alles bis auf influx retention policies anlegen/bearbeiten/löschen und die Benutzerverwaltung.

3.7.3 Benutzer Anlegen / Bearbeitet

Unten links in den Einstellungen findet man die Benutzerverwaltung.

Jeder Benutzer hat einen eindeutungen Benutzername. Dieser kann nicht geändert werden. Der Name ist optional. Hier kann der volle Name des Benutzers eingetragen werden. Jedem Nutzer ist Rolle zugeordnet.

Das Passwort des Benutzers ist nicht einsehbar, da es nach dem Anlegen nur noch als hash gespeichert wird. Über den Reset Button kann es jedoch auf ein neues Passwort zurückgesetzt werden.



3.7.4 Passwort Zurücksetzen

Hat ein Benutzer sein Passwort vergessen muss er einen Administrator kontaktieren, der das Passwort in der Benutzerverwaltung der App zurückgesetzt kann.

Geht das Administrator Passwort verloren kann dieses über das Collector App Command Line Interface auf das Standard Passwort zurückgesetzt werden.

Auf windows wird die CLI bei der Installation in des Installationsverzeichnis mit installiert. Sie kann über die Command Prompt (cmd) ausgeführt werden:

```
>"C:\Program Files\iTE-SI\Collector-App\collector-app-cli.exe" reset-admin-password
```

Im Docker Container:

```
docker exec {container-name} /usr/share/collector-app/api/collector-app-cli reset-admin-password
```

3.8 Release Notes

3.9 v2.4.0 2024-04-05

Features

- Anlegen und editieren von OPCUAEventListener Symbol
- Claim-basierte Authorisierung für Collectoren und Verbindungen

Fixes

- Fehler beim aktualisieren eines Benutzers behoben, bei dem das Passwort zurückgesetzt wurde.

3.10 v2.3.0 2023-12-01

Features

- Das collector-app-cli reset-admin-password tool kann direkt die Datenbank als Parameter nehmen.
- Die iTE_SI_Collector-App.exe kann ein konfigurierbares Ausführungsverzeichnis (--working-dir) anstelle des %APPDATA% Ordners verwenden.
- UX Verbesserungen bei der Navigation zwischen Symbolen
- UX Verbesserungen bei OPCUA Subscriptions und Deadband
- Ansteuerung über unterschiedliche Hostnamen verbessert

Fixes

- InstallFolder property wird im Installer korrekt verwendet
- Verbessertes Browsen bei opc.tcp://localhost URLs und mehreren OPCUABrowsern
- Mehrere ApplicationErrors behoben, bei der normalen Verwendung der App

3.10.1 v2.1.1 (27.07.2022)

Features

- Added help button in main menu which links to the online documentation

Bugfixes

- Windows build start failures that were introduced in v2.1.0 were fixed

3.10.2 v2.1.0 (14.06.2022)

Features

- OPC UA BulkRead Symbol
  - CRUD-Operationen
  - Verknüpfen manuell oder mit Automapping
- Einheitlicher schritt-basierter Dialog zum Anlegen von Symbolen
- Die App kann nun unbekannte Verbindungs- und Symbol-Typen anzeigen und erlaubt basis Operationen:
  - Editieren (Ein- und Aus-Schalten)
  - Löschen
  - Status-Anzeige

Bugfixes

- Probleme beim Speichern von Collector Einstellungen
- Falsche Größe von editierbaren Tabellen-Zellen mit Evaluierung

3.10.3 v2.0.1 (26.11.2021)

Features

- Influx1Connection Authentifizierung mit Benutzername und Passwort
- Automapping für MQTTPublishTopicPlain und MQTTPublishTopicObject Ausgangs-Symbole
- MQTT Verbindung mit Ausgangs-Symbole
- Bessere Performance beim anlegen von vielen Symbolen auf einmal
- Verbindungen duplizieren
- Verbindungen importieren und exportieren

Bugfixes

- Beim erstellen von Metadatenspalten kann aus den dynamischen keys ausgewählt werden (nicht alle).
- Grafana Datasource Einstellungen.

3.10.4 v2.0.0 (25.06.2021)

Features

- Feature complete mit v1 (außer Konfigurationsdatei import/export)

4. iTE Data Collector

4.1 Allgemeines

Der Data Collector ist optimiert, um Messdaten auf der Feldebene zu erfassen. Er sammelt und speichert die Daten von der Maschine oder Anlage.

4.2 Installation

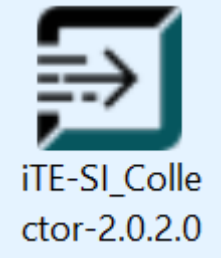
4.2.1 Windows

Installer herunterladen

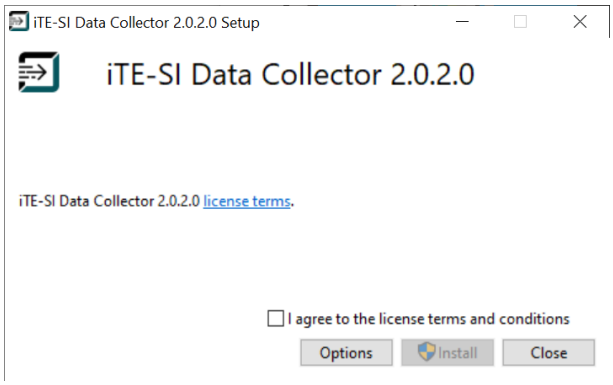
Der Windows Installer kann hier heruntergeladen werden: IIoT Building Blocks Downloads

Installieren

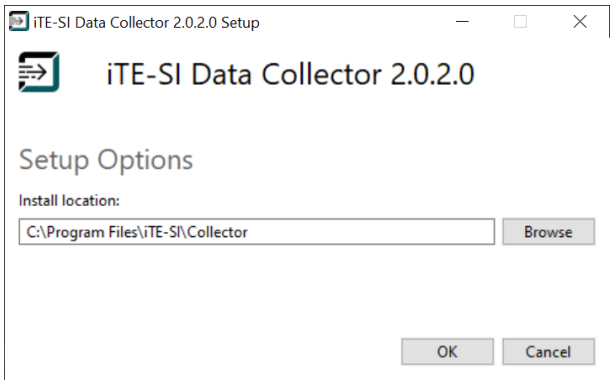
1. Den Installer ausführen:



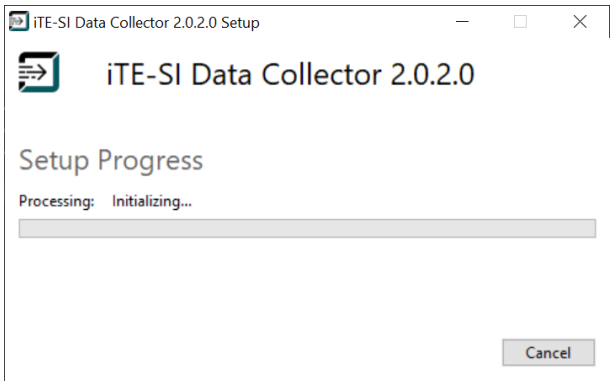
2. Die Endbenutzer-Lizenzvereinbarung lesen, gegebenenfalls akzeptieren und auf `Install` drücken. Ansonsten auf Cancel drücken um die Installation abzubrechen.



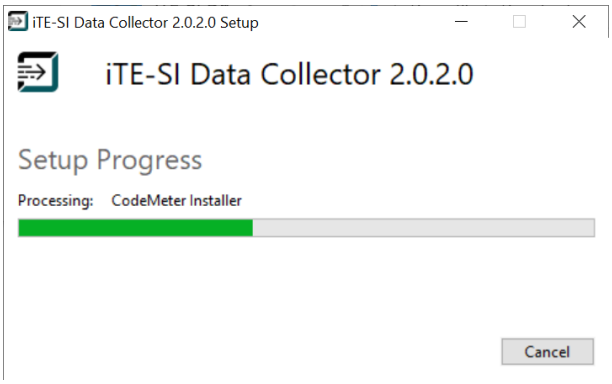
Optional: Auf Options drücken um den Speicherort zu wählen.



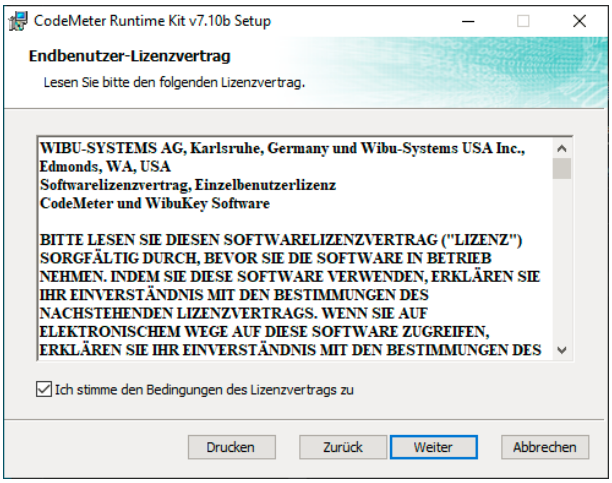
3. Kurz warten.



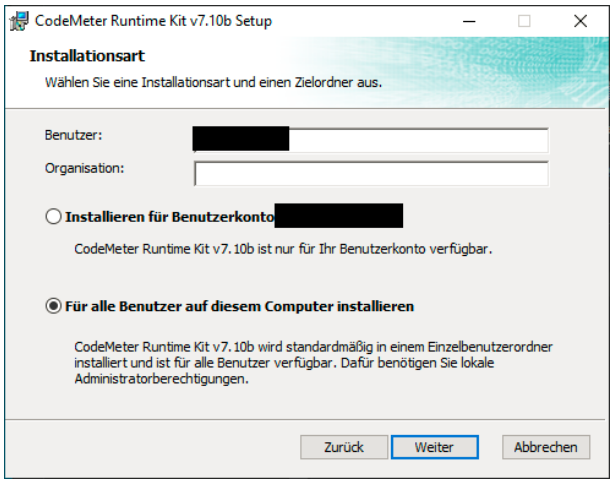
4. Der iTE-SI_Collector benötigt die CodeMeter Runtime. Falls diese nicht gefunden wird, wird sie über einen separaten Installer automatisch installiert. Ansonsten wird dieser Schritt übersprungen.

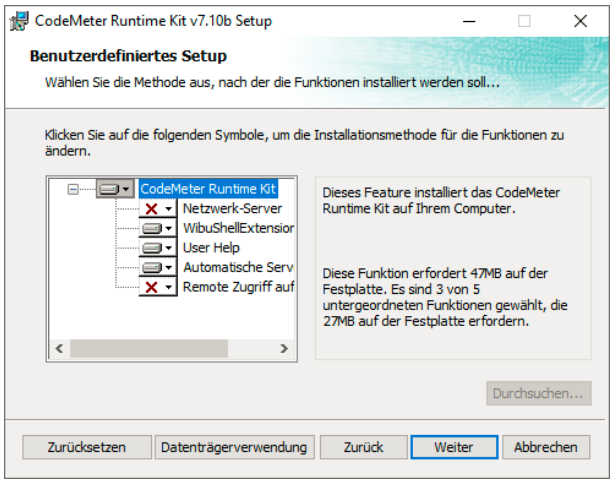


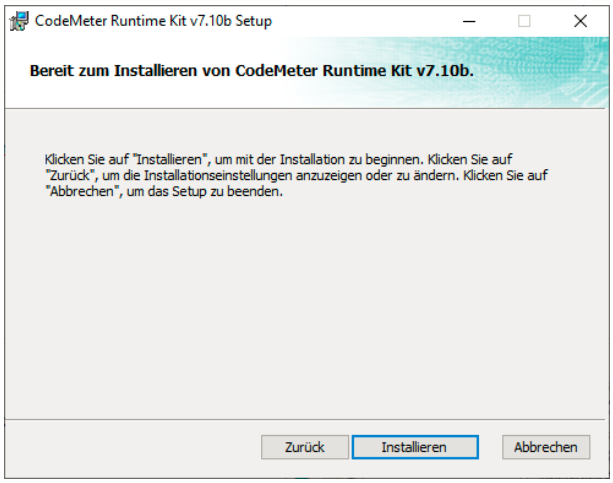
Auf den zweiten Seite vom Installer der CodeMeter Runtime muss man noch eine Endbenutzer-Lizenzvereinbarung lesen und gegebenenfalls akzeptieren.



Danach einfach ein paar mal auf Weiter drücken, Installieren und Fertig stellen.

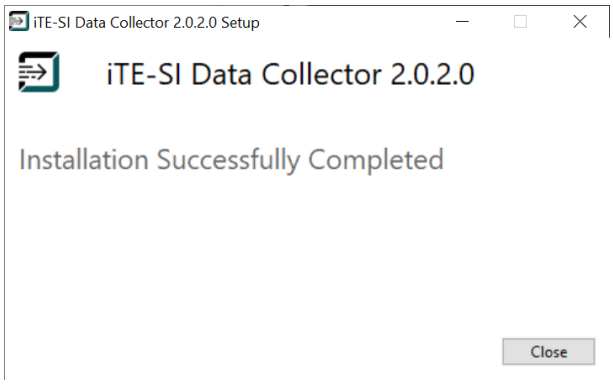






Nach der Installation der CodeMeter Runtime wird die Installation vom Collector fortgeführt.

5. Die Installation wurde erfolgreich abgeschlossen. Auf Close drücken.



6. Den Collector kann man jetzt über das Startmenü starten.

4.2.2 Docker

Das Docker Image kann hier herunterladen werden: IIoT Building Blocks Downloads

1. Laden des Images

Info

x.x.x.x muss durch die heruntergeladene Version ersetzt werden!

Info

Falls rancher-desktop mit containerd vernwedet wird, muss docker durch nerdctl ersetzt werden.

```
docker load -i iTE-SI_Collector-x.x.x.x.tar.gz
```

2. Container starten:

```
docker run -p 8001:8001 ite-si/collector:vx.x.x
```

Der Collector kann über Environment-Variablen konfiguriert werden. Zum Beispiel:

```
docker run -p 8001:8001 -e COLLECTOR_CONSOLE_LOG_LEVEL=debug ite-si/collector:vx.x.x
```

Mehr zur Konfiguration siehe: Collector Konfiguration

Um die Collector-Einstellungen persistent zu Speichern, muss der Konfigurationsordner im Container auf einen Ordner im Hostsystem gemappt werden.

```
docker run -p 8001:8001 -e COLLECTOR_CONFIG_DIR=/etc/ite-si/collector -v .config/collector:/etc/ite-si/collector ite
```

4.3 Konfiguration

4.3.1 Command Line Argumente

Die Command-Line Argumente nutzen Standardwerte, die mit Environment Variablen konfiguriert, aber per CLI überschrieben werden können.

run Command

Parameter	Beschreibung	Default
--console-log-level	Beschränkt das Loglevel der Console auf das Level	COLLECTOR_CONSOLE_LEVEL; 'info'
--log-dir	Setzt das Verzeichsis, in dem die Logdateien gespeichert werden. Der Collector legt das Verzeichnis an, wenn es noch nicht existiert. Der Collector muss Schreibrechte auf dem Ordner besitzen. Relative Pfade sind erlaubt und sind relativ zum 'Working directory'.	COLLECTOR_LOG_DIR; COLLECTOR_WORK_DIR/logs
dir	Das Ausführungsverzeichnis des Collectors. Der Collector legt Dateien entsprechend des Verzeichnisses an.	Linux: COLLECTOR_WORK_DIR;'' Docker: COLLECTOR_WORK_DIR;'/opt/ite-si/collector/tmp/' Windows: COLLECTOR_WORK_DIR;'%APPDATA%\iTE-SI\Collector\'
--config-dir	Das Konfigurationsverzeichnis des Collectors. Der Collector legt das Verzeichnis und leere Konfigurationsdateien an, wenn es nicht existiert.	Linux/Windows: COLLECTOR_CONFIG_DIR;$working-dir Docker: COLLECTOR_CONFIG_DIR;'/opt/ite-si/collector/etc/'

migrate-v1 Command

Setzt eine Collector v1 Konfigurationsdatei (config.toml) in die passenden Collector v2 Konfigurationsdateien um. Weitere Details gibt es über die Command-Line mit dem folgenden Befehl

```
iTE-SI_Collector migrate-v1 --help
```

4.3.2 Environment Variables

Variable	Beschreibung
COLLECTOR_CONSOLE_LOG_LEVEL	Gibt den Default Wert für das Console Log Level beim Start des Collectors.
COLLECTOR_LOG_DIR	Gibt den Default Wert für das Log Verzeichnis an
COLLECTOR_WORK_DIR	Gibt das Standard-Ausführungsverzeichnis an
COLLECTOR_CONFIG_DIR	Gibt das Standardverzeichnis für die Konfigurationsdateien (service.toml, objects.toml) an

4.3.3 Konfigurationsdateien

Der Collector verwendet zwei Konfigurationsdateien: service.toml und objects.json. Die Konfigurationsdateien werden angelegt, wenn Sie nicht existieren. Die 'service.toml' soll manuell editiert werden, die objects.json soll **nicht** vom Nutzer bearbeitet werden. Diese Datei wird im laufendem Betrieb modifiziert.

service.toml

Beispiel-Konfiguration

```
version = 1

[general]
hideConsole = false

[http]
port = 8001
token = "unsecure-plaintext-token"

[http.certificate]
certificateFile = "./cert.pem"
keyFile = "./key.pem"
```

Beschreibung der Felder

Section	Feld	Typ	Optional	Beschreibung	Default
	version			Internes Format der Json Datei. Dient zum automatischem Upgrade, bei Format Änderungen	1
general	hideConsole	bool		Versteckt das Fenster unter Windows.	false
http	port	uint16		Die REST API wird unter diesem Port gestartet	8001
http	token	string	true	HTTP Bearer Token für Authentifizierung der REST API. Kann weggelassen werden um	
http.certificate			true	HTTP Zertifikat zur Verwendung von ssl/tls in der REST API	
http.certificate	certificateFile	string		Pfad zum HTTPs Zertifikatim PEM-Format. Relativ zum Konfigurationsverzeichnis.	
http.certificate	keyFile	string		Pfad zum HTTPs Private Key im PEM-Format. Relativ zum Konfigurationsverzeichnis. Datei darf nicht Passwort geschützt sein.	

4.4 Release Notes

4.4.1 v2.4.0 (05.04.2023)

Features

- Globale JSON output Einstellungen der REST API hinzugefüt (MQTT output symbols)
- Automatische Zertifikatsgenerierung der OPCUAConnection

Fixes

- Verbessertes Influx URL handling, z.B. https://myhost/influxdb
- Collector verbindet sich korrekt mit mqtts:// URLs
- HTTP REST Fehlerbehebungen für OPCUAEventListener Symbol (query, patch)
- Kleinere Verbesserungen beim Herunterfahren des Collectors

4.4.2 v2.3.0 (01.12.2023)

Features

- "name" Query Parameter bei /connections und /symbols Endpuntken
- Neuer Symboltyp OPCUAEventListener

Fixes

- Einführung (default) HTTP Connect Timeout (5 Sekunden) um Fehler bei der Influx Konfiguration schneller zu entdecken
- OPC UA Bool Werte werden korrekt transformiert (influx/mqtt)
- Hostname handling von localhost bei Connections in den Metadaten und beim starten
- Kompabilität zu pythonua Library verbessert

4.4.3 v2.2.6 (22.08.2023)

4.4.4 Fixes

- Verbesserte Interoperabilität mit alten Collector-App Versionen.
- Fehler beim Ausführungen von migrate-v1 Command behoben.
- Erneutes Hochfahren der OPCUAConnection bei Änderungen an Zertifikat oder Credentials
- Absturz bei fehlerhaft konfigurierten Zertifikat bei OPCUAConnections behoben

4.4.5 v2.2.5 (02.03.2023)

Fixes

- Fehlender Lizenzstatus in der REST API korrigiert.

4.4.6 v2.2.4 (15.02.2023)

Features

- Neue Option [log] flushOnMessage in service.toml
- Neue Option [general] debugEventloop in service.toml
- Verbesserter Umgang mit Custom Datatypes und Siemens PLC

Bugfixes

- REST API Aufrufe funktionieren auch bei großen Anfragen schneller
- OPCUASubscription: verbessertes Handling mit der Collector-App
- Anzeige von Statistiken in der App verbessert.

4.4.7 v2.2.3 (17.10.2022)

Bugfixes

- Absturz bei Collector Statistiken bei abgeschalteter InfluxDB

4.4.8 v2.2.2 (2022-09-22)

Bugfixes

- Crash mit Windows Collector, wenn eine verschlüsselte Verbindung durch den OPC UA Server beendet wird
- Log-Dateien beinhalten verbesserten Output, wenn ein Crash des Collectors passiert

4.4.9 v2.2.1 (2022-09-09)

Fixes

- Installation Fehler, wenn eine neuere Version von vcredist.exe auf dem Rechner installiert ist.

4.4.10 v2.2.0 (03.08.2022)

Collector unterstützt das Zusammenführen von mehreren Datenpunkten (Join) in Influx1Measurement und MQTTPublishTopicObject.

Info

Collector v2.2.0 kann mit der Collector App v2.1.X vollständig konfiguriert werden.

Features

- Die Konfiguration von mehreren Input-Symbole in Influx1Measurement und MQTTPublishTopicObject ist möglich
- Der Join-Vorgang operiert auf den Zeitstempeln der Datenpunkten und funktioniert nach einem Best-Effort Prinzip

Bugfixes

- Upgrade der intern verwendeten Libraries
- In Ausnahmefällen hing der Data Collector beim Schließen der Software
- Metadata 'Host' hat für OPC UA BulkRead Symbole gefehlt
- Lineare Transformation wurde nicht im MQTTPublishTopicObject umgesetzt

4.4.11 v2.1.0 (14.06.2022)

Neues OPC UA BulkRead Input-Symbol. Unterstützung für MQTT und zwei dazugehörige Output-Symbole mit denen Daten zum Broker im JSON format geschrieben werden können.

Info

Mit der Collector App in Version 2.1.X können die neuen Symbole konfiguriert werden. Mit der Collector App in Version 2.0.X können die neuen Symbole weder angezeigt noch konfiguriert werden.

4.4.12 Features

- OPC UA BulkRead Input-Symbol
  - OPC UA Server verwendet Trigger-Mechanismus um neuen Datenpunkt zu signalisieren
  - Gleichzeitiges Auslesen (BulkRead) mehrerer OPC UA Knoten und zusammenführen in Datenstruktur.
  - Optionaler Handshake with OPC UA server: Der Data Collector hat den neuen Datenpunkt (Struktur) gelesen; oder der Data Collector hat den Datenpunkt (Struktur) in das Output-Symbol geschrieben. Influx und MQTT Output-Symbole werden im Handshake unterstützt.
- MQTT Unterstützung zum Schreiben von Datenpunkten an einen MQTT Broker
  - Datenpunkte werden als JSON übertragen
  - Konfiguration der MQTT Topics
  - Übertragung reiner Datenpunkte (MQTTPublishTopicPlain) oder als Json-Objekt mit Timestamp, Status, Wert und anderer Metadata (MQTTPublishTopicObject)

4.4.13 Bugfixes

- Probleme mit der Dateistruktur im Docker-Image behoben

4.4.14 v2.0.1 (20.09.2021)

Features

- Upgrade des OPC UA Stacks zur Unterstützung von string-basierten NodeIds in Typbeschreibungen

Fehlerbehebungen

- Probleme mit Abtastintervallen in OPCUASubscription behoben
- Bumped abhängige Bibliotheken
- Korrigierte Link-Informationen in der Collector-App angezeigt

4.4.15 v2.0.0 (24.06.2021)

- Erstes Release der Version 2.0.

5. iTE OPCUA Browser

5.1 Allgemeines

Der OPCUA Browser ermöglicht es der Collector App den OPC UA Addressraum zu durchsuchen. Der Benutzer kann dadurch unterschiedliche OPC UA Knoten auswählen, welche anschließend vom Collector gespeichert werden.

5.2 Installation

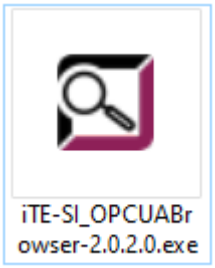
5.2.1 Windows

Installer herunterladen

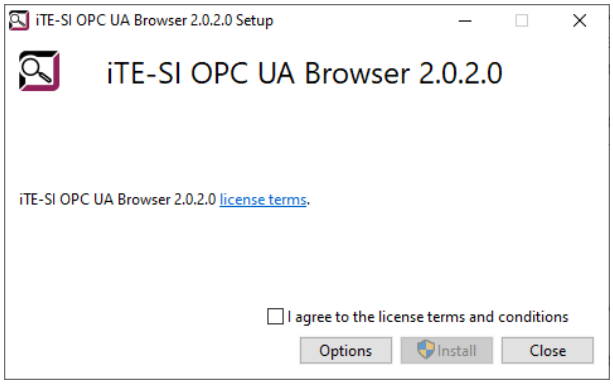
Der Windows Installer kann hier heruntergeladen werden: IIoT Building Blocks Downloads

Installieren

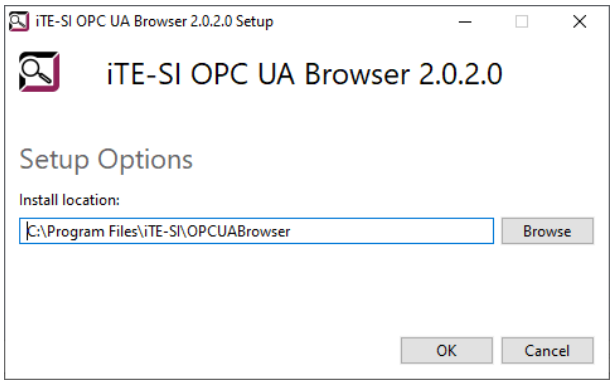
1. Den Installer ausführen:



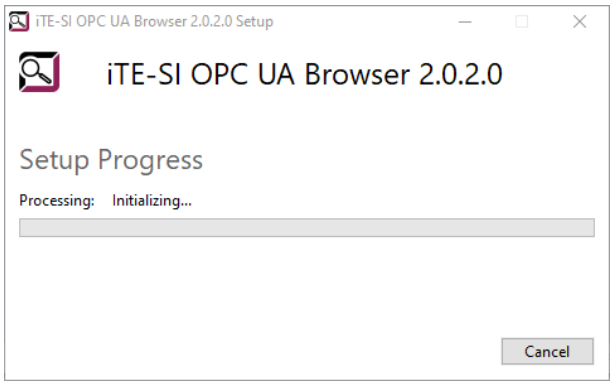
2. Die Endbenutzer-Lizenzvereinbarung lesen, gegebenenfalls akzeptieren und auf `Install` drücken. Ansonsten auf Cancel drücken um die Installation abzubrechen.



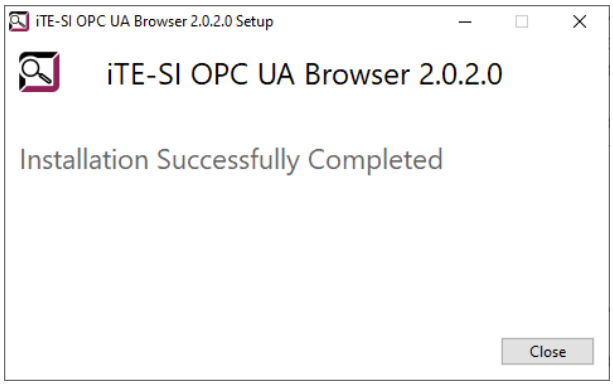
Optional: Auf Options drücken um den Speicherort zu wählen.



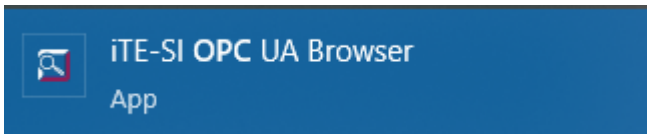
3. Kurz warten.



4. Die Installation wurde erfolgreich abgeschlossen. Auf Close drücken.



5. Den OPCUA Browser kann man jetzt über das Startmenü starten.



5.2.2 Docker

Das Docker Image kann hier herunterladen werden: IIoT Building Blocks Downloads

1. Laden des Images

Info

x.x.x.x muss durch die heruntergeladene Version ersetzt werden!

Info

Falls rancher-desktop mit containerd verwendet wird, muss docker durch nerdctl ersetzt werden.

```
docker load -i iTE-SI_OPCUABrowser-x.x.x.x.tar.gz
```

2. Container starten:

```
docker run -p 8001:8001 ite-si/opcua-browser:vx.x.x
```

Der Browser kann unterschiedlich konfiguriert werden: Browser Konfiguration

Um die Browser-Einstellungen persistent zu Speichern, muss der Konfigurationsordner im Container auf einen Ordner im Hostsystem gemappt werden.

```
docker run -p 8080:8080 -v /path/to/config.json:/opt/ite-si/opcua-browser/config.json ite-si/opcua-browser:vx.x.x
```

5.3 Konfiguration

Der OPCUA Browser verwendet die Konfigurationsdatei config.json. Diese befindet sich im Docker container unter `/opt/ite-si/browser/config.json` und unter Windows in `%APPDATA%\iTE-SI\OPCUABrowser/config.json`.

Beschreibung der Felder:

Feld	Typ	Beschreibung	Default
Http.Port	int	REST Port	8080
MaxNodesPerRead	int	Maximale Anzahl an OPCUA Knoten die gleichzeitig gelesen werden	Server response
MaxNodesPerBrowse	int	Begrenzt die maximale Anzahl an Knoten im BrowseRequest	Server response
UseSecurity	bool	Default Einstellung für die Verschlüsselung. Kann per Request Basis überschrieben werden.	false
ClientPrivateKey	string	Dateipfad zur OPC UA Client Key-Datei in DER form.	
SecurityPolicyUri	string	Die Standard OPCUA Scurity Policy URI; http://opcfoundation.org/UA/SecurityPolicy#None oder http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256 falls Zertifikat hinterlegt.	http://opcfoundation.org/UA/SecurityPolicy#None
MessageSecurityMode	string	OPCUA Message Security Mode.	None
ServerAuthToken	string	Das (Bearer) Authentification Token der REST API	
ServerCertificateFile	string	Server Zertifikat der REST API in PEM Format	
ServerPrivateKeyFile	string	Server Private-Key-Datei der REST API in PEM Format	

5.4 Release Notes

5.4.1 v2.2.0 (2024-04-05)

Features

- Automatische Generierung von Zertifikaten für die OPC UA Verbindung
- EventFields können über die REST Schnittstelle abgefragt werden (OPCUAEventListener)
- Zusätzlicher MaxBrowseDepth Parameter beim Browsing möglich

Fixes

- Fehlerhafte Konvertierung von ByteString-basierten NodeIDs oder Werten

5.4.2 v2.1.1 (2023-09-11)

Features

- TestTimeoutInMs Konfiguration für den Test-Endpunkt

Fixes

- Erreichbarkeitstest /ping verbessert.
- Tray-Probleme behoben

5.4.3 v2.1.0 (2023-08-22)

Features

- Neue Optionen: "QueryValues, "QueryTypenames","QueryDatatypes", "DebugMode" in config.json

Fixes

- Verbesserte Interoperabilität zu Collector-App

5.4.4 v2.0.4 (2022-09-27)

Fixes

- Verschlüsselte Verbindungen konnten nicht mit aktueller Collector App aufgebaut werden.

5.4.5 v2.0.3 (2022-09-09)

Fixes

- Installation konnte aufgrund neuerer installierter Version von vcredist.exe fehlschlagen.

5.4.6 v2.0.2 (2021-11-26)

Features

- Der Installer ist signiert.

5.4.7 v2.0.1 (2021-10-12)

Fixes

- Installation konnte aufgrund installierter vcredist.exe fehlschlagen.

5.4.8 v2.0.0 (2021-06-24)

- Erstes Release der version 2.0.