IIoT Building Blocks

Collect

iT Engineering Software Innovations GmbH

Table of contents

1. Introduction

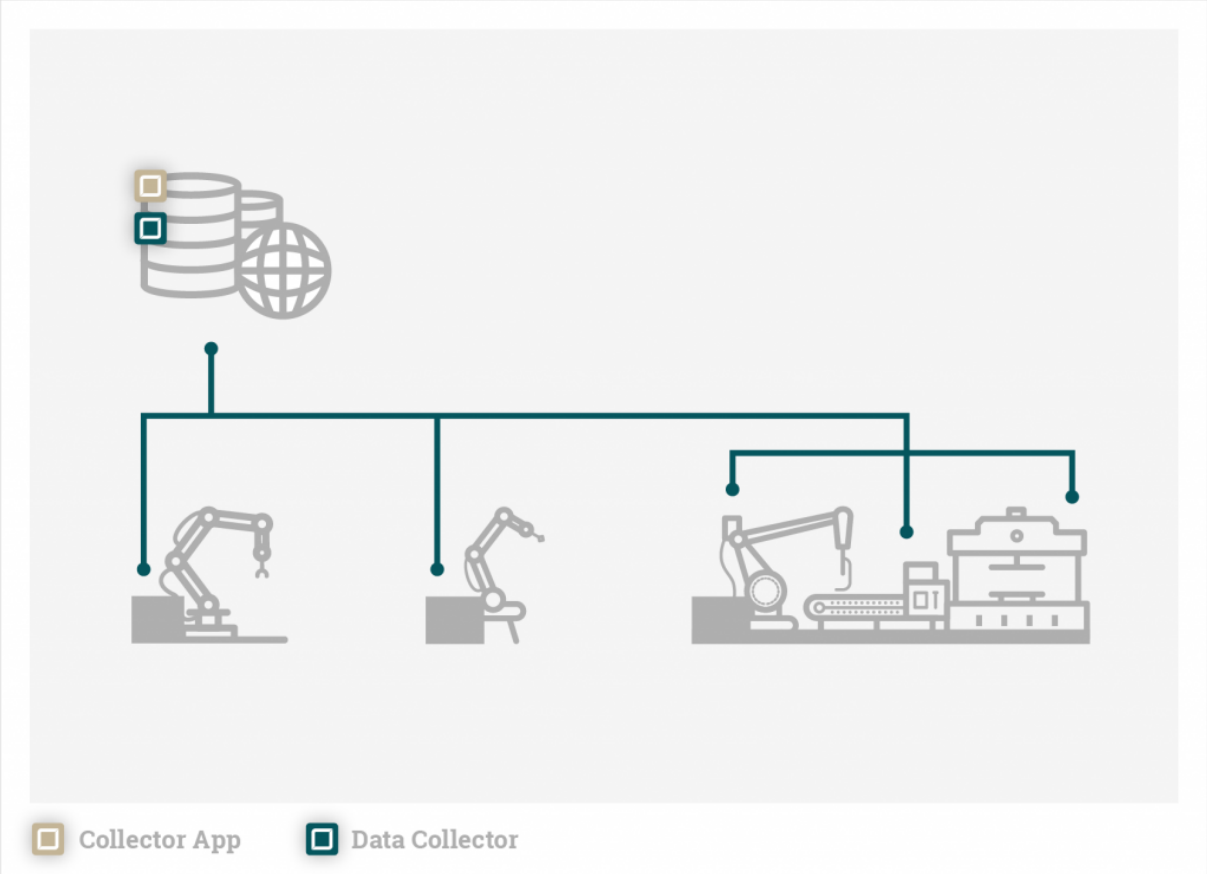
1.1 Deployment variants

The components of the IIoT Building Block Stack are very modular and based on a microservice architecture. Therefore, the deployment is as flexible as possible and fits very well into existing infrastructures.

1.1.1 One server installation

The entire stack is installed on one machine/server:

- Collector
- Collector App
- Database

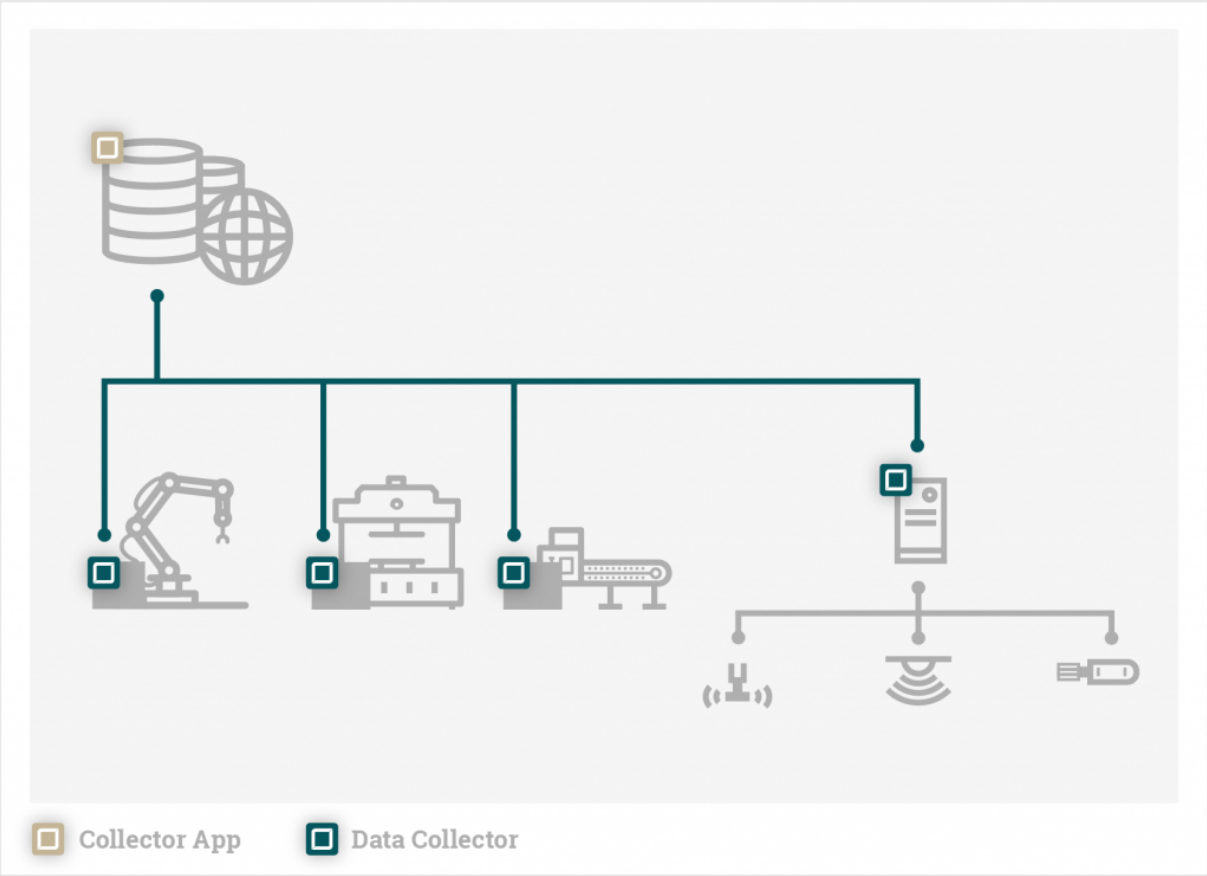


The Collector collects data from different machines and assets via a network interface and stores it on the central database installed next to the Collector on the same machine. Via the Collector App, also on the same server, one more Collector has to be configured.

If the network is disturbed, no data can be transferred. Data may be lost under certain circumstances.

1.1.2 Distributed system

Several Collector instances can be installed in the network. The Collector can even run directly on PC-based controllers. The Collector App and the database are located on a central machine computer/server.



In the Collector App all collectors in the network can be configured via a user interface. The collectors can buffer the data in case of a network failure and send suggestive over the network to the database after the failure is fixed.

1.2 Getting Started

For a quick start, it is recommended to install the entire stack on a machine first.

1.2.1 Quick start with docker-compose

The easiest and most reliable way to deploy the stack is via Docker, since the applications run in defined container environments and thus have no dependencies on the host system. In addition, the containers can be completely removed from the system after termination.

Installing Docker (Linux, Mac and Windows): Get-Docker

A license is required for Windows and Mac Docker installation. Alternatively, Rancher-Desktop can be installed as a container runtime. If Rancher-Desktop is used with containerd the docker cli must be replaced by nerdctl in the following.

To be able to start multiple containers directly with one command, Docker Compose is used. In a docker-compose file multiple containers can be defined with configuration and linking. In this case, the containers are as follows:

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

Therefore, make sure that you have loaded the images for Collector, Collector-App and OPC UA Browser according to the installation instructions.

The InfluxDB and Grafana image is automatically loaded from the official online repository. InfluxDB is used as demo data sink and Grafana is used to display data stored by the Data Collector.

1. Creating a folder

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

2. Creating the Shard Volume Folder

Since containers do not have persistent storage, all persistent files and databases must be stored on the host system so that the state is not lost after the container reboot.

```
mkdir docker
```

The folder will be used in the Container volumes.

3. Creating the compose file

Linux

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

Windows

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

Open file with your favourit editor and past that in:

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

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

4. Starting the stack

```
docker-compose up
```

or

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

To start in the background:

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

or

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

5. Launch Collector App

Once launched, the app is accessible in the browser at http://localhost:4000/collector-app.

Default login:

username	password
admin	admin

An introduction to the Collector App can be found here.

6. Stopping/Removing the Stack

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

or

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

1.2.2 Quick start under Windows (native)

1. Installer

First, the components:

- Collector
- Collector-App
- OPC UA Browser

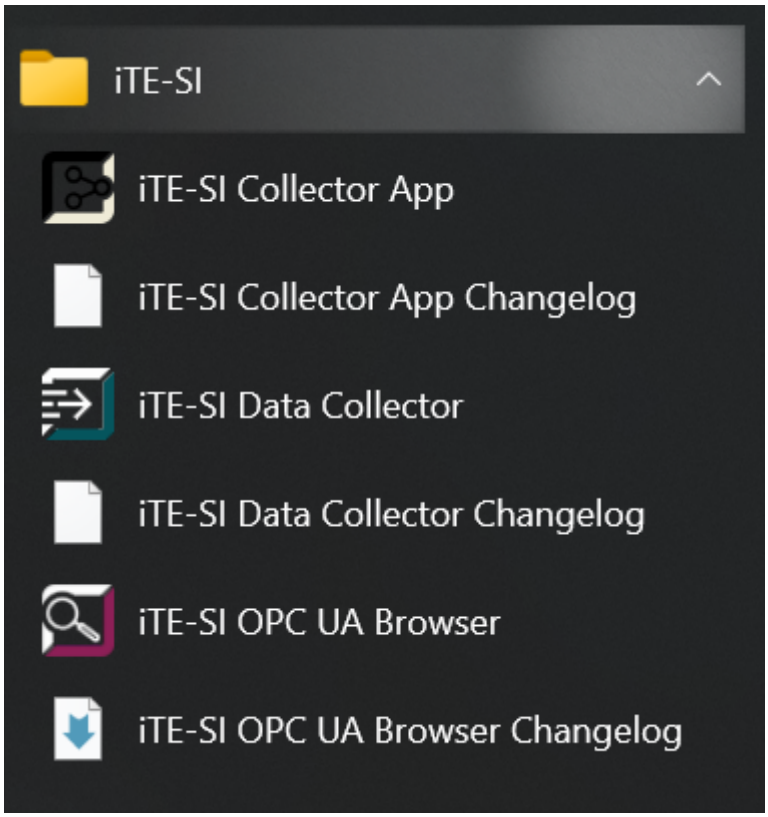
should be installed via the installer according to the Windows installation instructions.

As Demo Output the InfluxDB can be installed.

As demo input the OPC UA Sample Server from Unified Automation is recommended.

2. Starting the components

The components can be started via the Windews start menu:



After all three components have been started, a Windows tray icon appears in the taskbar. Here the log can be displayed and the application can be closed.

3. Start Collector App

After starting, the app can be accessed in the browser at http://localhost:4000/collector-app.

Default login:

username	password
admin	admin

An introduction to the Collector App can be found here.

Translated with www.DeepL.com/Translator (free version)

2. Licensing

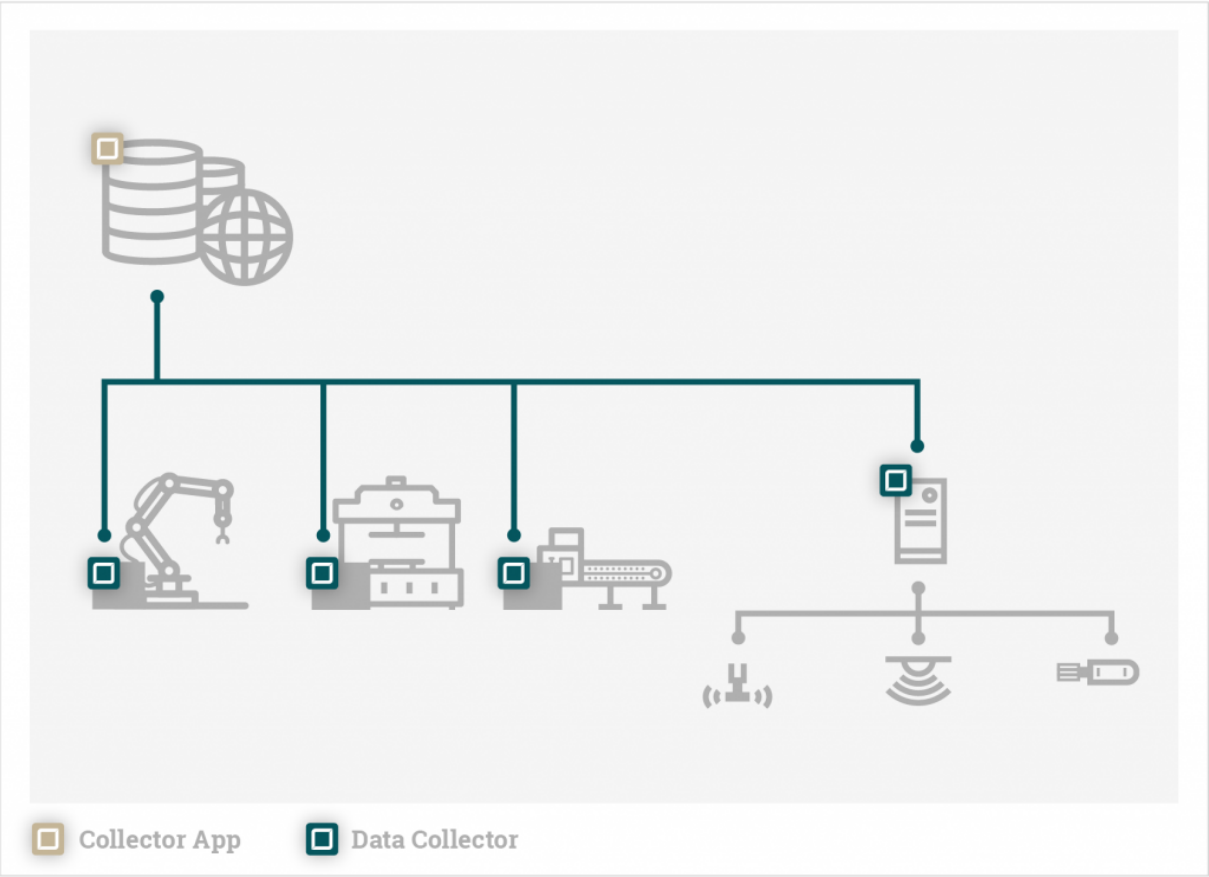
Each connection of a Collector must be licensed. A Collector writing data from an OPC UA server to an Influx database requires a single license for each connection.

Collectors can be licenses in two ways: purchase and subscription version. Details on the versions can be found under Preismodelle. Using the purchase mode, you have to buy connection-type specific licenses which can be used infinitely. Using the subscription version, a specific number of blocks can be purchased. These blocks can be used to activate any type of connection in a flexible way. When the subscription expires, the blocks cannot be used anymore.

Each collector can be operated in purchase or subscription mode. You also can use some collectors in purchase mode and some in subscription mode in the same local network.

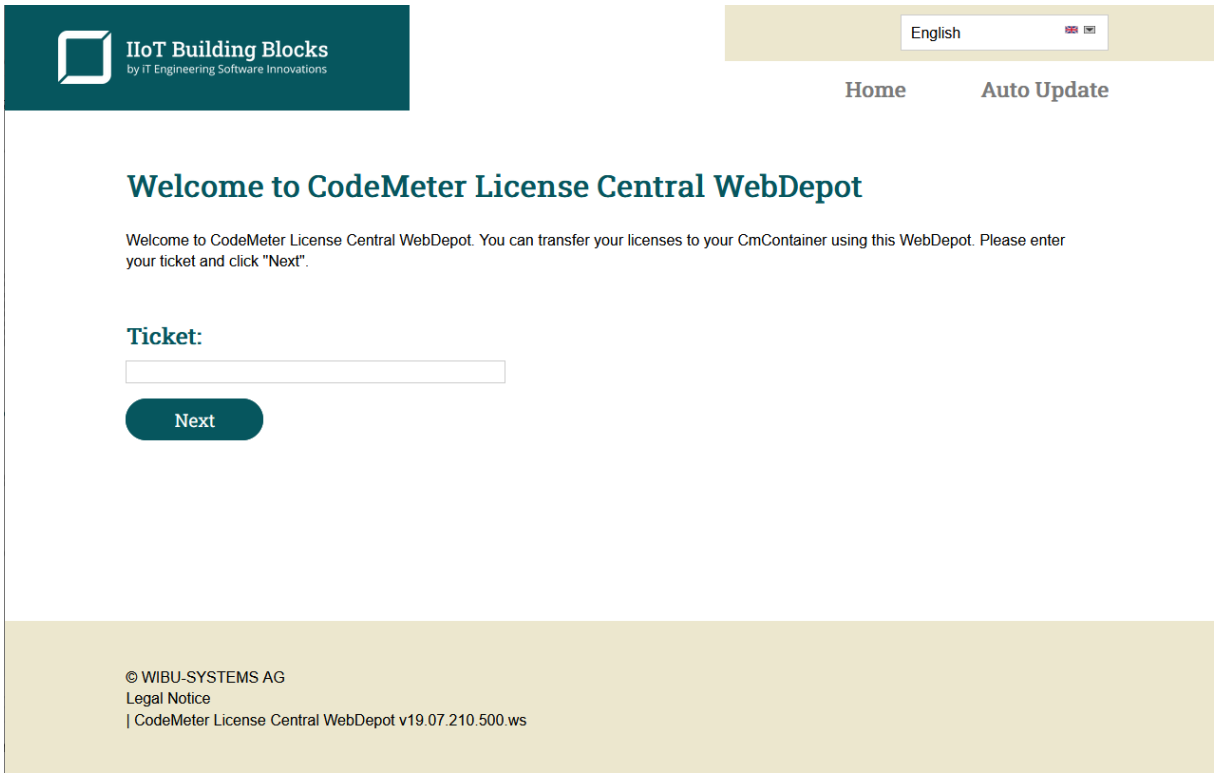
2.1 Installation of Licenses

Both types of licenses are installed as "floating network licenses" in your local network. The collector searches the licenses within the local network. This allows to install licenses on a central location within the network.

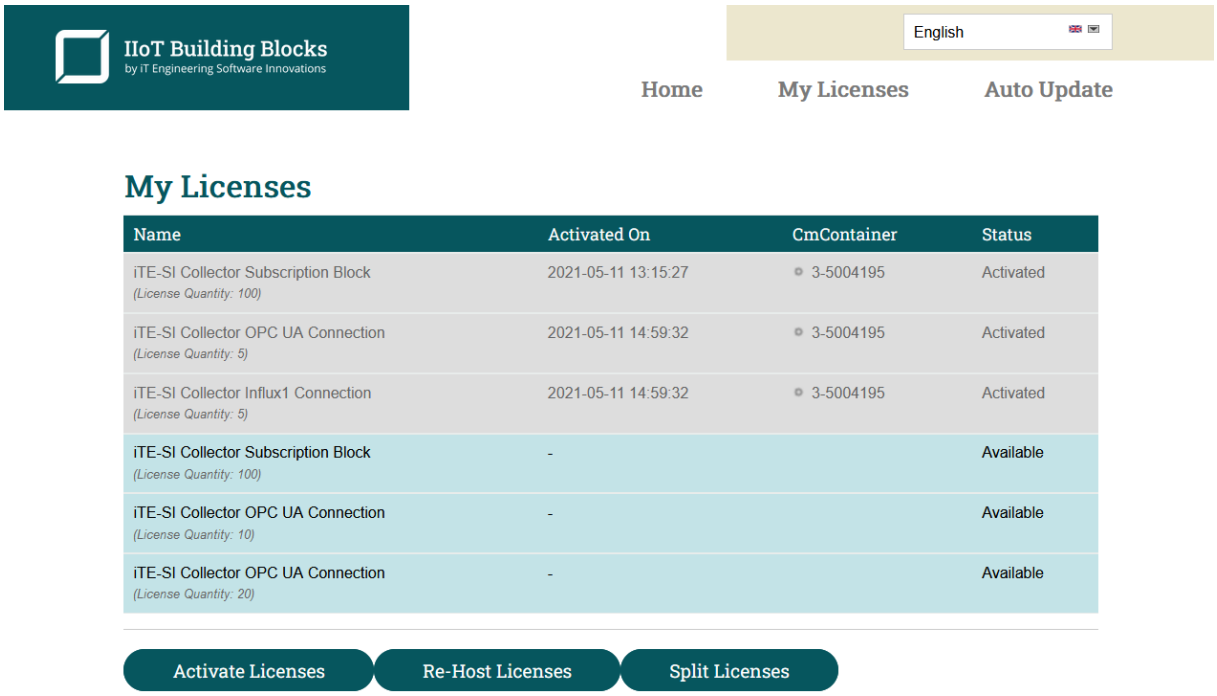


In the deployment shown above, you can either install all licenses on the same server as the Collector App or you can install the specific licenses on each device, where the collector is installed.

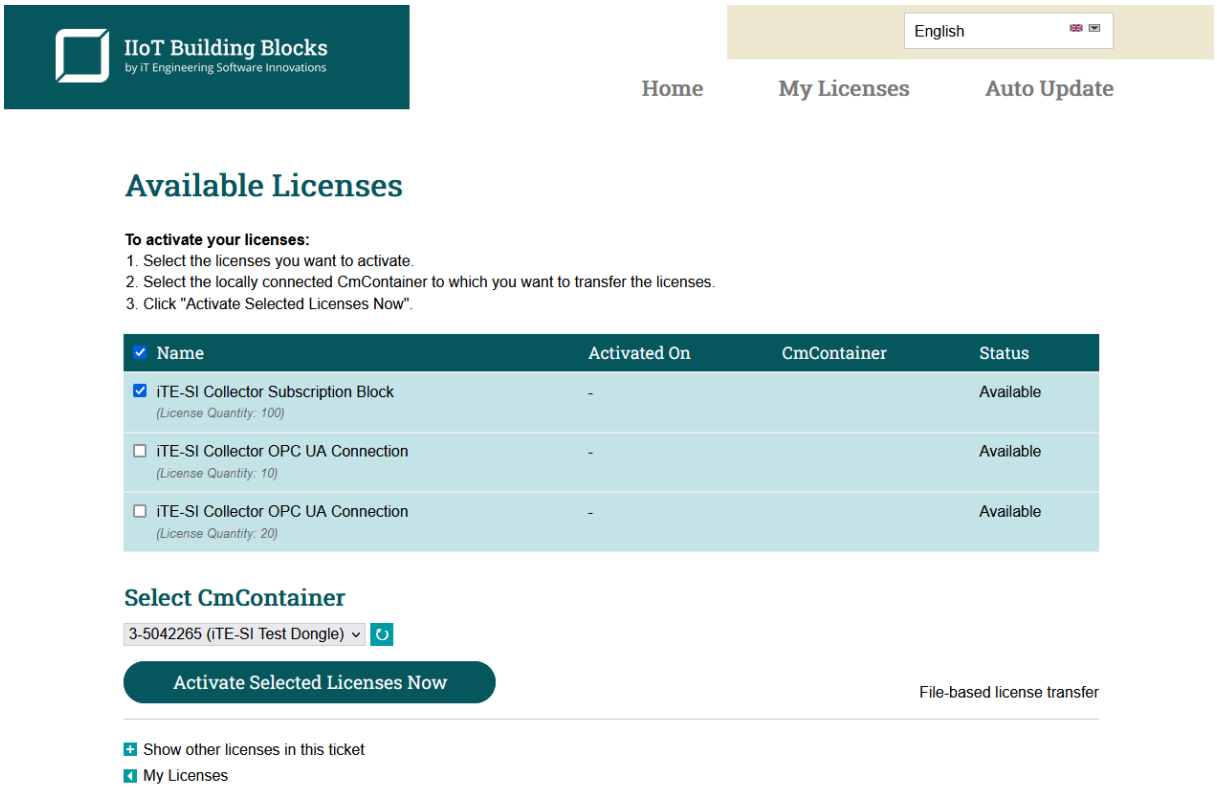
You can download your licenses through the iTE-SI license portal. You should open the website from the PC where you want to install the licenses to.



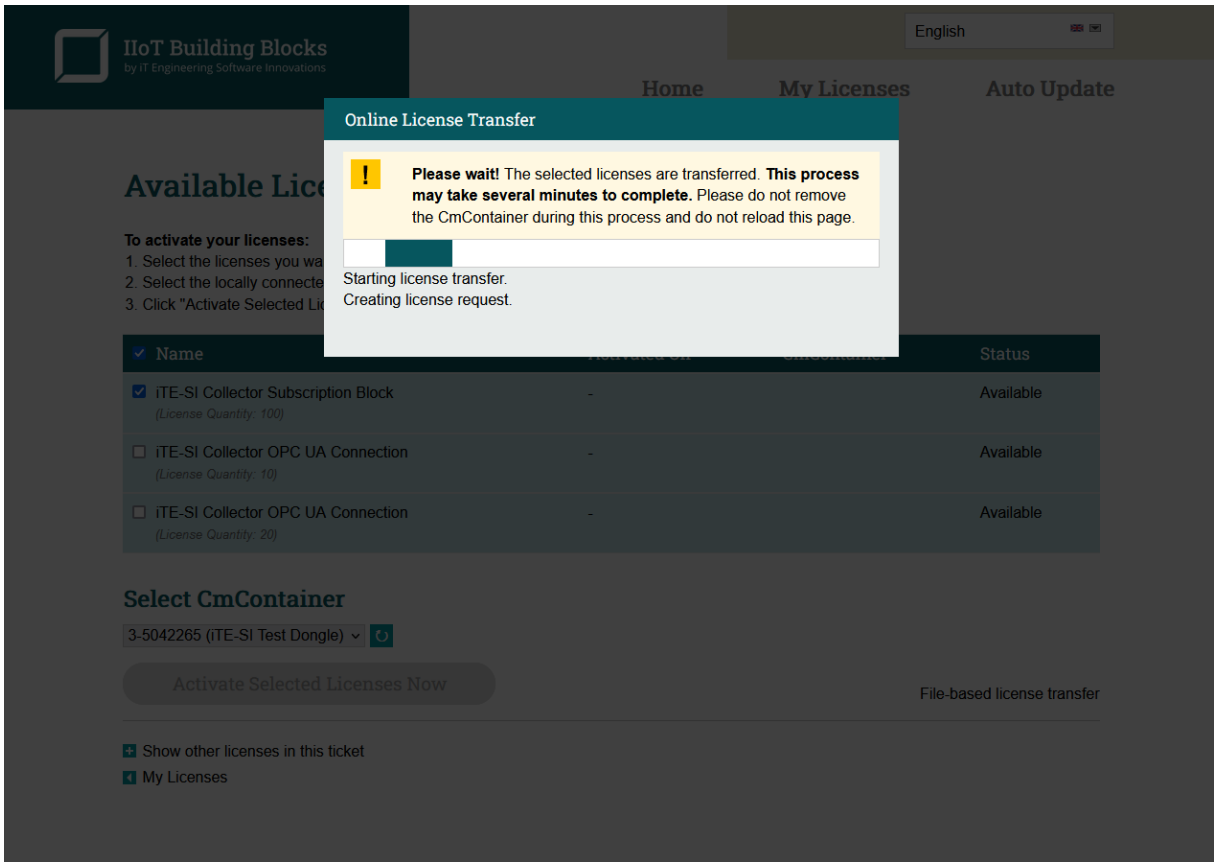
The ticket number (provided with your order) must be entered to go the license overview page.



Licenses can be activated, moved to different hosts or can be split into different sizes.



You can install multiple licenses in one go to one PC.

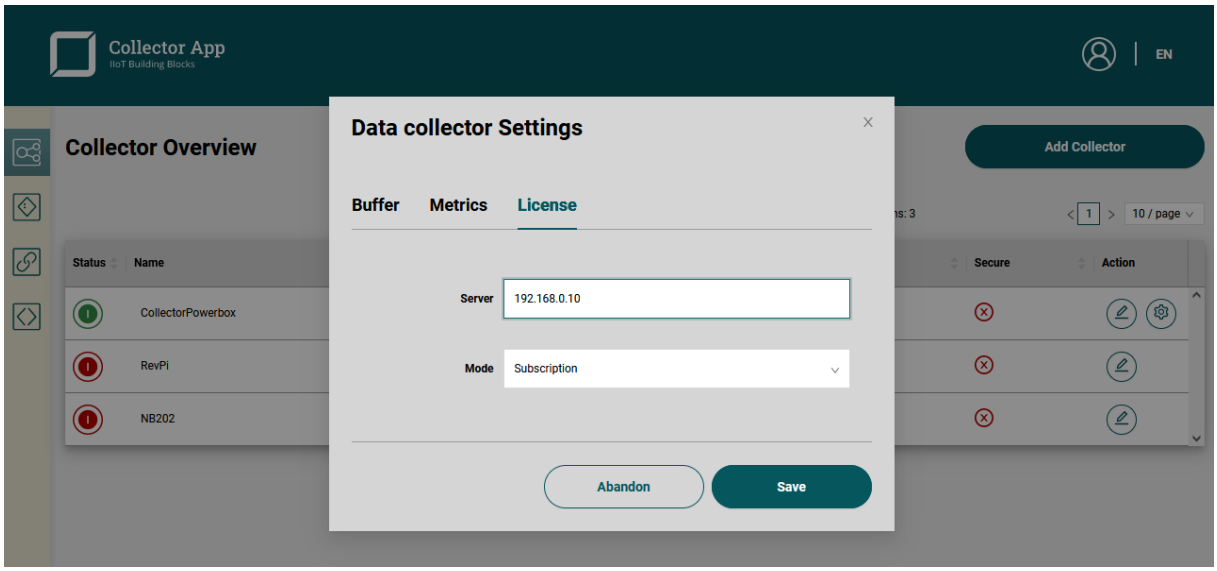


2.2 Operating Without Licenses

A collector will operate each connection for a limited time frame of 6 hours. If no license can be found after this time, the connection will be shut down.

2.3 License Configuration in the Collector App

You can configure the license mode and the license server for each Collector individually in the Collector App.



Normally, the license server must not be configured manually. The Collector does find its server on its own. Under specific conditions, e.g. virtualizion, VPN, or disabled UDP broadcast, the server might not be found and the Hostname or IP address of the server must be configured.

The Collector can be operated in purchase, subscription or test mode. In the test mode, the Collector will not fetch licenses from the server, even if licenses are available. Connection will time-out in 6 hours. In the other modes, the Collector will search for the configured licenses.

3. iTE Collector App

3.1 General

With its user interface, the Collector App enables easy setup of the system, configuration of data inputs and outputs. The data of different production machines and plants can thus be clearly combined.

3.2 Installation

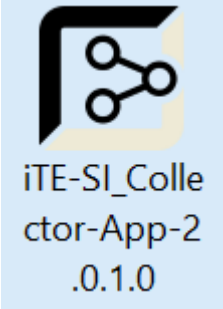
3.2.1 Windows

Download Installer

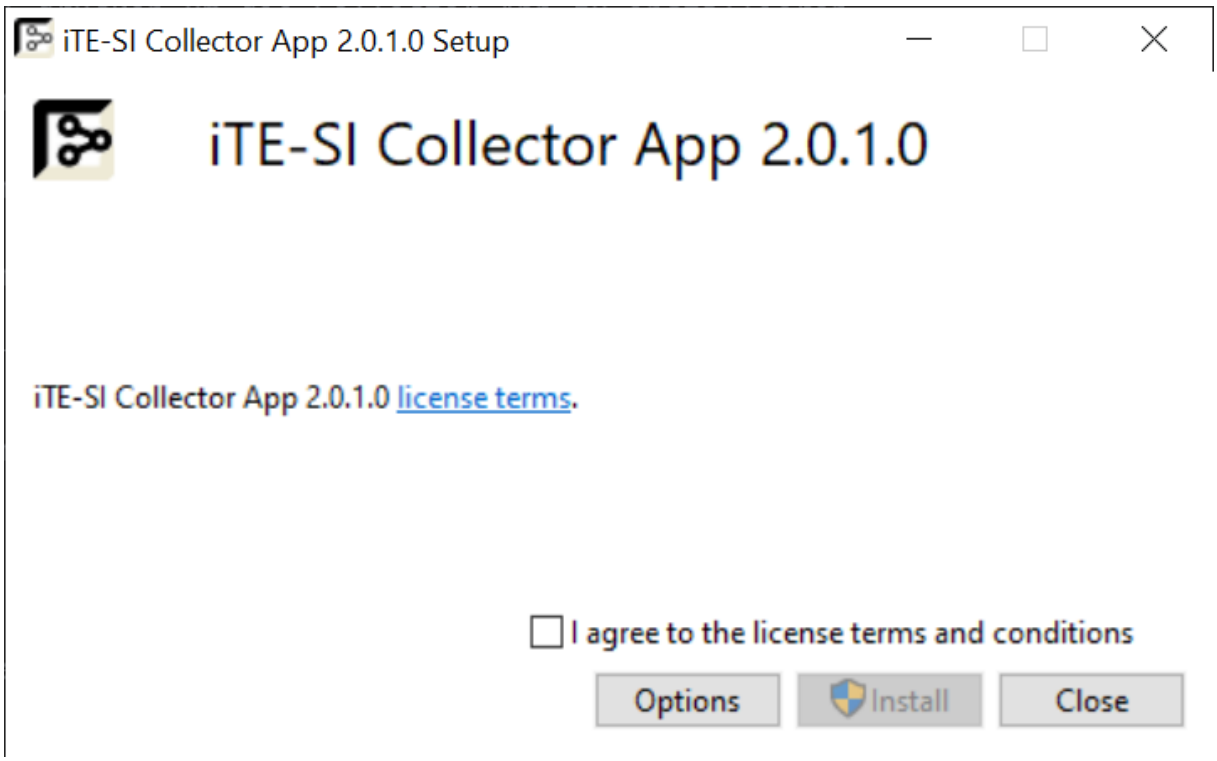
The Windows installer can be downloaded here: IIoT Building Blocks Downloads

Install

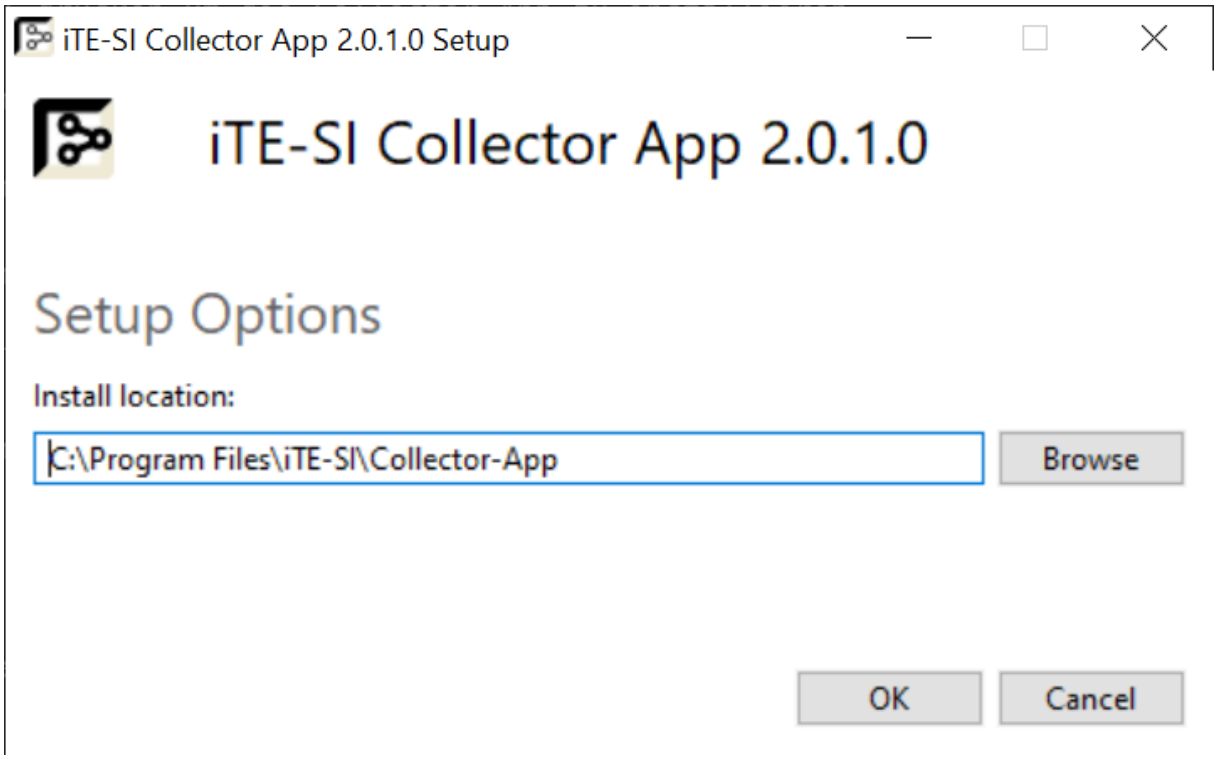
1. Run the installer.



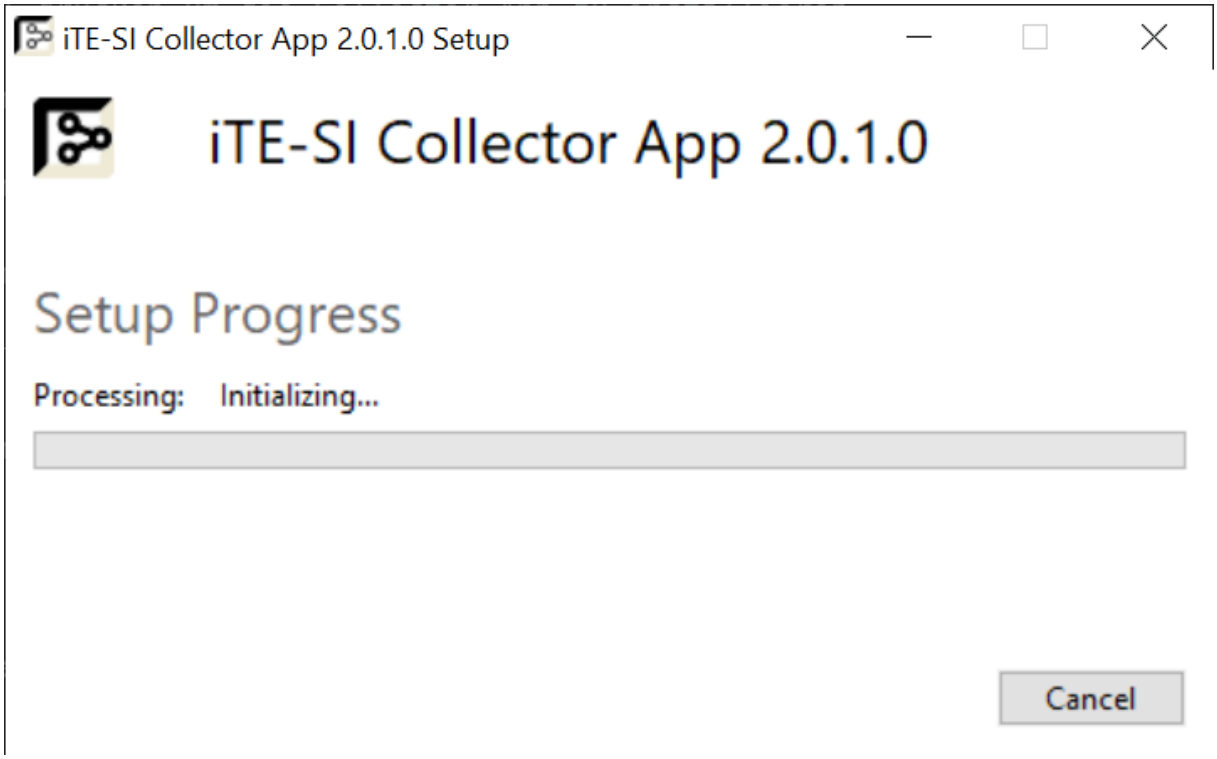
2. Read the end user license agreement, accept it if necessary and press `Install`. Otherwise press Cancel to abort the installation.



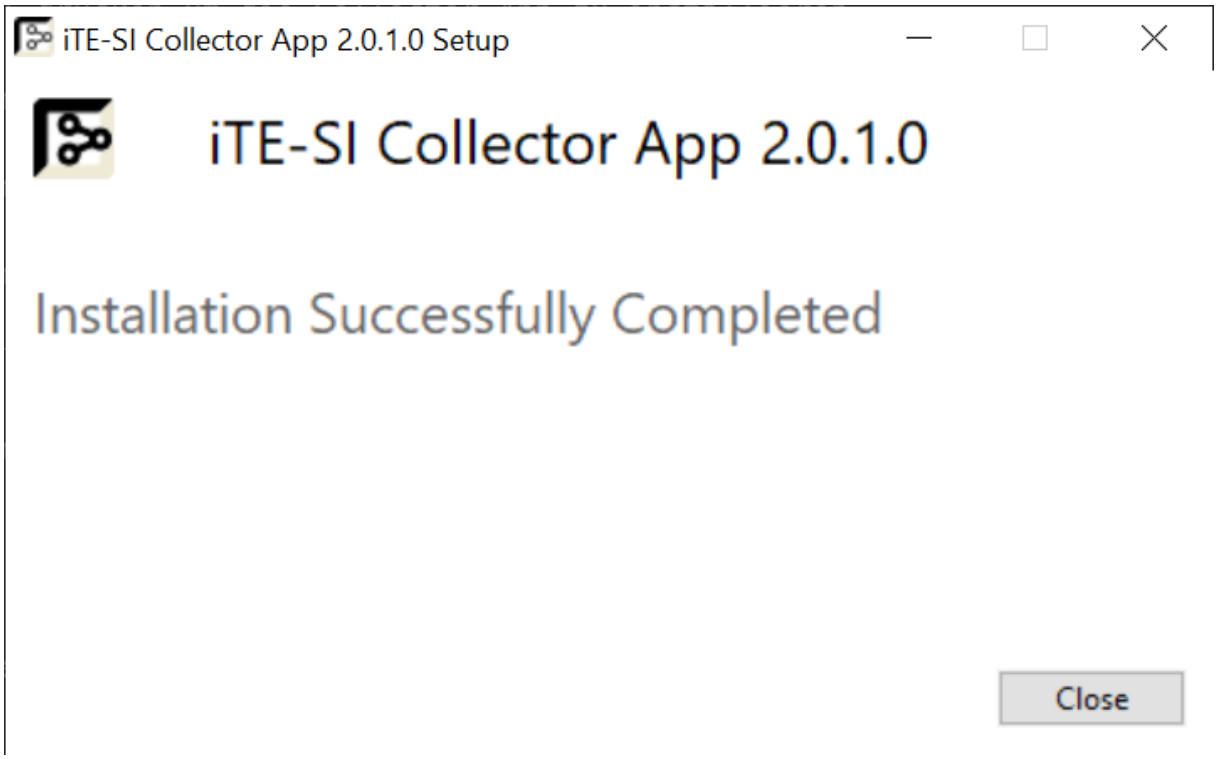
Optional: Press Options to select the location.



3. Wait a short time.



4. The installation was completed successfully. Press Close.



5. The Collector App can now be started from the Start menu.

3.2.2 Docker

The Docker image can be downloaded here: IIoT Building Blocks Downloads

1. Loading the image

Info

x.x.x.x must be replaced by the downloaded version!

Info

If rancher-desktop is used with containerd, docker must be replaced by nerdctl.

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

2. Start container:

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

The collector app can be configured using environment variables. For example:

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

For more on configuration, see: Collector App Configuration.

To save the Collector App settings persistently, the folder with the SQLite database in the container must be mapped to a folder in the host system.

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

3.3 Configuration

3.3.1 Parameters

The following parameters can be configured for the Collector App:

Parameter	Description	Default
host	The IP address of the interface through which the web server of the app should be made available. If `localhost` is selected, the app will only be accessible from the server on which it was installed. By default, the app binds to all interfaces and is thus reachable from outside.	0.0.0.0
port	The port on which the app's web server should run	4000
tokenSecret	The key used to sign the user cookie. The minimum length is 32 characters.	-
encryptionKey	The key used to encrypt Collector Api tokens and store user passwords hashed in the database. The minimum length is 32 characters.	-
corsOrigin	The IP addresses or domains under which the Collector App can be reached over the network. This is for security purposes. Requests from web pages with a different IP or domain will be blocked. Multiple IP addresses or daomains are separated by a `;`	http://localhost:4000
apiURL	The url to the Collector App Graphql server. This configuration is for the the front end. The backend always starts the API under the route `/api/graphql`	http://localhost:4000/api/graphql
wsUrl	The url to the collector app web socket. This configuration is for the front end. The backend always starts the web socket under the route `/api/subscriptions`	ws://localhost:4000/api/subscriptions
database	Different SQL databases can be configured as described here	typeorm SQLite config

The Collector App needs a database to store users and settings. Different SQL database management systems can be configured for this purpose. By default, the Collector App uses a SQLite database in the form of a file. So no database server is needed.

There are several ways to configure the Collector App which are explained below.

3.3.2 Secure web communication (TLS)

The Web/API server of the Collector App runs unencrypted (http). It is also not possible to configure encryption (https). The reason for this is that otherwise the configuration effort would be much more complicated, especially due to certificate management. In a trusted, internal network, unencrypted operation can be tolerated under certain circumstances.

Warning

As soon as the Collector App is operated in a network that could be accessed by untrusted persons or even on the Internet, encryption is very important, otherwise transmitted passwords can be read!

To run the Collector App via TLS (https), the web server nginx is recommended. This can act as a so-called reverse proxy. You can configure the Collector App to run only on localhost or a secure VPN. Nginx is installed on the same server or on one in the VPN and connects to the Collector App via the unencrypted http protocol. Outward to the insecure network, nginx provides a secure https server. Thus nginx is able to forward the requests coming over the secure protocol to the collector.

Example Nginx configuration as a reverse proxy for the Collector app:

```
events {}
http {

  upstream collector_app {
     server {http_collector_url};
  }

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

# Requests to http are redirected to https
  server {
     listen 0.0.0.0:80;
     server_name {server_name};
     server_tokens off;
     return 301 https://$host$request_uri;
  }

# The secure 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/;
    }
  }
}
```

The value for the collector app url {http_collector_url} and the nginx host name {server_name} must be configured accordingly. You also need a signed certificate {path_to_cert} and key {path_to_key}. If the Collector App is operated over the Internet, the certificate service Let's Encrypt can be used.

A self-signed certificate can also be created for testing purposes:

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

3.3.3 JSON file

The default configuration file is located

- under Windows in the App Data directory: `%appdata%/iTE-SI/Collector-App/config/default.json`
- and on Docker/Linux `/etc/collector-app/default.json`.

By default, the file has the following contents:

```
{
  "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"
  }
}
```

In the `database` section different SQL databases can be configured. For this we refer to the documentation of typeorm: https://github.com/typeorm/typeorm/blob/0.2.45/docs/connection-options.md

By default a sqlite3 database file is used, so that the Collector App can run standalone without other dependencies. This database is usually sufficient for the performance of the app.

Custom Configuration

If you want to customize the default settings, you should not do this to the `default.json` file, but create a copy `production.json` in the same directory. This file can now be customized as you like.

Info

It is recommended to change the two secrets `tokenSecret` and `encryptionKey` after the installation.

3.3.4 Environment variables

For the deployment via Docker container but also for setting the secret keys, environment variables are especially useful. If a variable is set, it is used in preference to the value in the configuration file.

General environment variables

- 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

Database Environment Variables

To configure the database, the typeorm environment variables are set as described here.

- 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

After the basic setup has been installed according to Basic Getting Started, the Collector App can be set up and the Collector can be configured.

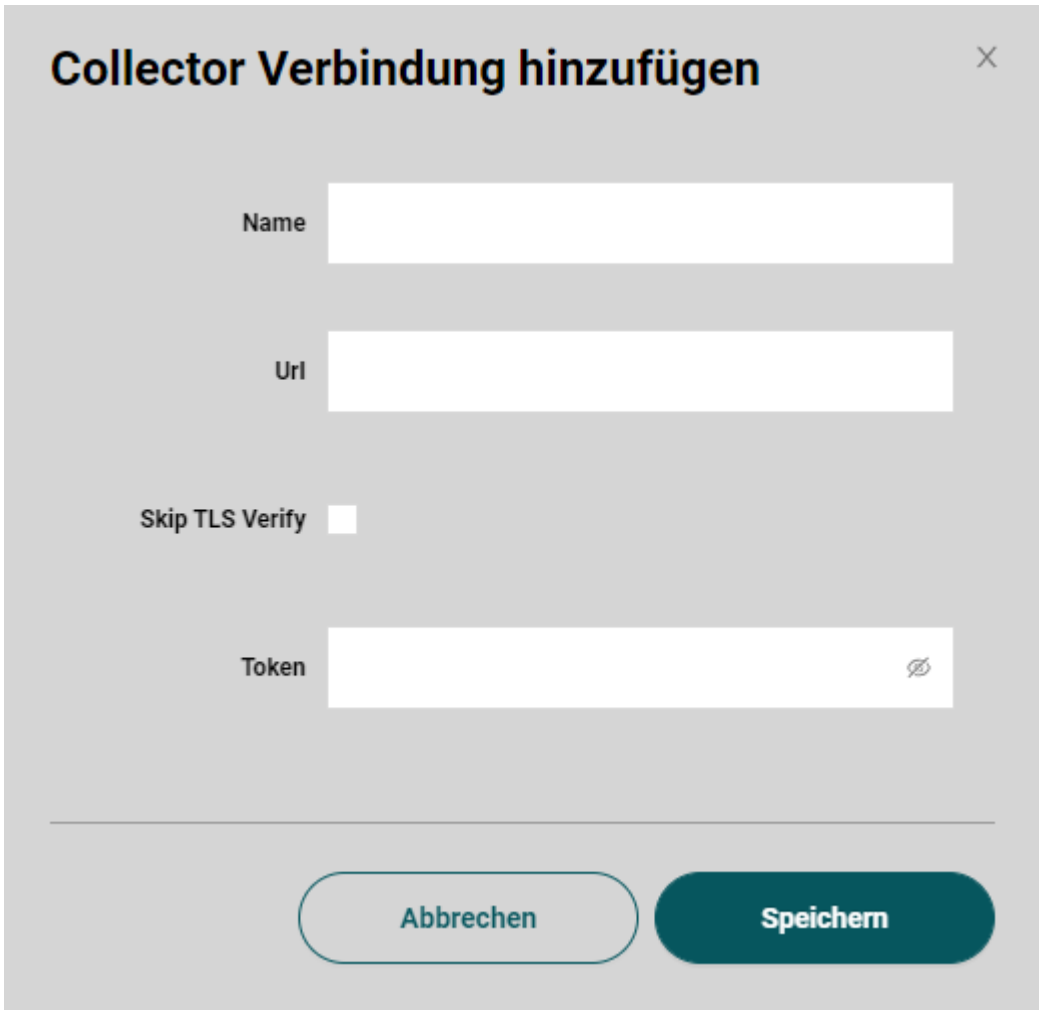
3.4.1 Include Collector(s).

In order for the Collector App to configure data collectors, it needs their network endpoint.

1. Go to the Data Collectors page via the navigation bar: http://localhost:4000/collector-app/de/collectors.

2. Click on the `Add Collector` button.

3. Any name can be assigned. The url starts with `http://` for an unencrypted connection and with `https://` for one encrypted with TLS, followed by the IP address/hostname and the port (default: `http://localhost:8001`). If an encrypted connection with self-signed certificates is used, TLS verification must be turned off. The Data Collector configuration interface can be secured with a token. By default, no token is configured.



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

4. Click on Save. The collector appears in the collector overview table.

Collector settings

1. By clicking on the gear icon in the action column of the overview table of a collector column, the collector settings open.

2. The buffer setting allows to store data locally on the computer on which the collector is running in case of a network interruption. A file path and the maximum buffer size must be specified. The local error memory stores information when data could not be transferred correctly.

3. By the Metrics setting the collector writes cyclic information about its own state into an Influx database.

4. If the license server is not found automatically by the broadcast search, its IP or hostname can be configured. In addition, the license mode in which the collector is to be operated is specified here.

3.4.2 Include OPC UA Browser

To be able to search OPC UA servers, an OPC UA browser is required. The Collector App also needs the network endpoint. The configuration is done equivalent to the Collector on the OPC UA Browser page.

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

3.4.3 Create connections

The first thing to do is to configure the connection information to the data sources and sinks. This can be done on the connections page.

OPC UA Server connection

The OPC UA Server connection will be created as data source.

1. Click on the 'Create connection' button. A dialog with 3 steps opens.

2. A connection can be created on several collectors at the same time. For example if you want them all to write to a central database. 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.

The following values are configured for the quick start configuration:

Translated with www.DeepL.com/Translator (free version)

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

The import function allows to load previously exported connections in form of a json file.

4. Finally the OPC UA connection type has to be selected. Then OPC UA specific security settings can be made. In the default case no security mechanisms are applied.

5. After clicking on Save the connection appears in the connection overview table.

Influx database connection

The connection to the Influx database serves as a data sink.

**The Create dialog is run a second time. This time with the following values for the demo setup:

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

The 3rd step is to select the Influx connection and configure the database name. If the database does not exist yet, it will be created automatically. The remaining parameters are set by default.

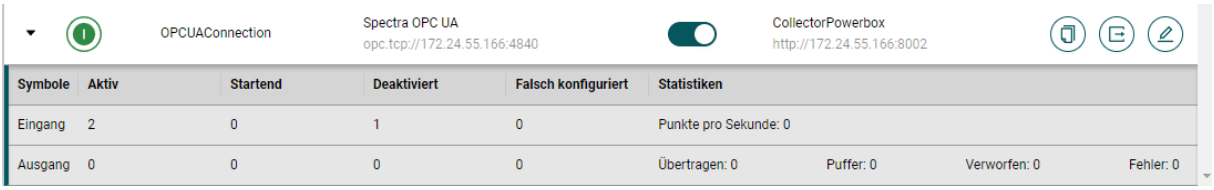
**After saving, the Influx database connection will also appear in the table.

3.4.4 Create symbols

Each link has subordinate symbols. There are output and input symbols, which are linked to each other. An input symbol can be linked to several output symbols. The following steps explain how to create an OPC UA Subscription symbol and link it to an Influx Measurement.

Create OPC UA Subscription

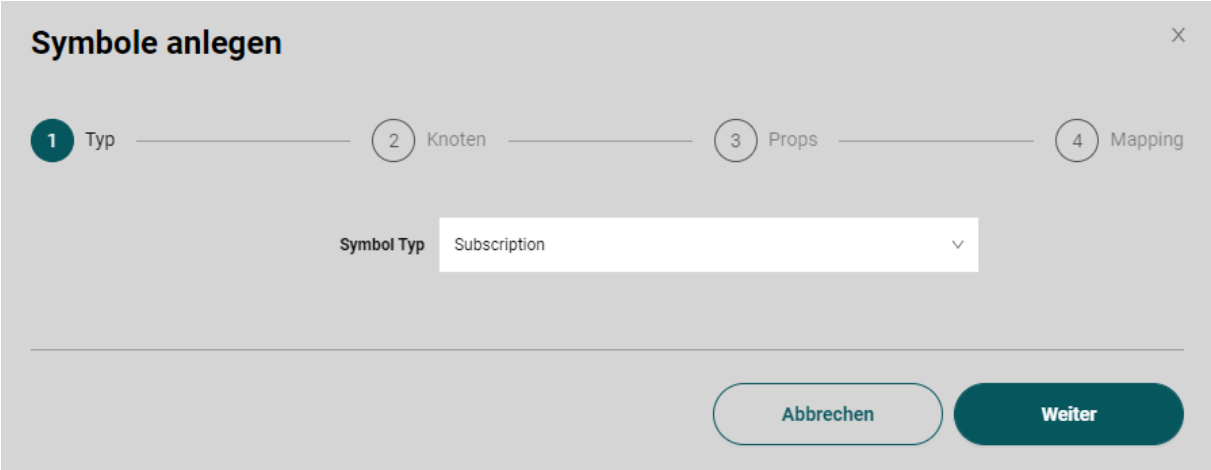
1. There are basically two different ways to get to the symbol overview: Via the navigation bar or if you click on the OPC UA connection in the connection table, the row will be expanded. A summary of the input and output symbols appears. By clicking on the input line you are automatically forwarded to the symbol page, so that the connection is directly pre-filtered.



2. On the symbol page the OPC UA Sample Server connection must be selected in the parent filter. To select from all connections click on the magnifying glass icon.

3. After a connection has been selected, the 'Create Symbol' button appears and can be clicked.

4. A dialog with several steps opens. As type the `Subscription` is selected.



5. In the next step the OPC UA Server can be searched with the help of the browser. By clicking on the checkbox OPC UA variables can be selected. At least one variable must be selected. For each selected variable an icon is created.

6. Afterwards optionally different properties can be configured.

7. In the last step 'Automapping' different connections can be selected, for each of which output symbols are created and automatically linked to the OPC UA Subscription symbols. Here the InfluxDB is selected.

8. After clicking Save, OPC UA Subscription and Influx Measurement symbols are created and linked automatically. If the symbols are switched on, the Data Collector starts working. The symbols appear in the table. By clicking on a symbol row the linked symbols are displayed.

Adjust Influx Measurement

1. Change the parent filter to the InfluxDB connection.

2. Expand an Influx Measurement in the table.

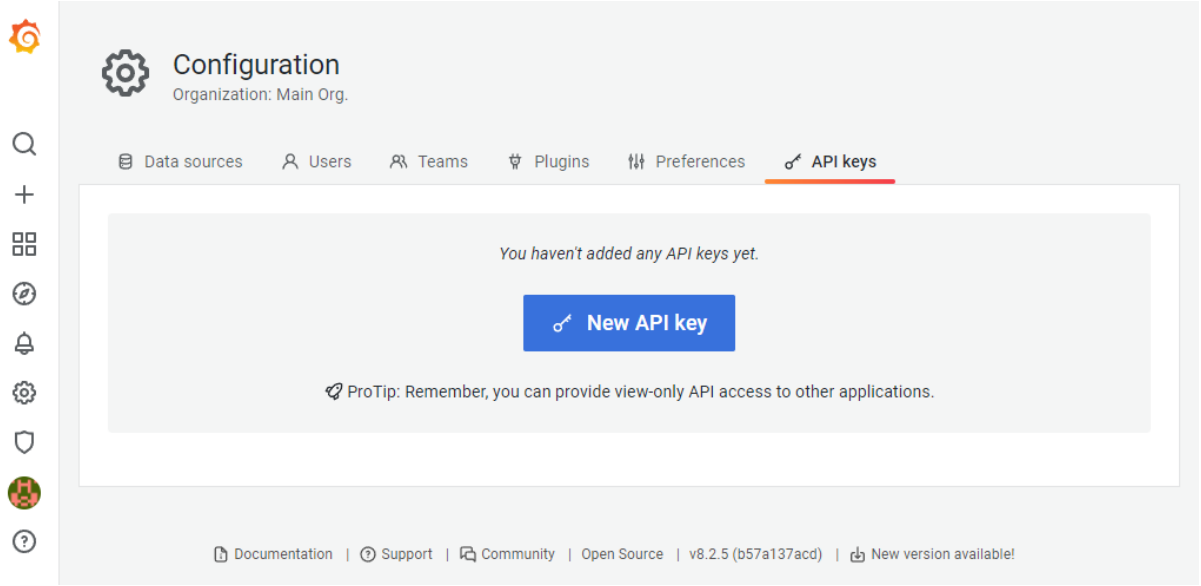
3. The structure of the measurement is displayed in the expanded area. Fields and tags can be configured here (Influx Doku). By default, the variable path in the OPC UA Tree is created as Tag and the value of the linked symbol as Value Field. The structure can be further customized here.

3.4.5 Grafana integration to visualization

Grafana is a web-based tool to visualize time series data. In the Collector App, a connection to Grafana can be created so that the data collected by the Collector can be displayed directly.

Create API Key

An API key can be created in the Grafana settings. This requires the admin role. More info in the Grafana doku.



Enter API Key

1. First select the settings page via the navigation bar.

2. In the 'Explore' module the Grafana integration is configured. Here the URL and the previously created API Key is entered. After clicking on Save you will be asked if Grafana Datasources should be created automatically based on the connections in the Collector App.

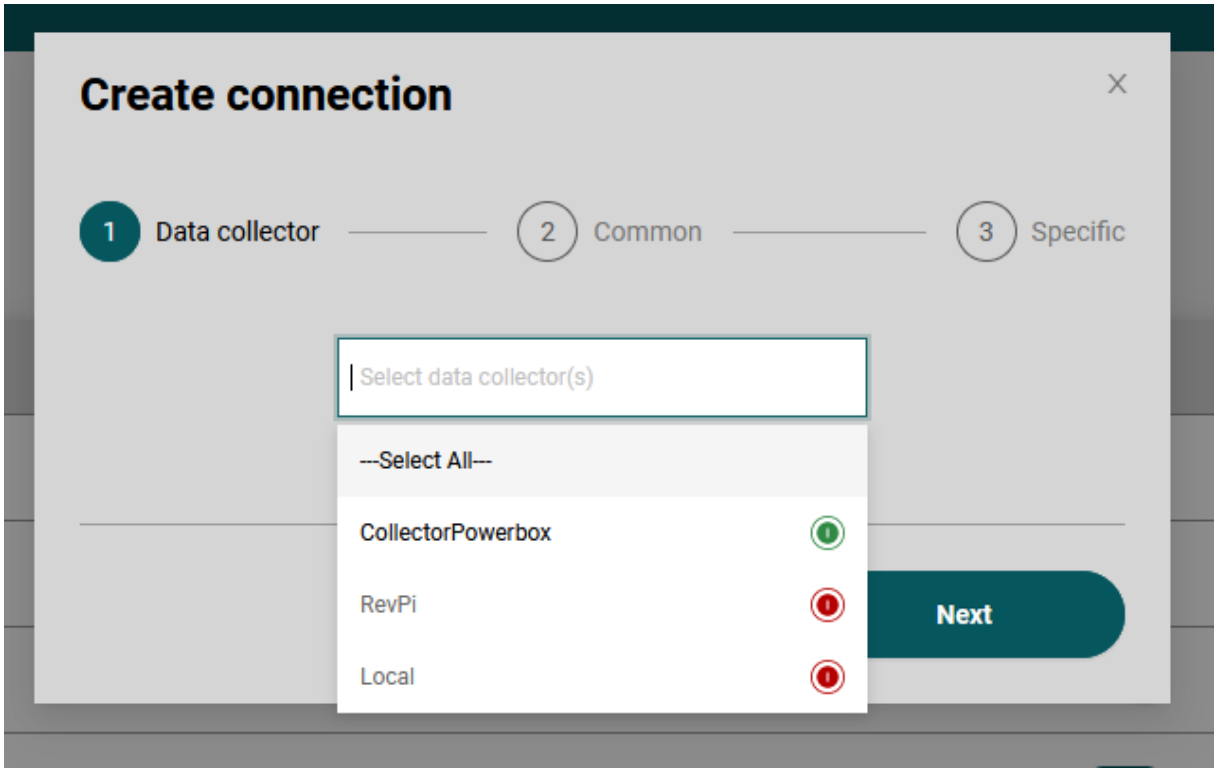
3. Now you can click on the link button in the symbol table for Influx Measurements. Then a new browser tab opens with a Grafana panel in which the measurement is displayed.

3.5 Configuration of Connections

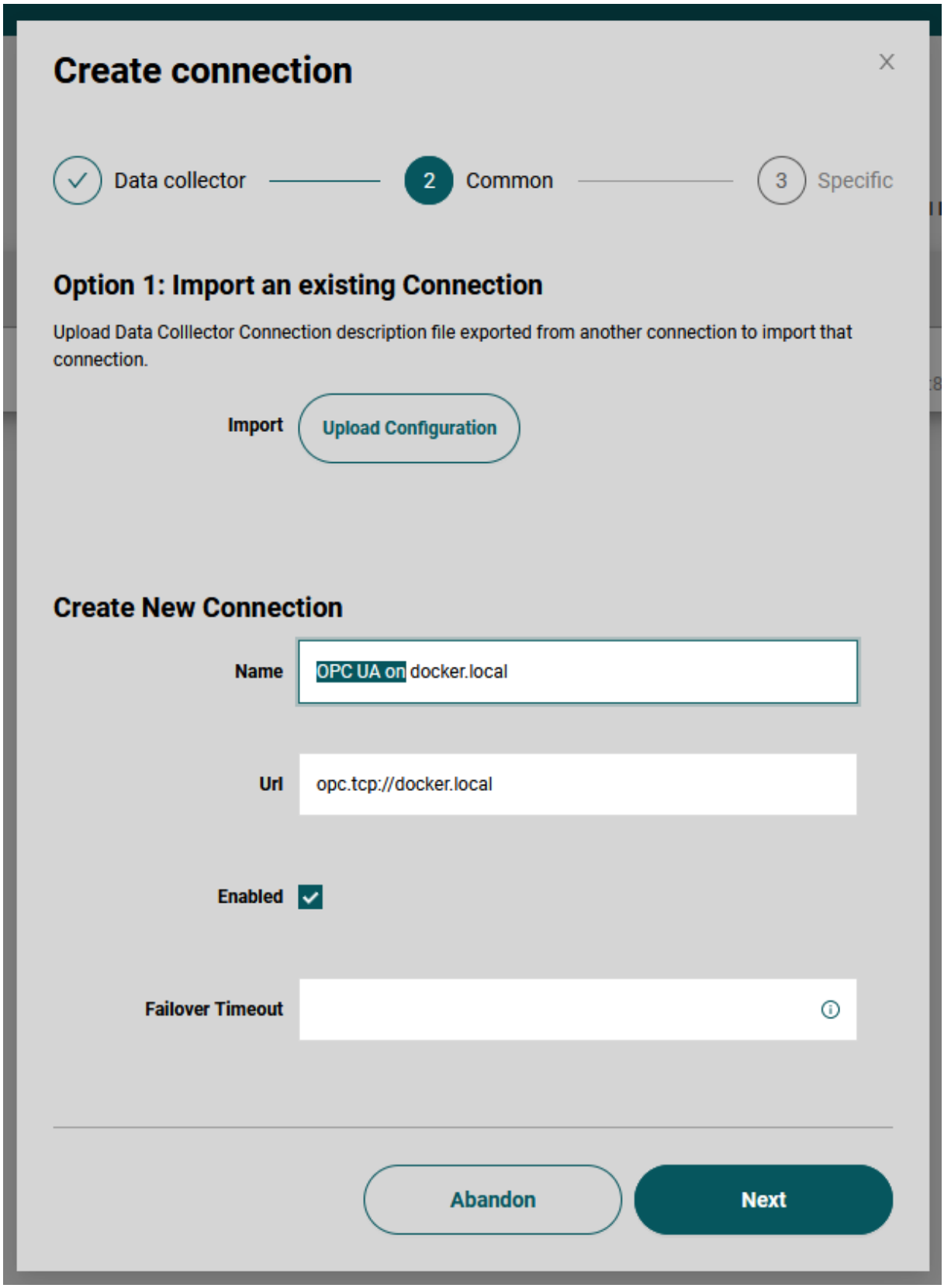
Connections can be added using the connections tab in the navigation bar on the left side. Press the "Create connection" button to start the wizard.

3.5.1 Create Connection Wizard

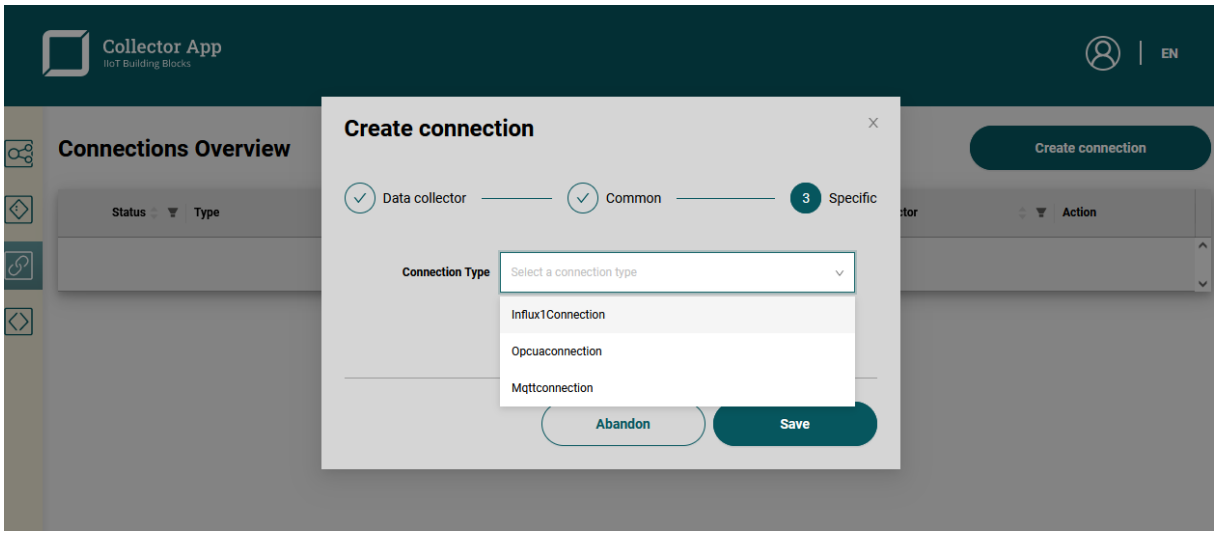
On the first page select the collectors the connection should be created on.



On the second page, enter the name and URL for the connection, e.g. opc.tcp://hostname, or mqtt://hostname.

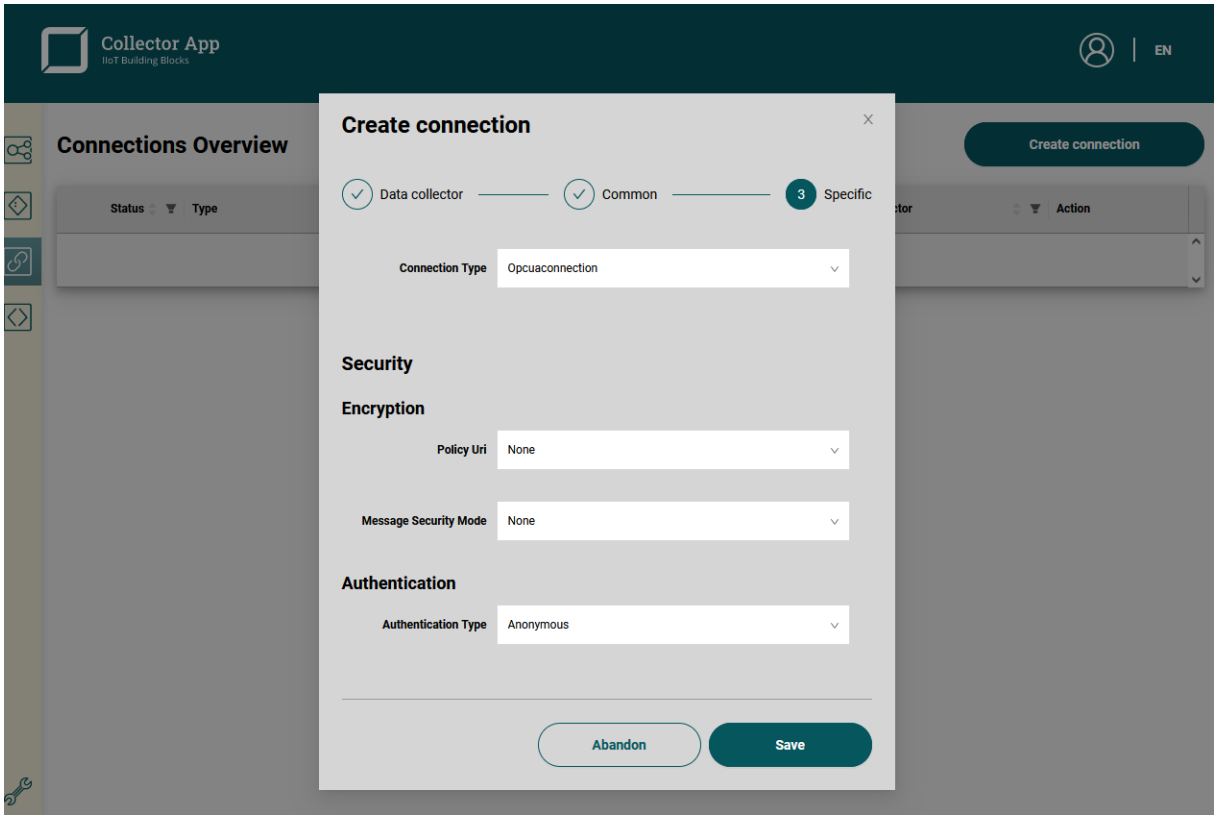


On the third page, the connection type will be selected and connection specific settings will appear.



OPC UA Connection

The OPC UA specific settings are used to configure the secure connection.

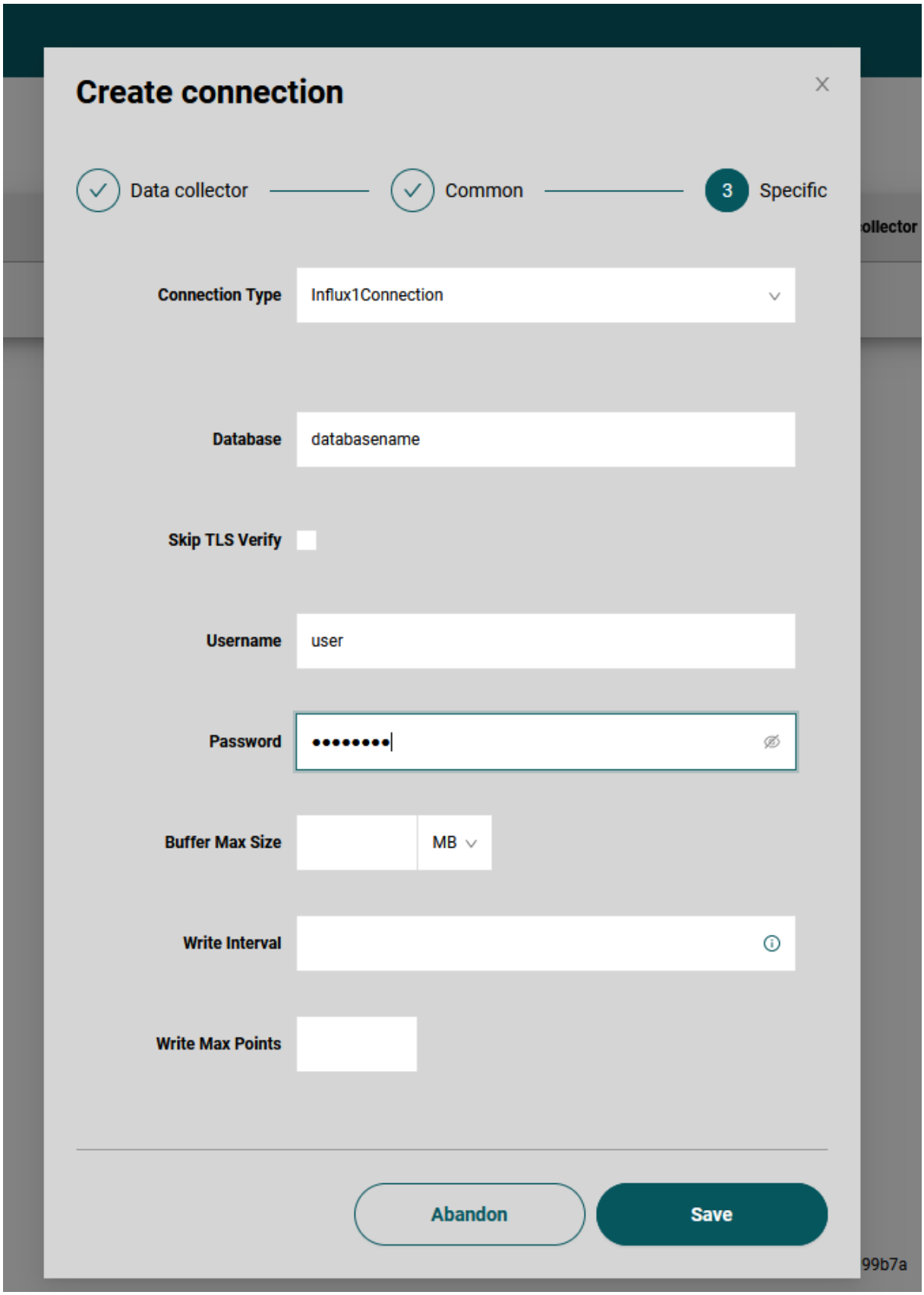


The default settings are to connect without encryption and password.

Select appropriate Policy Uri, Message Security Mode and Authentication, e.g. username and password, when the server requires additional setup.

Influx Connection

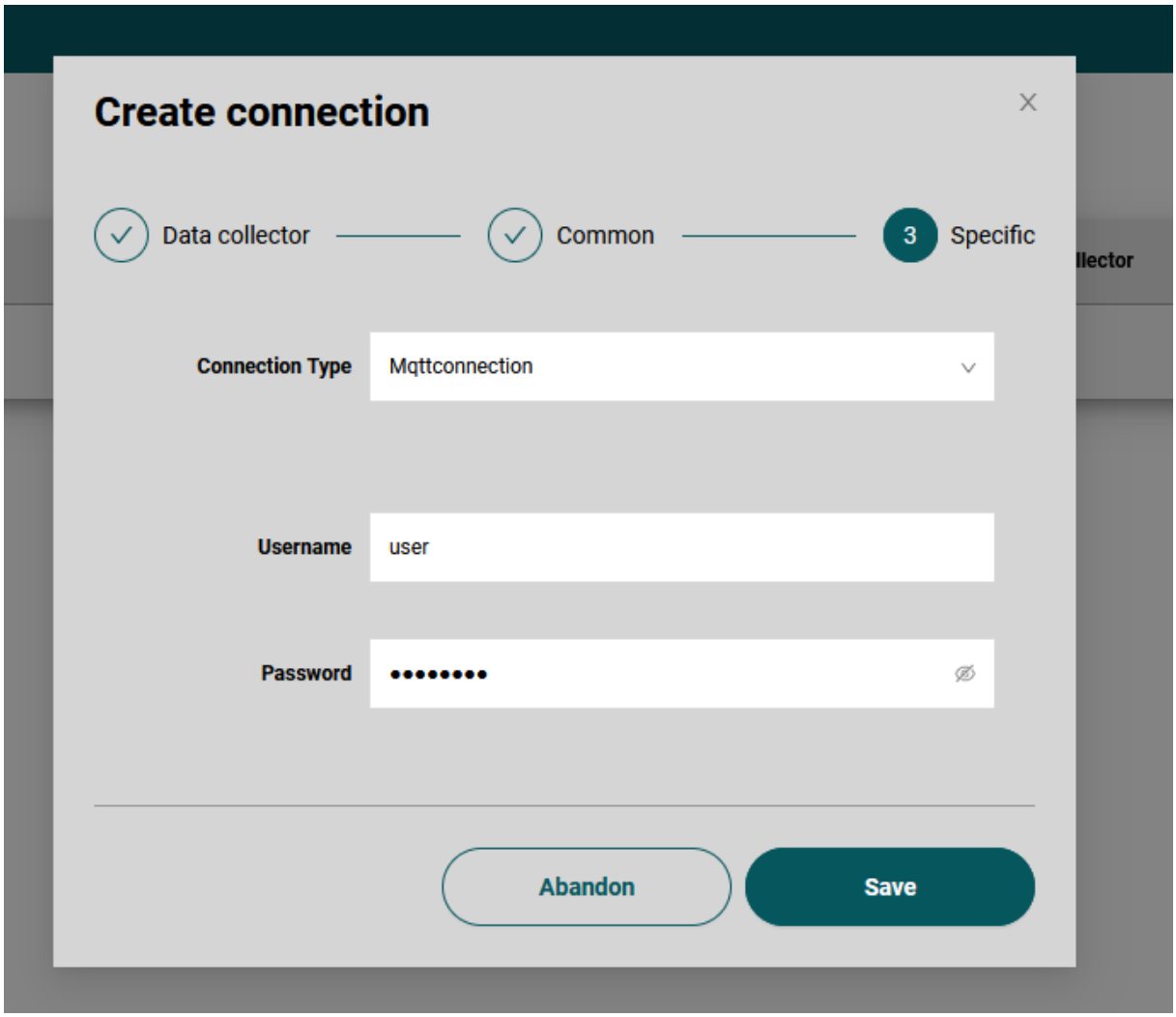
It is necessary to enter a database name for the influx connection. For each database on a influx, a separate Data Collector connection must be created.



The remaining options can be left blank, so they are filled with its default values. - **Skip TLS Verify** option is required if the Influx DB uses a self-signed certificate and a https endpoint. - **Username** and **Password** should be filled in, if the InfluxDB uses authorization. - **Buffer Max Size**, **Write Interval** and **Write Max Points** can be adjusted to optimize collector specific internals. Unless needed otherwise, keep the options at their default, i.e. empty.

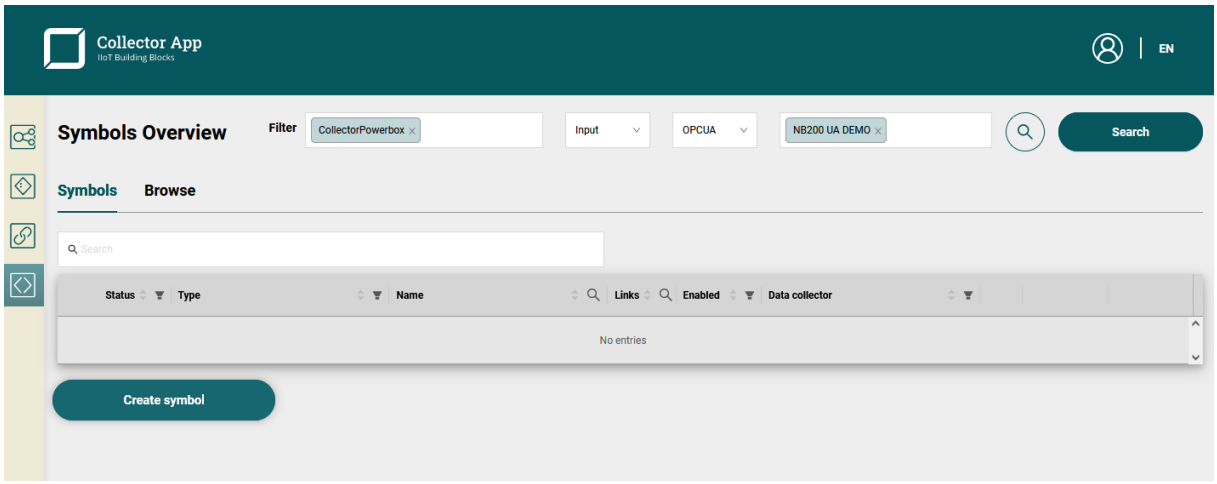
MQTT Connection

For MQTT it is possible to specify a user/password configuration, to authenticate against the MQTT server.



3.6 Configuration of Symbols

To create symbols it is necessary to select a single connection in the filter bar. In the example it is the NB200 UA Demo connection which is a OPC UA connection on the CollectorPowerbox Data Collector.

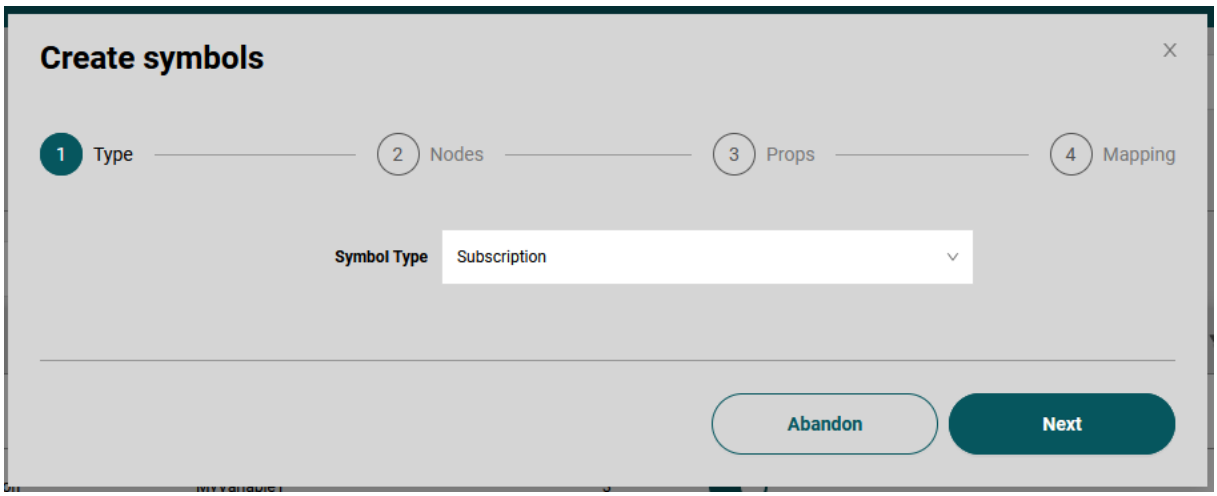


Pressing the "Create Symbol" opens the (connection specific) wizard.

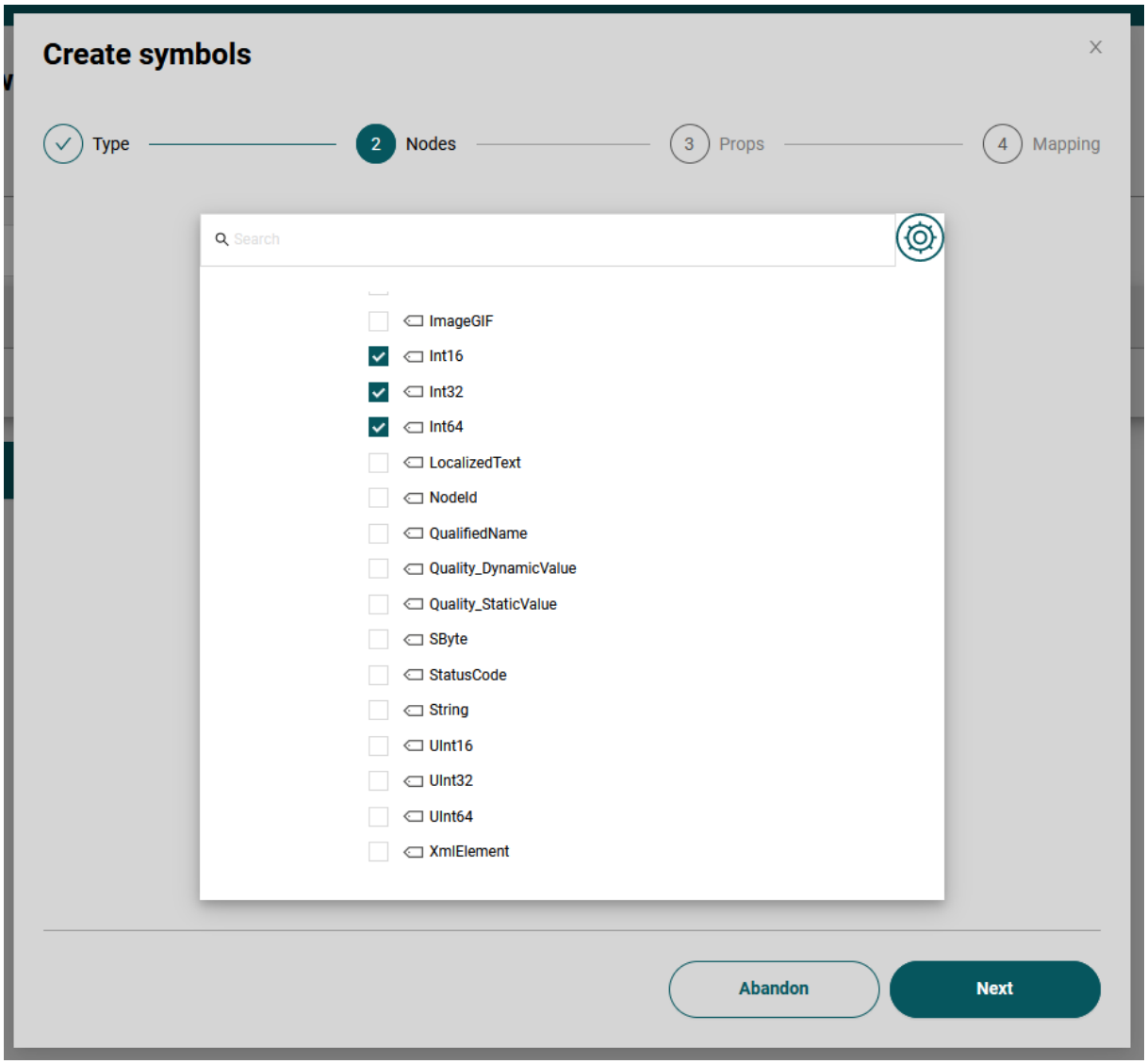
3.6.1 Create OPC UA Subscription Symbols

A OPC UA Subscriptions samples a single OPC UA nodes and collects the data. The data can then be written by an output symbol, e.g. an Influx measurement.

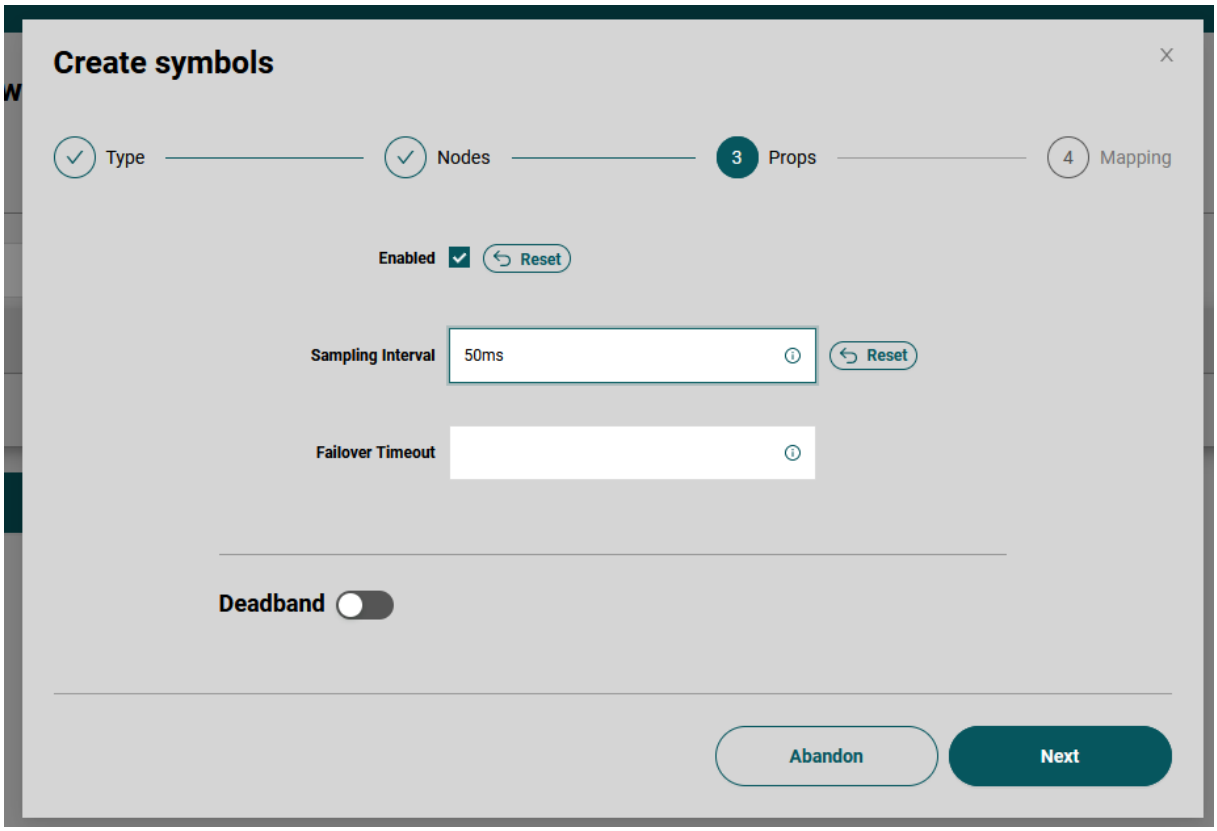
Select the symbol type Subscription.



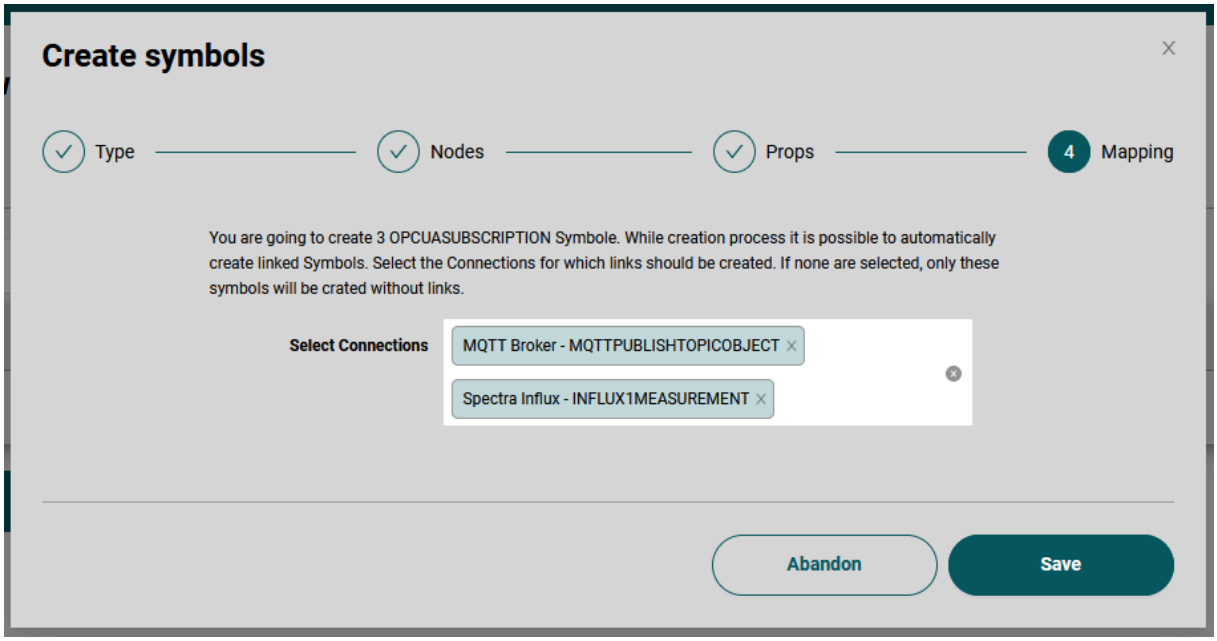
The OPC UA address space will be browsed and displayed. Select the nodes you like to sample.



Set additional properties such as sampling interval.



Configure auto mapping from the created input symbols to one or more output symbol of a specific type. In the example, the collector creates a MQTTPublishTopicObject symbol on the connection 'MQTT Broker' and an Influx measurement on the connection 'Spectra Influx' for each selected node in the previous step.



3.6.2 Create OPC UA BulkRead Symbols

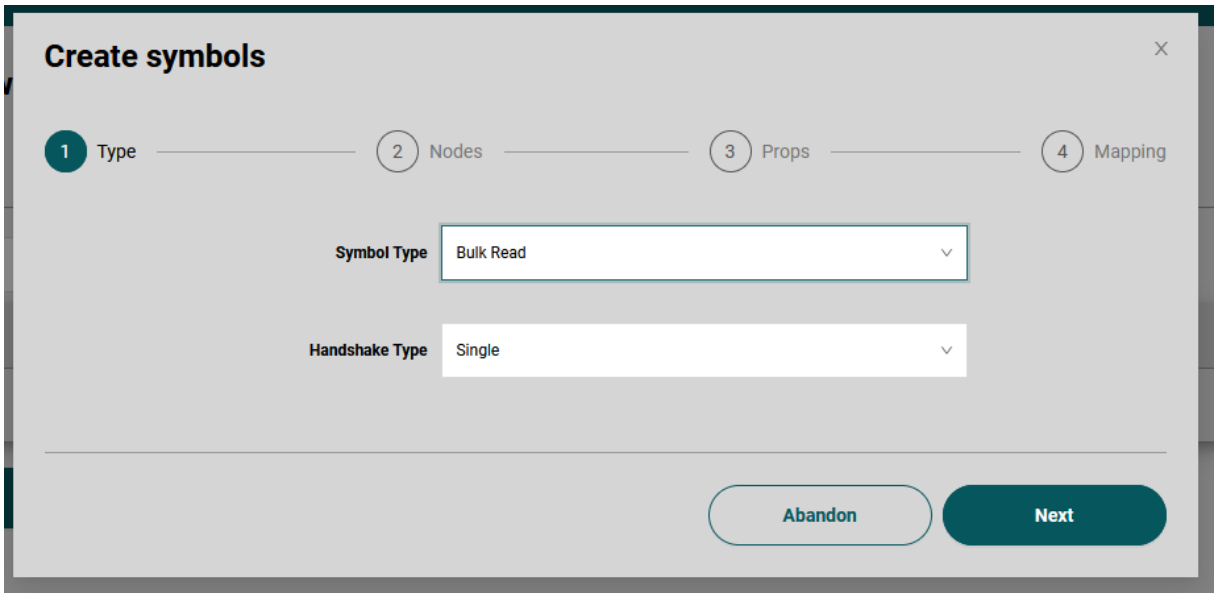
The BulkRead symbol is used to read a set of OPC UA nodes as a single structure. The collector and the OPC UA server communicate a handshake to signal when data is ready to read and when the collector has processed the data point.

Select the symbol type BulkRead in the 'Create Symbol' wizard and select an appropriate handshake type.

The handshake type **Single** requires the server to have a writeable boolean OPC UA node that is set by the server to *true* when the values should be read. The collector reads the values and sets the handshake variable two *false* after the successful read or write.

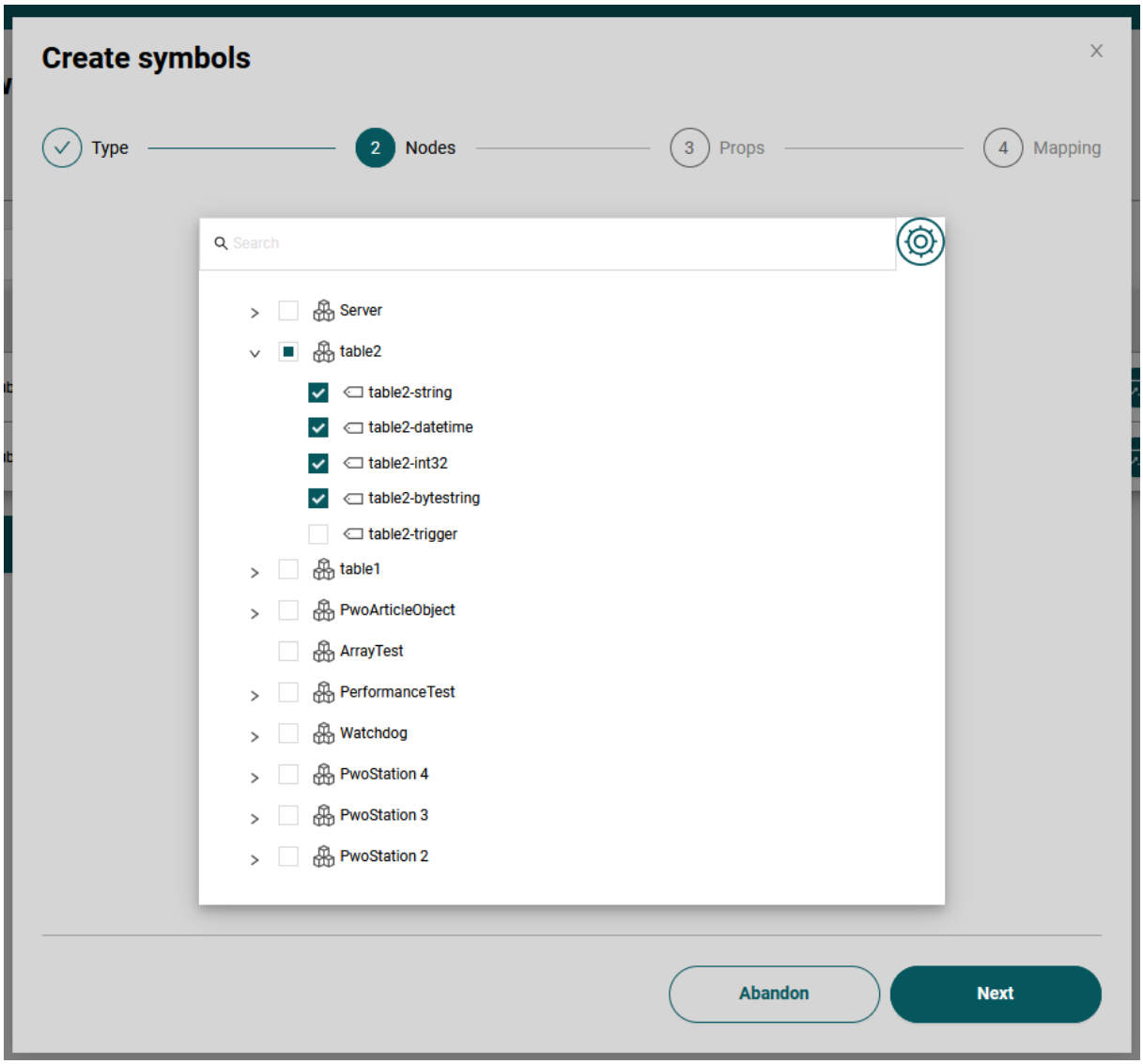
The handshake type **Input/Output** uses two boolean OPC UA nodes for the handshake. The OPC UA server sets the input OPC UA node to *true* when the collector should read the data nodes. The collector sets the output OPC UA node to *true* when the data points are successfully read or written. Then the OPC UA server must set the input node to *false*. Afterwards, the collector sets the output node to *false*, completing the handshake.

The handshake can be configured in both variants by specifying the 'Confirm After' type. With **Read** the collector confirms the handshake after successful read. The data point may not be written to an output symbol. With **WriteAny**, the collector confirms the handshake if at least one output symbol has written the data point successfully. With **WriteAll** the collector confirms the handshake if all output symbols have written the data point successfully.

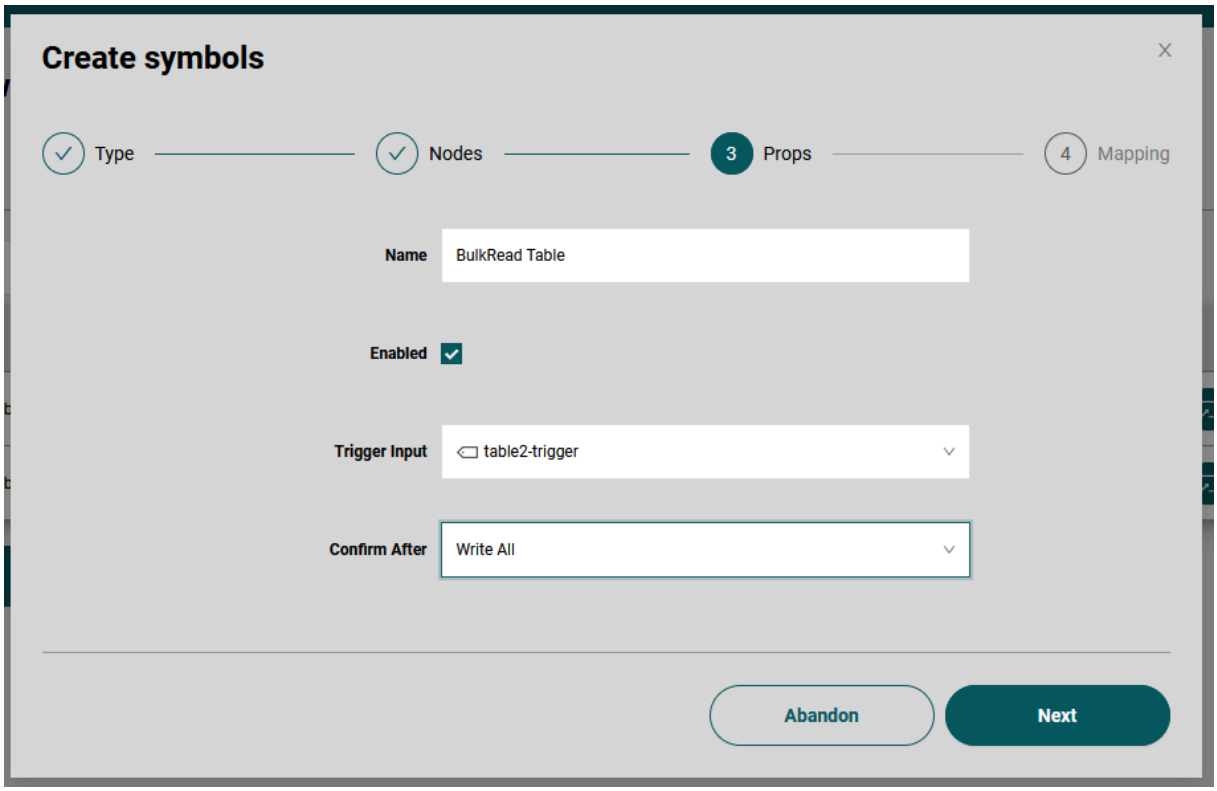


Single Handshake Configuration

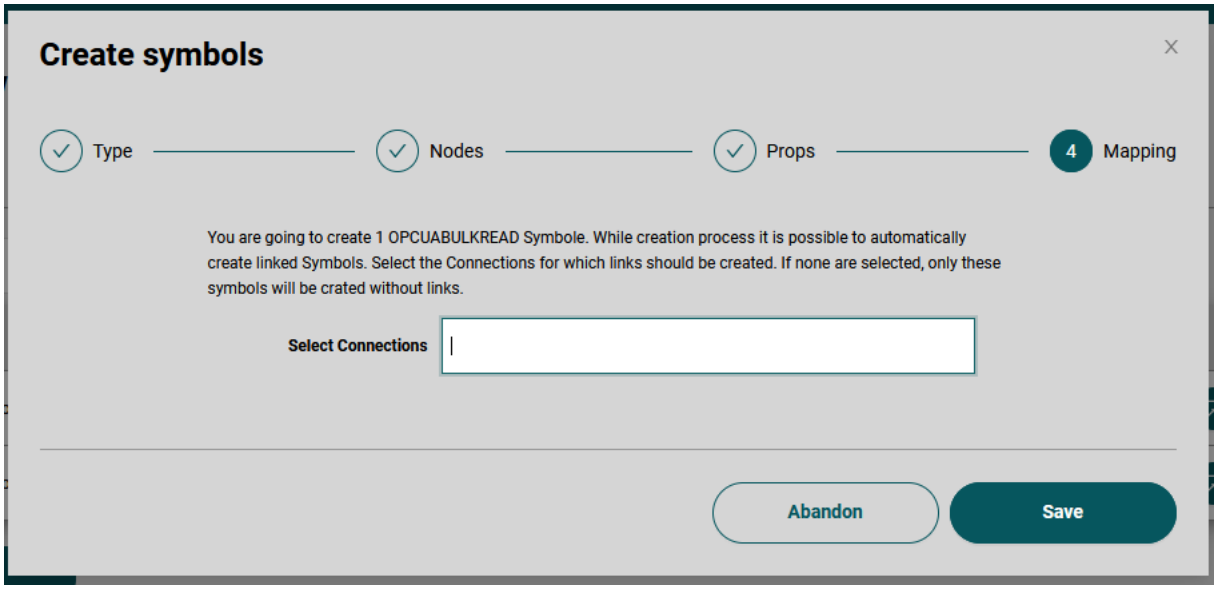
The OPC UA address space will be browsed and displayed. Select the nodes you like to sample. Do not select the trigger node but only the nodes that contain the data.



Select the trigger node.

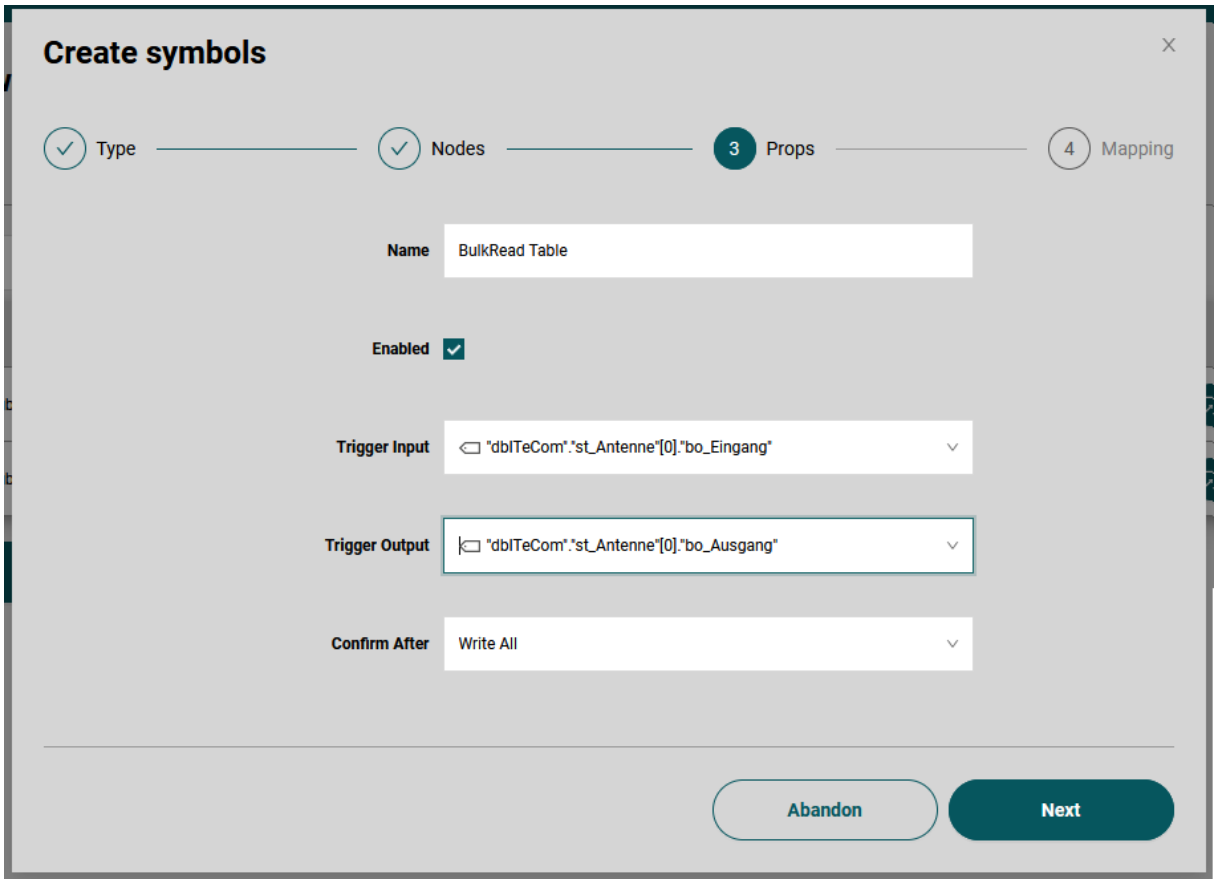


Configure auto mapping from the created input symbols to one or more output symbol of a specific type. In the example, the collector creates a MQTTPublishTopicObject symbol on the connection 'MQTT Broker' and an Influx measurement on the connection 'Spectra Influx' for each selected node in the previous step.



Input/Output Handshake Configuration

The Input/Output handshake type is analogously configured as the single handshake type. The third page is different and allows to select the input and the output trigger nodes.

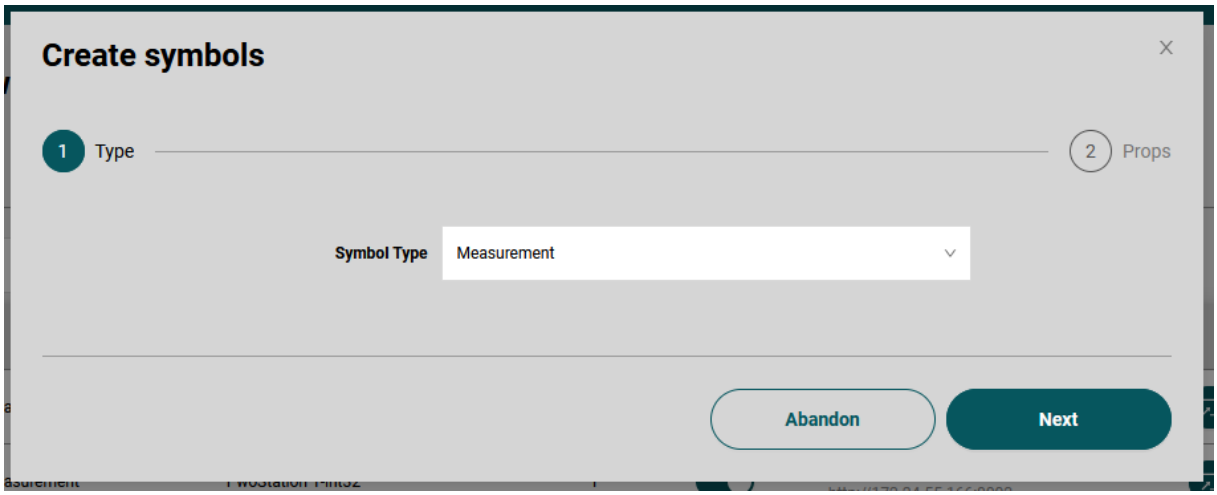


Note: Do not select the input and output trigger node on page two, otherwise you cannot select them on page three.

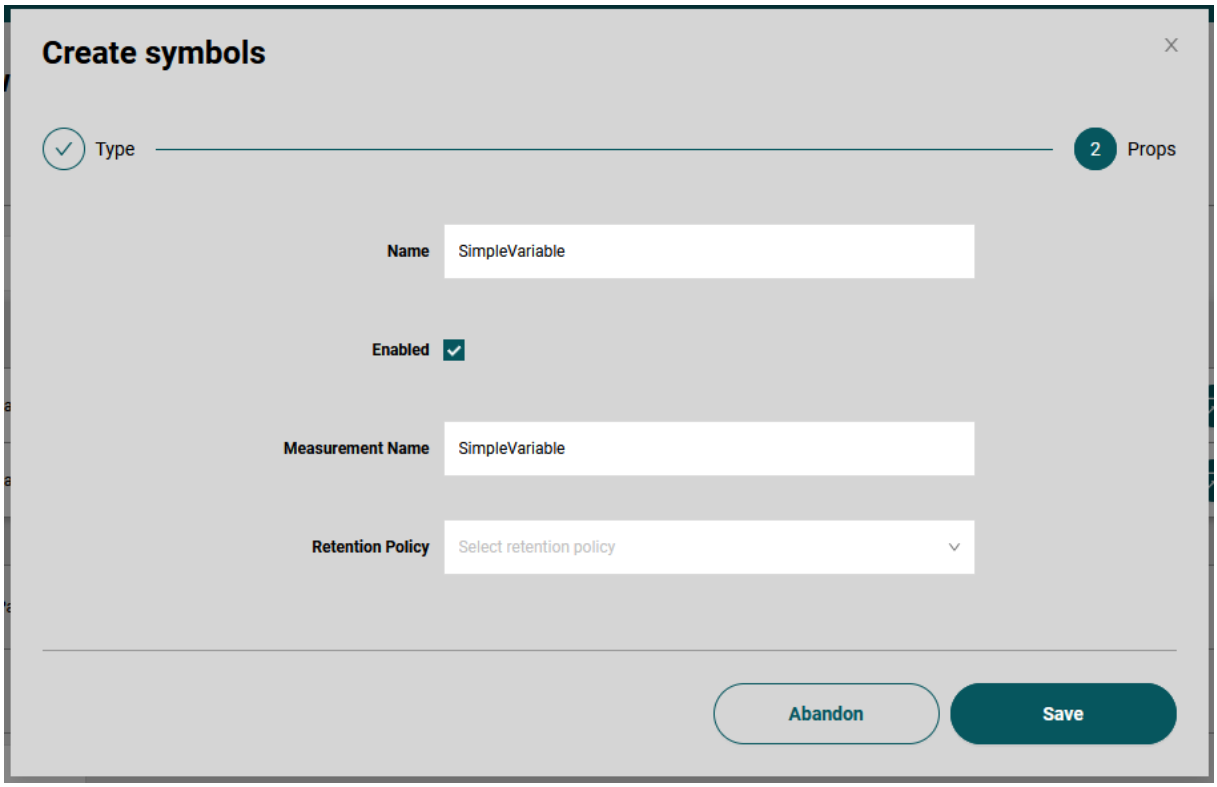
3.6.3 Influx1Measurement

Most of the time influx measurements will be created using the auto mapping features when creating one or more input symbols. However, often it is useful to create some measurements manually.

First select the influx connection that shall be used and create symbol. Select the symbol type **Measurement**.

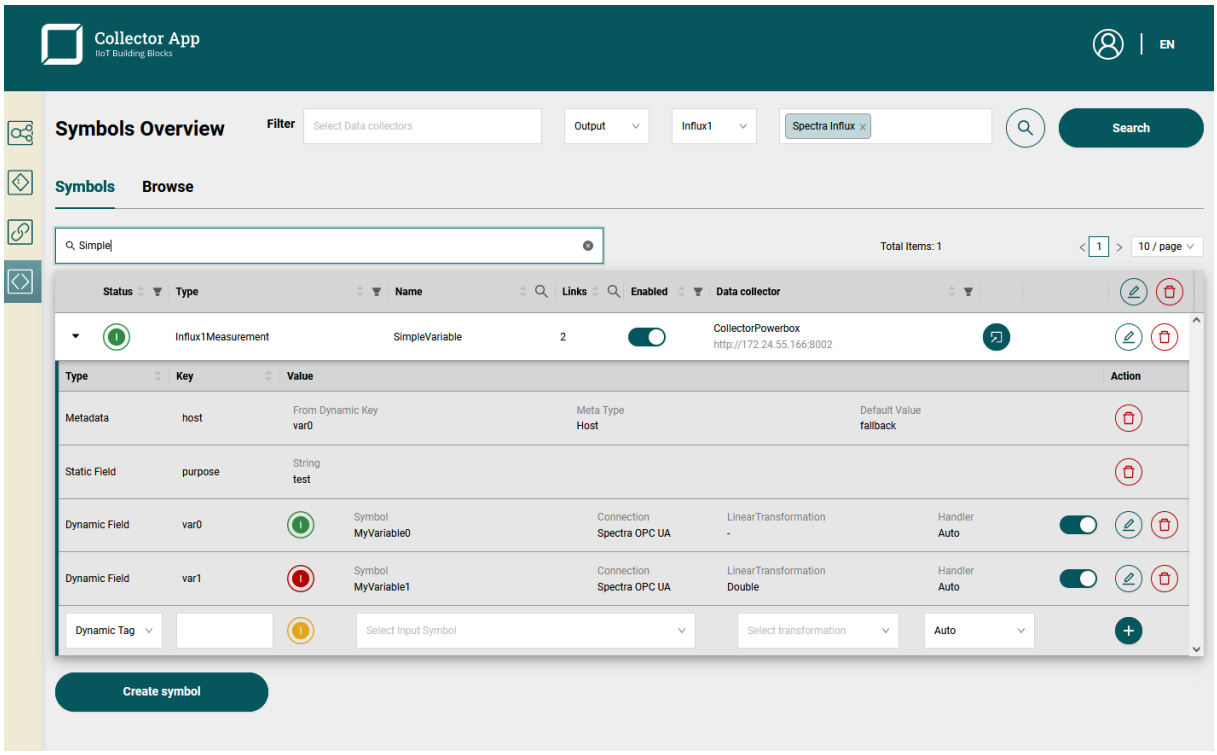


In the wizard only the most basic settings must be filled.



The **Name** will be displayed in the App. The **Measurement name** corresponds to the Influx measurement name. The **retention policy** can be left open and data points will be written in the default retention policy of the influx database.

Finally, the tags and fields can be configured in the created measurement. Click on the measurement to expand the tags and fields. New fields can be added using the last row.



There are 5 types of configurable rows:

1. **Metadata**: fills in metadata information from the selected input symbol. Provide a fallback value to prevent points to be discarded, when metadata is missing. An example for the metadata is the **Host** metadata which is used to tag the hostname of the connection of the input symbol. Only metadata of configured dynamic tags or fields can be used.
2. **Static field**: A simple text that will be added as a field of the measurement. *Note*: currently only strings will be stored, even if the text corresponds to a number.
3. **Static tag**: A simple text tag which will be added to the tag section of the measurement.
4. **Dynamic field**: The data points of the configured input symbol will be inserted as the field with the configured key. It is possible to configure a linear transformation and restrict the type. In most cases it is useful to keep the auto type.
5. **Dynamic tag**: The data points of the configured input symbol will be inserted as a tag with the configured key. The data type of the data points must be a number (integer, not a fractional), a boolean or a string. Other types are not supported as a tag. **WARNING**: Dynamic tags may reduce the performance of the influx database. For each unique tag value, a separate time series is used internally in the Influx database which restricts performance. However, the feature can be useful since Influx Queries can use *WHERE* queries on tags but not on fields.

Note that if multiple dynamic tags or fields are used, the restrictions of join functionality will apply.

Handling of Arrays in Influx Measurements

If a data point that should be written to a measurement field arrives, the collector creates multiple influx data points from the array. It uses the tag 'index' to identify the index of the single point in the array. If a static or dynamic tag is configured with the key 'index' it will be overwritten. **Note** that for each index a separate time series is created by the Influx database internally.

Handling of Structures in Influx Measurements

If a data point is a structure, e.g. through a OPC UA BulkRead or by the data type in a OPC UA Subscription, the structure can also be written in the influx database by flattening the structure. Only dynamic fields support structures; dynamic tags cannot be structures.

Take a look at the example structure (as json):

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

The corresponding line protocol for a dynamic field with key 'data' at timestamp 132909231230223 looks like:

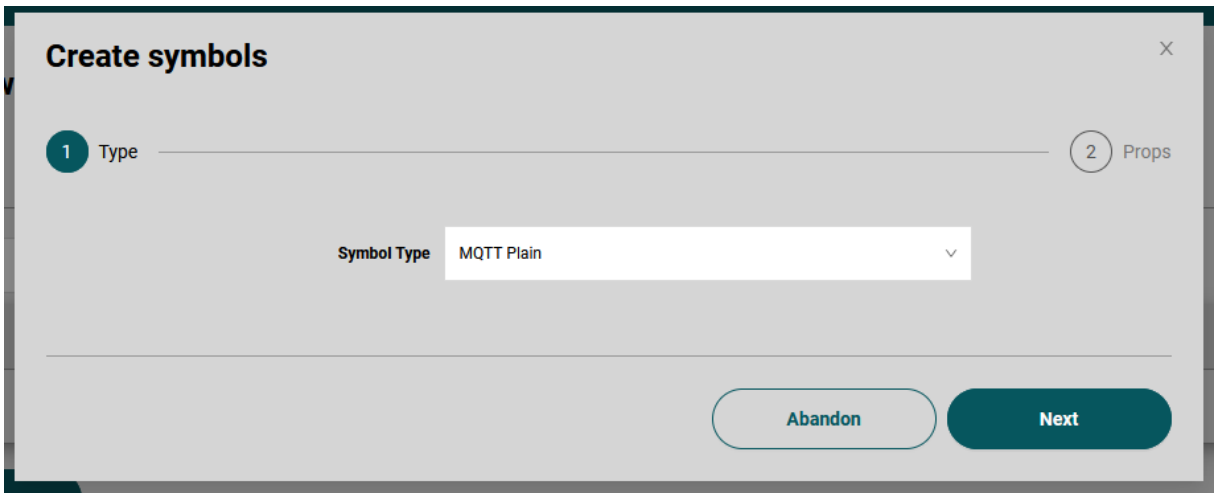
```
name data_anBool=f,data_anInt=3,data_nested_aString="world" 132909231230223
```

Note that arrays inside structures are not supported. However, an array of structures is supported.

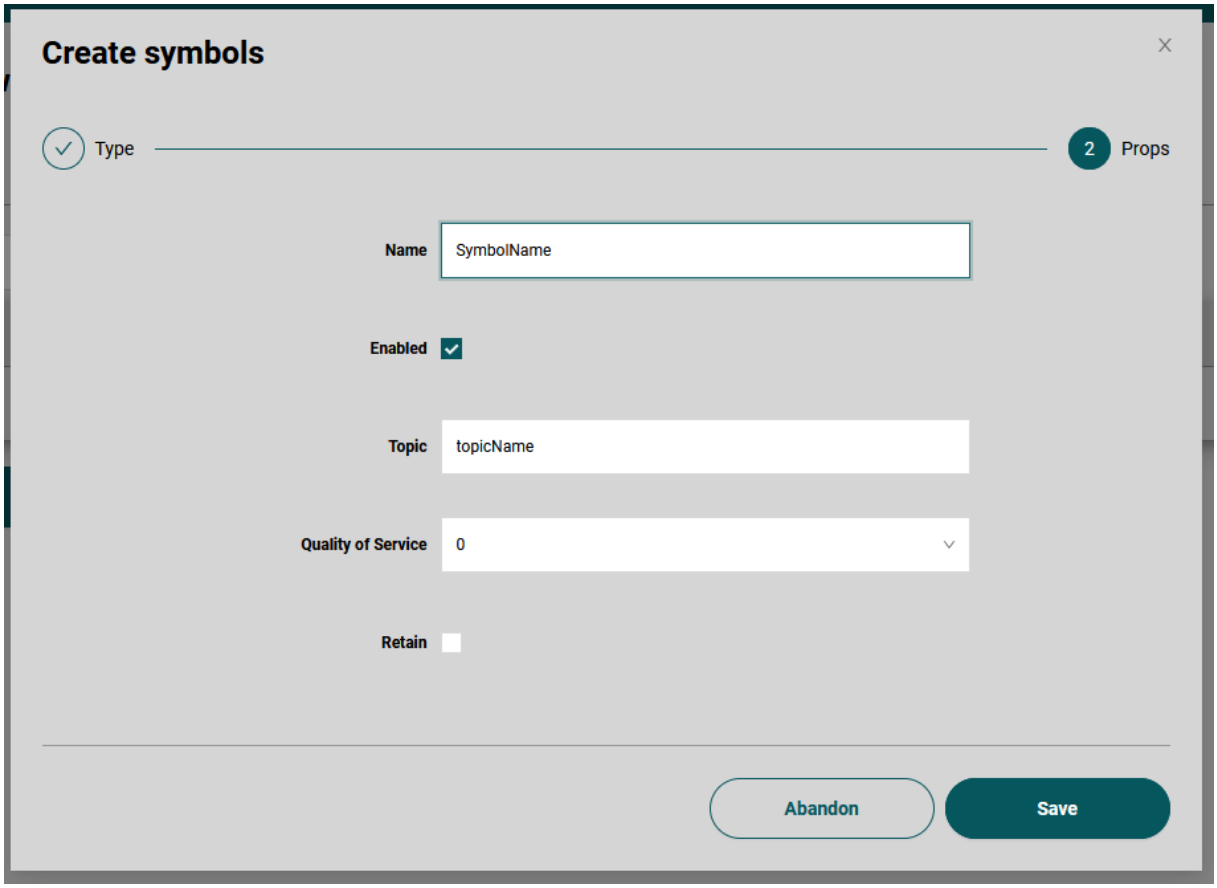
3.6.4 Create MQTT Publish Topic Plain

Most of the time MQTT Publish Topic Plain will be created using the auto mapping features when creating one or more input symbols. However, often it is useful to create some symbols manually.

First select the MQTT connection that shall be used and create symbol. Select the symbol type **MQTT Plain**.

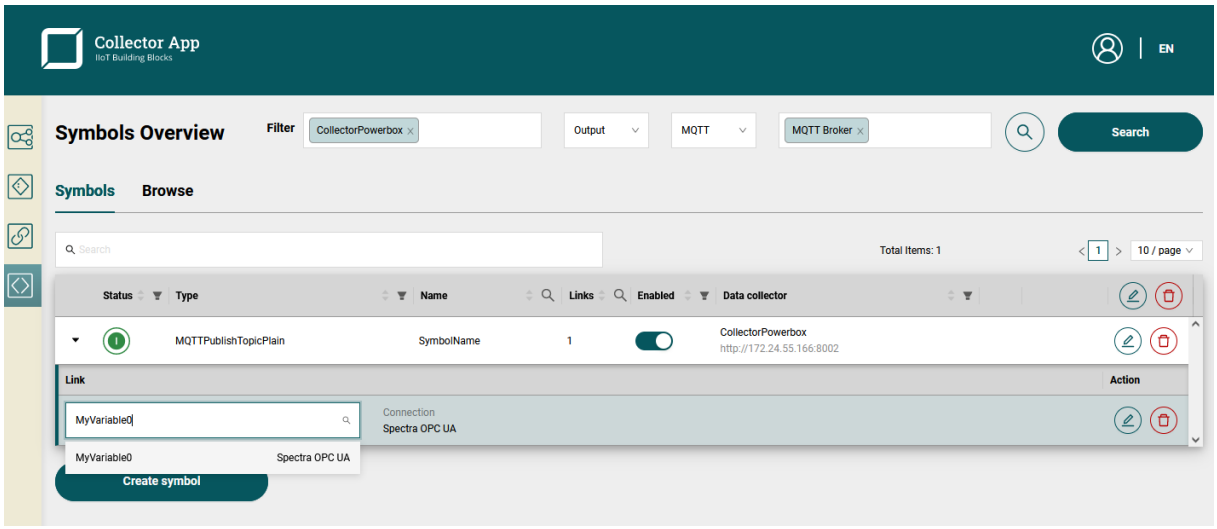


In the wizard only the most basic settings must be filled.



The **Name** will be displayed in the App. The **Topic** corresponds to the MQTT topic where the data points are published to. The quality of service can be configured. The retain flag can be configured as well.

Finally configure the input symbol in the expanded row in the symbols table.



The data points will be written as json object. The keys of the json object can be configured. The result of a number data point will be the json

```
413658032
```

a string will be the json

```
"hello world"
```

an array will be converted to a json array

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

and a structure will be converted to a json object

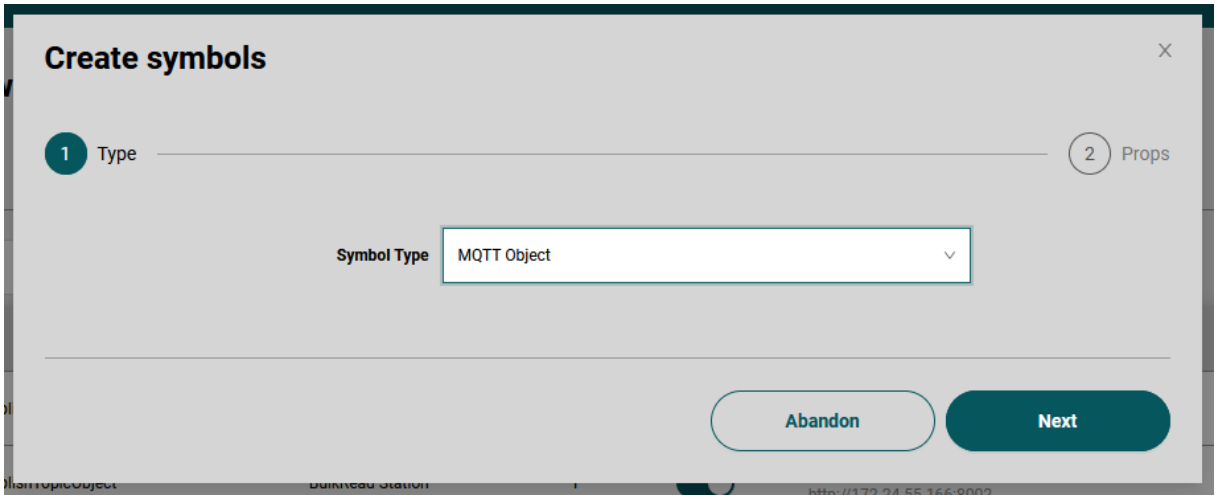
```
{
  "hello": "world",
  "anotherKey": false,
  "nested": {
    "structures": 2,
    "arePossible": -23.3
  },
  "anArray": [true, false, true, false]
}
```

3.6.5 Create MQTT Publish Topic Object

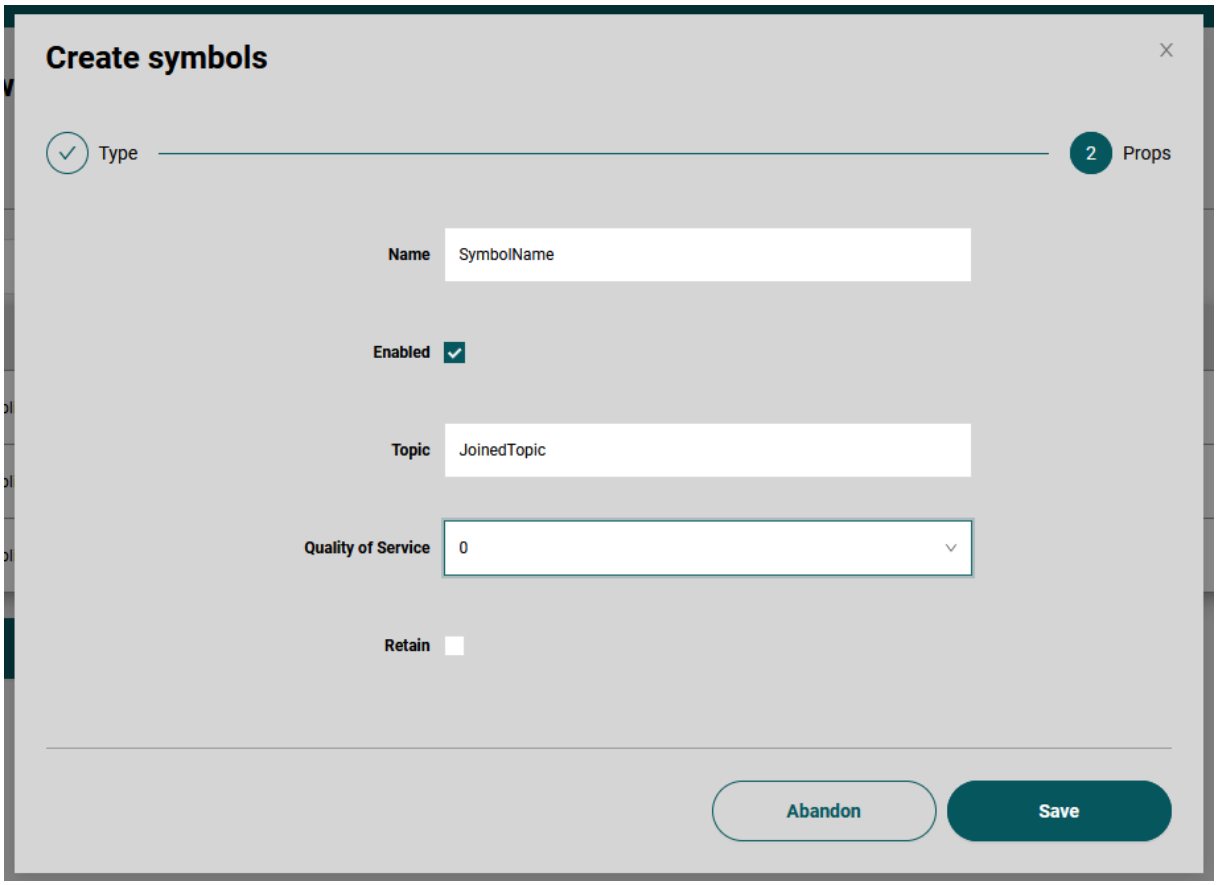
Most of the time MQTT Publish Topic Objects will be created using the auto mapping features when creating one or more input symbols. However, often it is useful to create some symbols manually.

The data points are converted to json and the timestamp in (ISO8061) will be added to the object as well. An example of a converted data point is shown at the end of the section.

First select the MQTT connection that shall be used and create symbol. Select the symbol type **MQTT Object**.

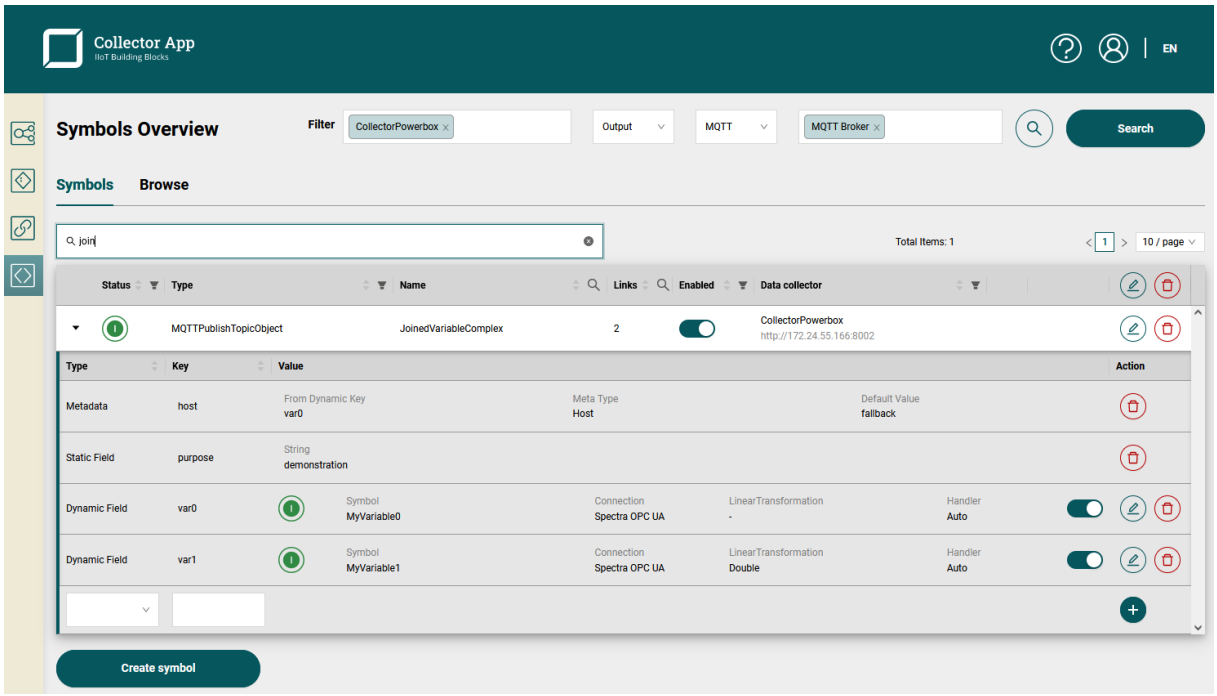


In the wizard only the most basic settings must be filled.



The **Name** will be displayed in the App. The **Topic*** corresponds to the MQTT topic where the data points are published to. The quality of service can be configured. The retain flag can be configured as well.

The data points will be written as json object. The keys of the json object can be configured.



There are 3 types of configurable rows:

1. **Metadata**: fills in metadata information from the selected input symbol. Provide a fallback value to prevent points to be discarded, when metadata is missing. An example for the metadata is the **Host** metadata which is used to tag the hostname of the connection of the input symbol. Only metadata of configured dynamic tags or fields can be used.
2. **Static field**: A simple text that will be added as a field of the measurement. *Note*: currently only strings will be stored, even if the text corresponds to a number.
3. **Dynamic field**: The data points of the configured input symbol will be inserted as the field with the configured key. It is possible to configure a linear transformation and restrict the type. In most cases it is useful to keep the auto type.

The result of the example configuration is:

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

Note that if multiple dynamic fields are used, the restrictions of join functionality will apply.

3.6.6 Restrictions of Join Functionality

The Data Collector allows to join data points from multiple input symbols on a best-effort principle. The join functionality can be used by specifying multiple input symbols in an output symbol. For an example see the configuration of an Influx1Measurement using multiple dynamic fields.

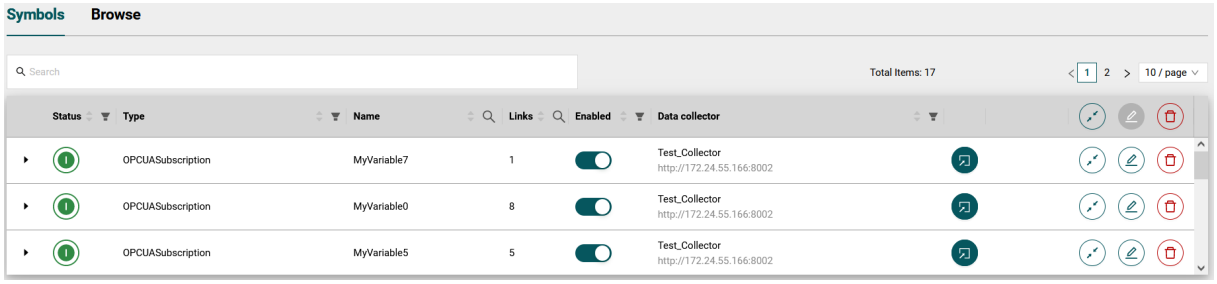
Best-effort principle means that the collector will join data points in the order that they are sent from the source to the data collector. This means that two data points are joined correctly if they have the same time stamp and arrive at the same time at the Data Collector. If two data points have the same time stamp but arrive with a jitter of 2 seconds, the joined data points will be incorrect until the second data point arrives. For most OPC UA servers, this is true for OPC UA MonitoredItems, which are used in the input symbol OPC UA Subscription.

If input symbols of different connections are used, joining will not be very accurate. The reason is that OPC UA transfers bulks of data points per OPC UA server. Most likely data points joined from multiple connection will very at least a few hundreds milliseconds, but may be inaccurate by multiple seconds. Moreover, connection losses on different OPC UA servers increase the inaccuracy even further.

3.6.7 Set Mass Symbols

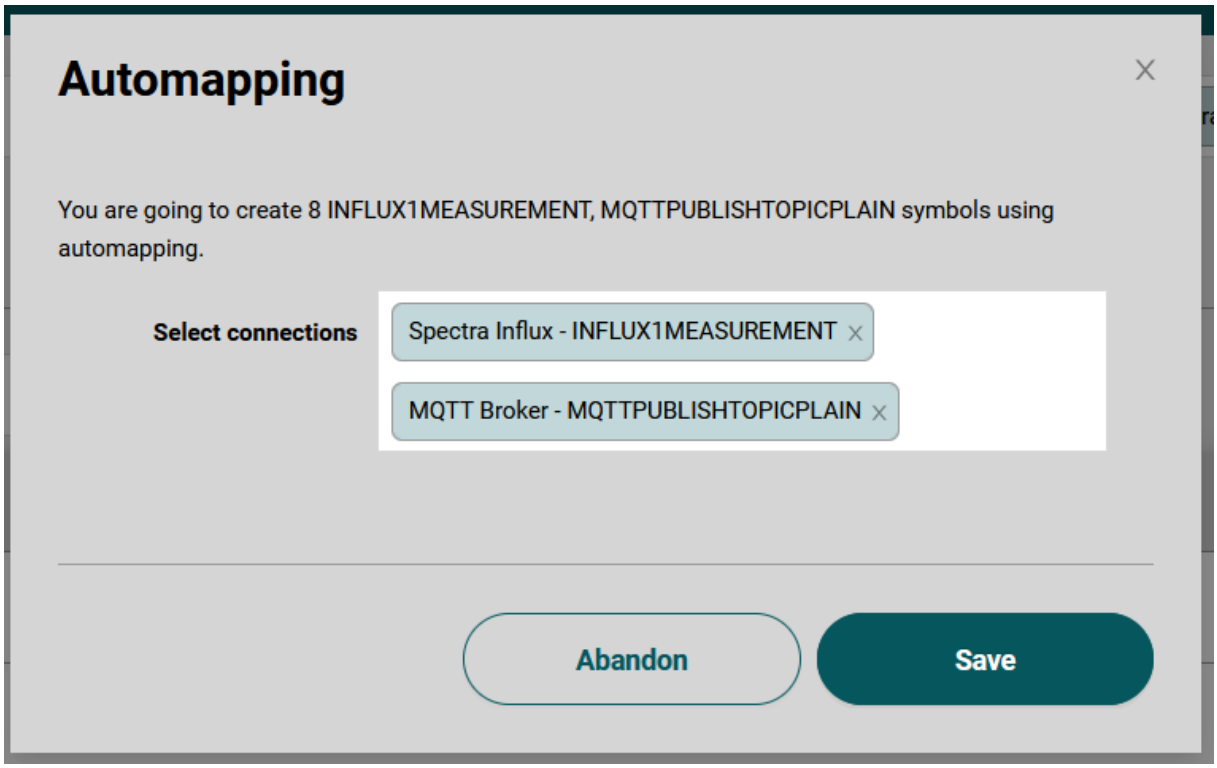
In the symbol table in the header there are buttons on the right, that can edit or delete all symbols in the table at the same time. These are called mass buttons.

A buttons that are located in a single row operates only on its symbol. A mass button operates on all symbols selected in the table. To operate only on a subset of the symbols, use the filter mechanisms of the search field and table header to select the symbols to modify. Note that all selected symbols are edited not only the visible page of the table. In the figure below, all symbols of table page 1 and 2 will be modified if the mass button is pressed.



Automapping Button

The automapping function allows you to quickly create for one or more input symbols one output symbol per output type. To do this, the connections of the output types must be selected.

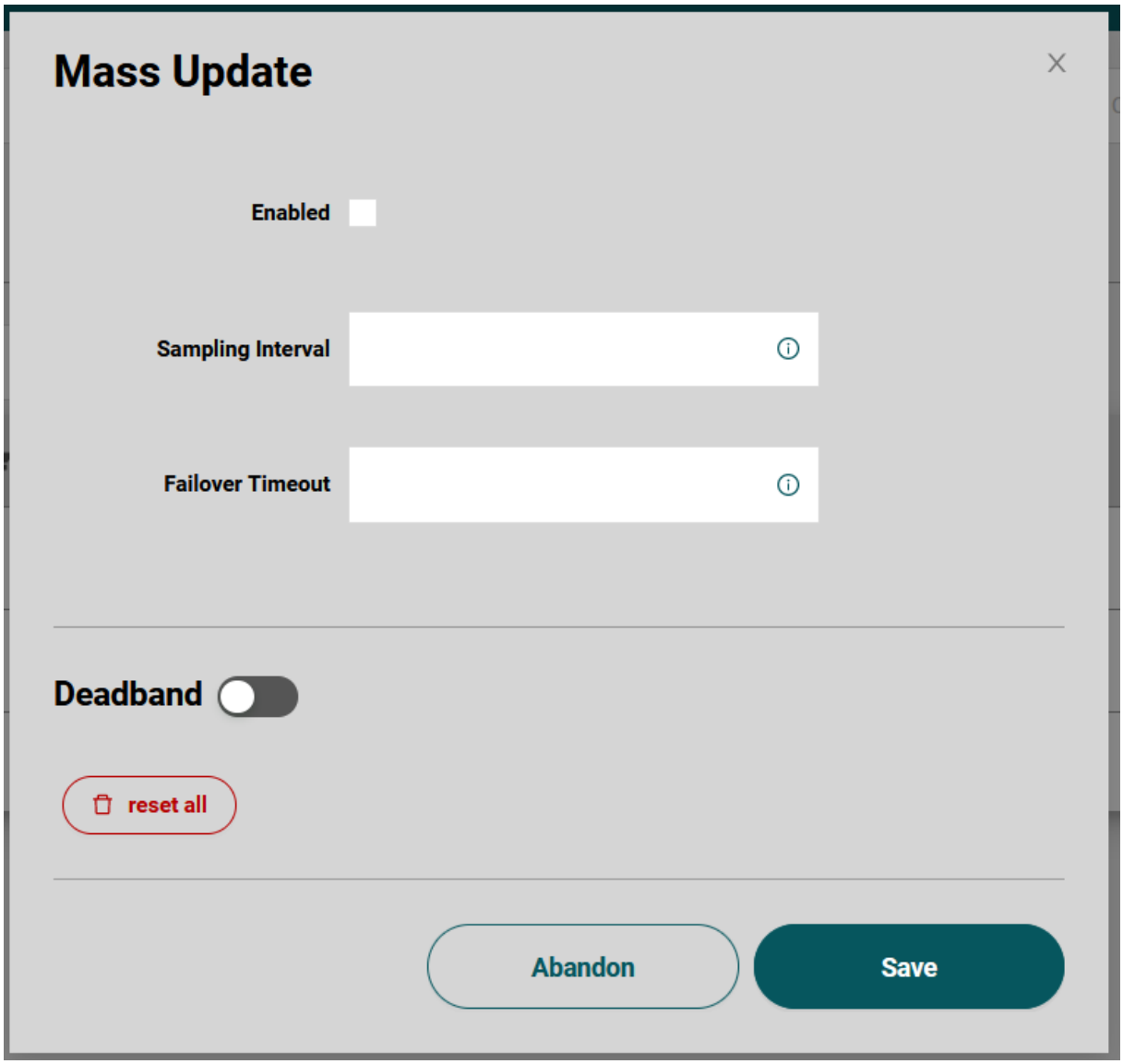


An automapping button in a symbol row will create an output symbol for each selected connection. Using the mass automapping button in the table header will create for each input symbol an output symbol for each selected connection.

Edit Button

The Edit button allows you to edit one or more symbols. For example an interval, a timeout and a deadband can be set for OPCUASubscription. In the normal editing function for single symbols, the deadband can be switched off by deactivating the switch and saving the modal.

Note that it is only possible to mass edit symbols of the same type. For example, if the table contains OPCUASubscription and OPCUABulkread symbols, the mass edit button will be disabled. Filter to a single type using the table header filter to enable the mass edit functionality in such a case.

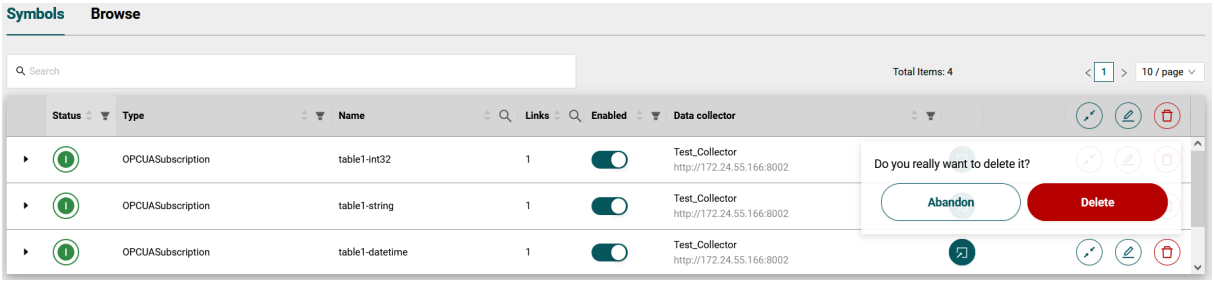


With the multiple edit function there is an additional button, with which the deadbands of all symbols can be reset at the same time. Here the modal must also be saved for the reset of the deadbands to be effective.

Delete button

The Delete button allows you to delete one or more symbols. Individual symbols are deleted by the delete button in the respective symbol line. If the multiple delete button is pressed in the header then all symbols listed in the table will be deleted at the same time.

In order to avoid deleting something by mistake, the user must confirm the delete action in the popup dialog before the symbols will be deleted.



3.7 User administration

Initially an admin user is created:

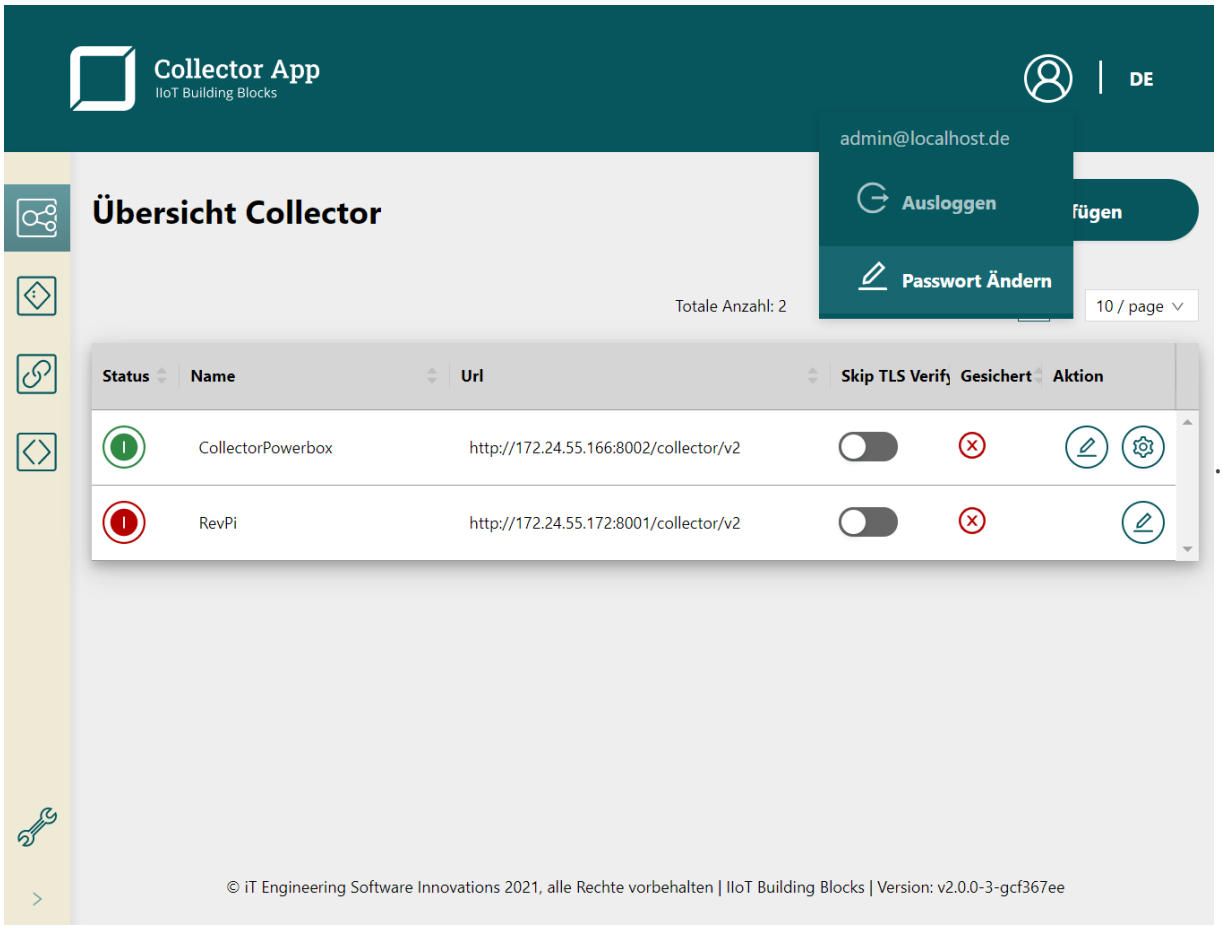
username	password
admin	admin

Warning

After the first login with the admin user the password should be changed!

3.7.1 Change password

If you hover over the user menu in the upper right corner you will find the entry Change Password:



3.7.2 User roles

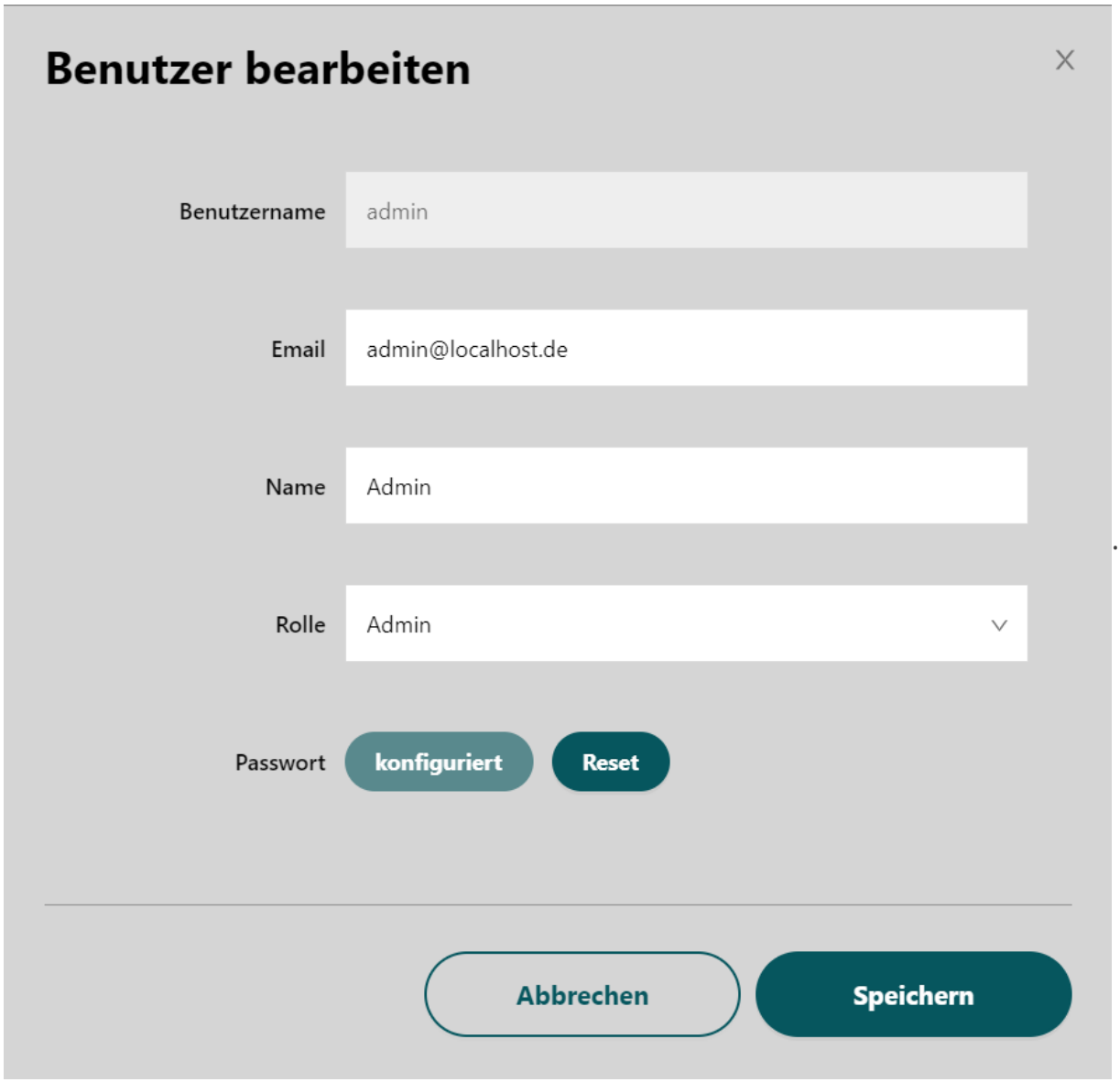
Role	Description
Admin	The role "Admin" has the permission to manage users. That is to create users, delete them, and reset passwords.
Developer	Developers can anything expect add/edit/delete influx retention policies and users management.

3.7.3 User Create / Edit

At the bottom left of the settings you will find the user management.

Each user has a unique username. This cannot be changed. The name is optional. Here you can enter the full name of the user. Each user is assigned role.

The password of the user is not visible, because it is only stored as hash after creation. However, it can be reset to a new password using the reset button.



3.7.4 Password reset

If a user has forgotten his password he has to contact an administrator who can reset the password in the user management of the app.

If the administrator password is lost it can be reset to the default password via the Collector App Command Line Interface.

On windows, the CLI is installed in the installation directory during installation. It can be executed via the command prompt (cmd):

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

In the 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

- Introduced support for configuring OPCUAEventListener
- Added claim-based authentification for fine-granular access on collectors and connections

Fixes

- Fixed password errors when updating a user

3.10 v2.3.0 2023-12-01

Features

- The collector-app-cli reset-admin-password now accept database path argument
- The iTE_SI_Collector-App.exe can be started using a custom working directory (--working-dir) instead of using the %APPDATA% folder
- The bundled license runtime is removed from the installer and the collector apps works without it. Improved information about licensing can still be retrieved from App, when runtime is downloaded separately
- UX Improvements like storing number of entries in a table or selected connections when returning to a page
- UX: Improve resetting deadband configurations for OPCUASubscriptions
- Improve URL handling for different URLs/IPs when browsing the collector app.

Fixes

- Respect InstallFolder property in Installer
- Immediate logout when session cookies become invalid instead of showing an error
- Fix issues on browsing or showing information when navigating from one symbol to another
- Fix invalid validation on URLs
- Improved OPCUABrowser handling when opc.tcp://localhost URLs are used and multiple OPCUABrowsers are used
- Some ApplicationErrors in different scenarios.
- Improve handling of URLs for Collector URLs and OPCUA Browser URLs.
- Fix an application error on some symbol filters

- UX: LinearTransformation popup did not show in some scenarios

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 operations
    - Linking manually or with automapping
- Unified step-based dialog for creating symbols
- The app can now display unknown connection and symbol types and allows base operations:
    - Edit (switch on and off)
    - Delete
    - Status display

Bugfixes

- Problems with saving collector settings
- Wrong size of editable table cells with evaluation

3.10.3 v2.0.1 (26.11.2021)

Features

- Influx1Connection authentication with username and password
- Automapping for MQTTPublishTopicPlain and MQTTPublishTopicObject output symbols
- MQTT connection with output symbols
- Better performance for mass create symbols
- Duplicate connections
- Import and export connections

Bugfixes

- When creating metadata columns, it is possible to select from dynamic keys (not all).
- Grafana Datasource settings.

3.10.4 v2.0.0 (25.06.2021)

Features

- Feature complete with v1 (except configuration file import/export)

4. iTE Data Collector

4.1 General

The Data Collector is optimized to collect measurement data at the field level. It collects and stores the data from the machine or plant.

4.2 Installation

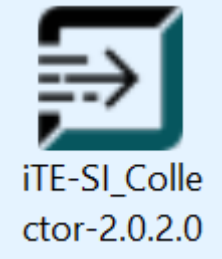
4.2.1 Windows

Download Installer

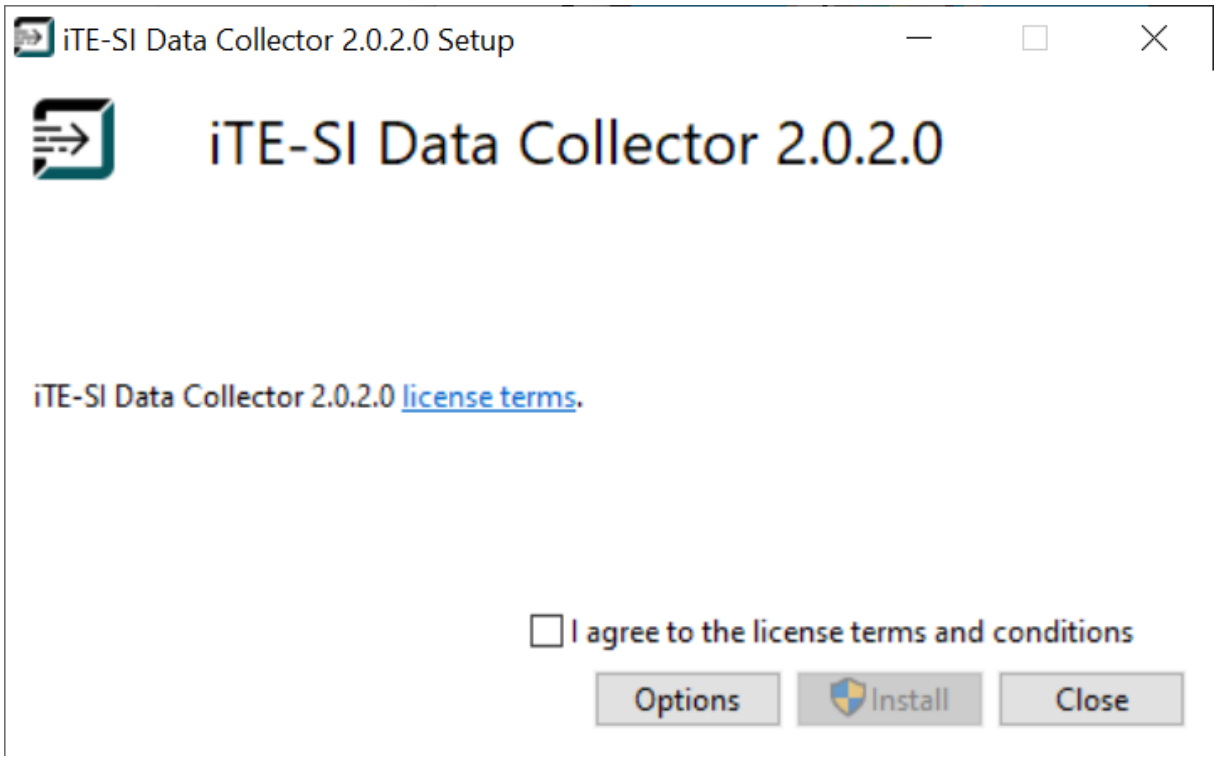
The Windows Installer can be downloaded here: IIoT Building Blocks Downloads

Installieren

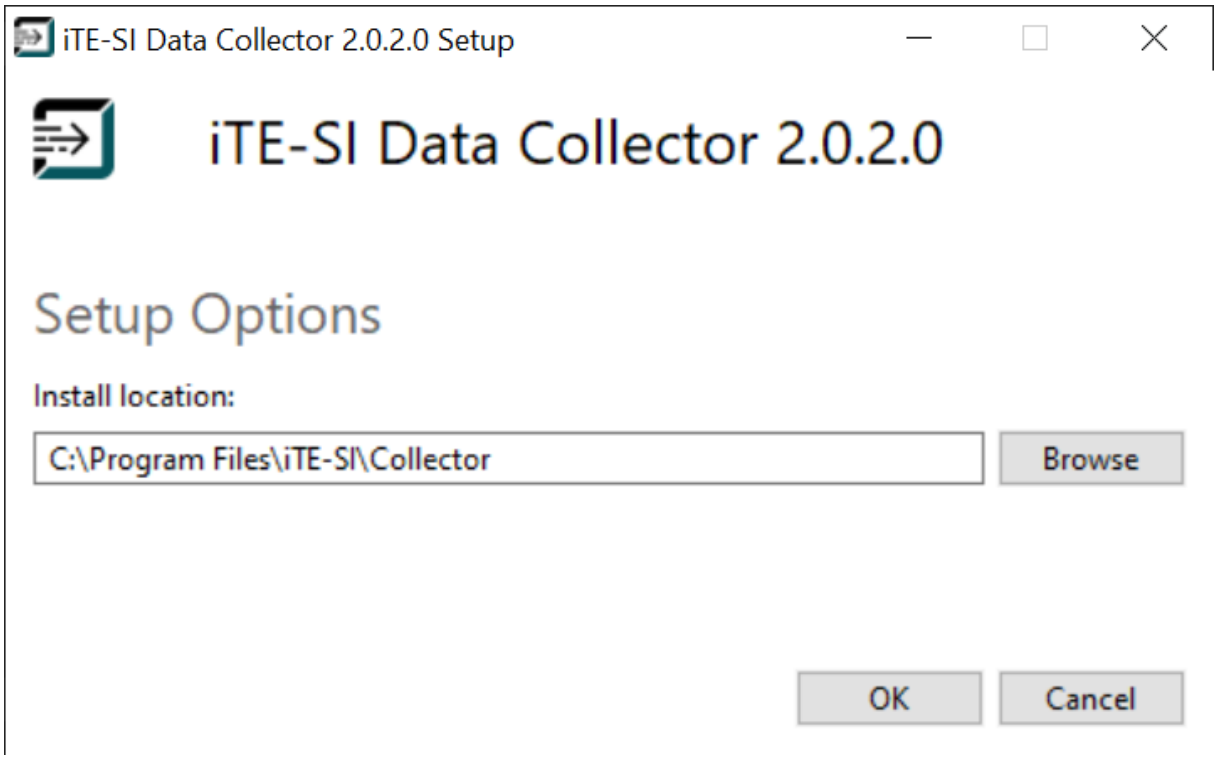
1. Invoke the installer.



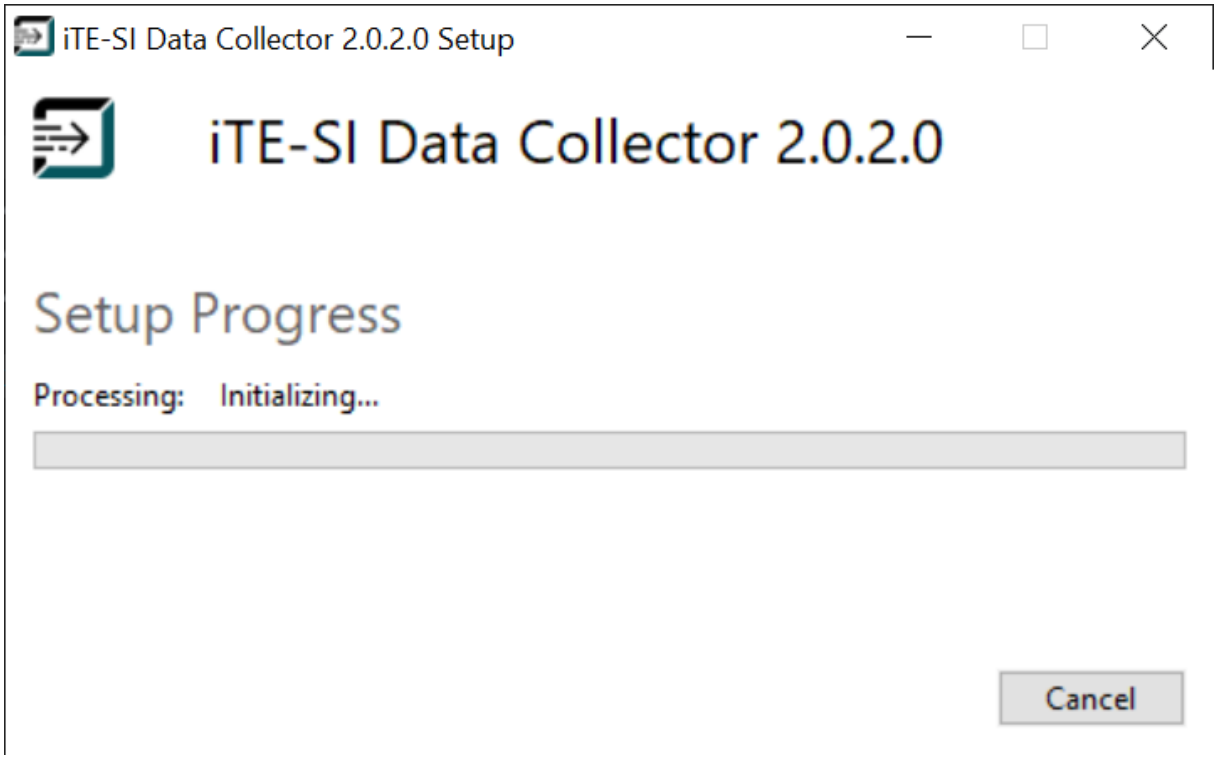
2. Read the End User License Agreement, accept it if necessary and press `Install`. Otherwise press Cancel to abort the installation.



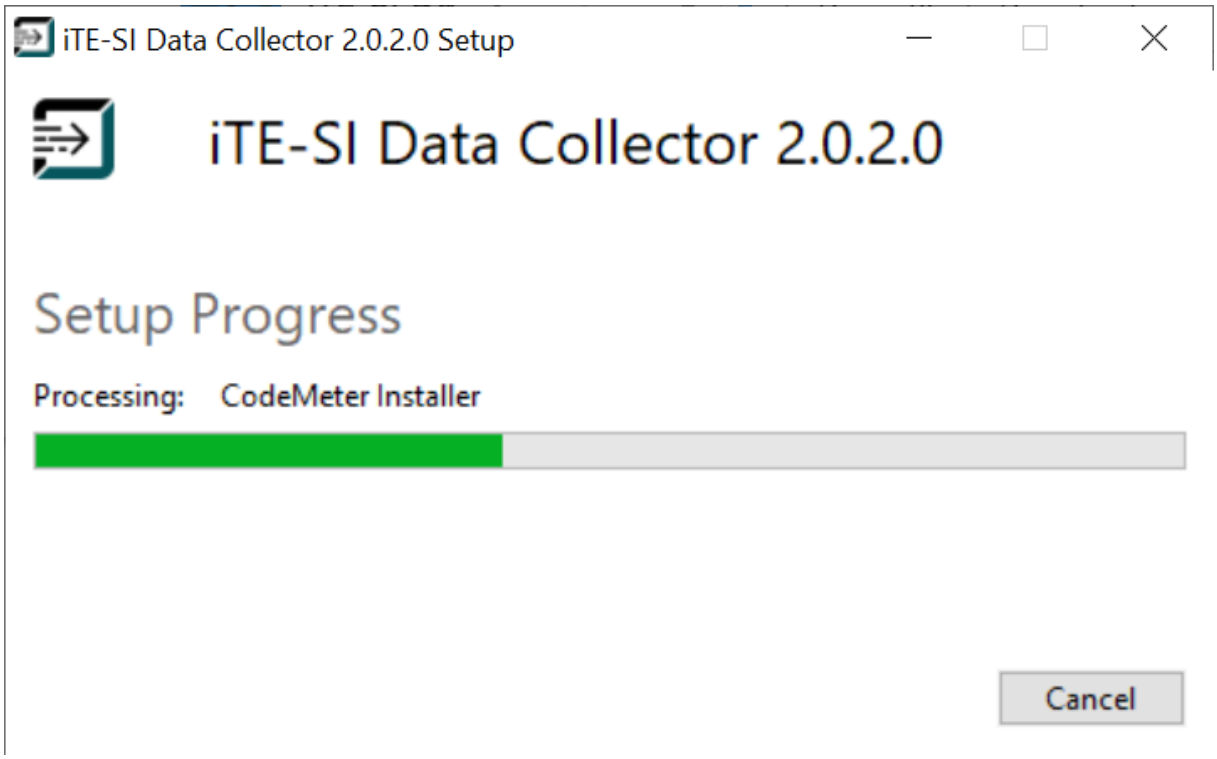
Optional: Select the installation directory in Options.



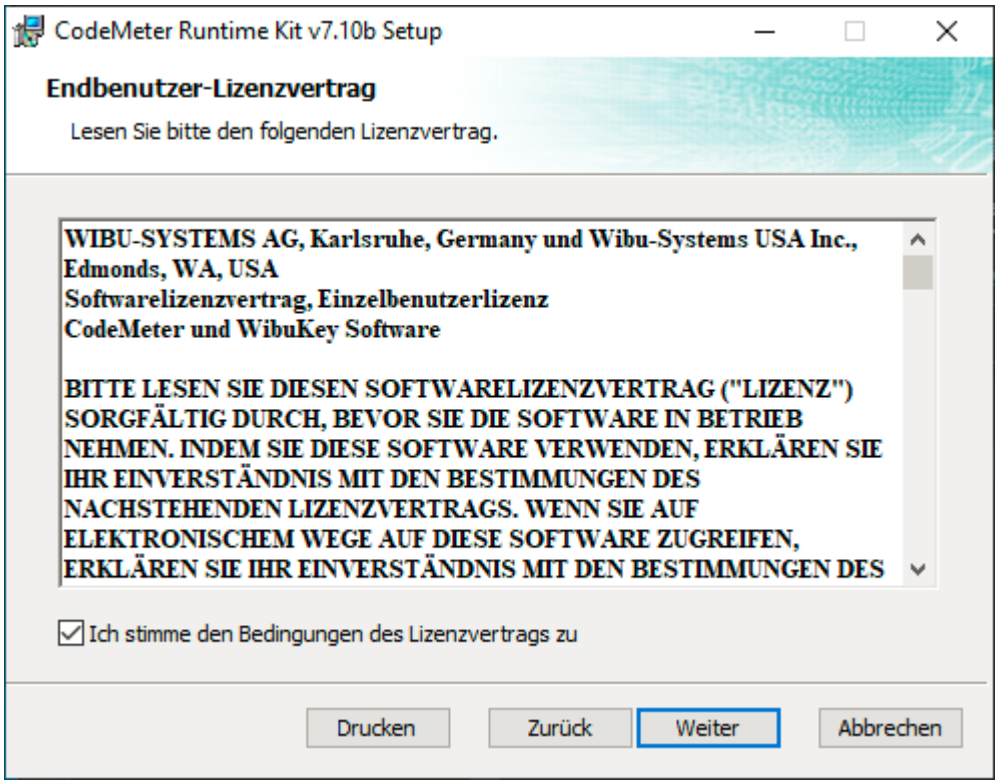
3. Wait a second...



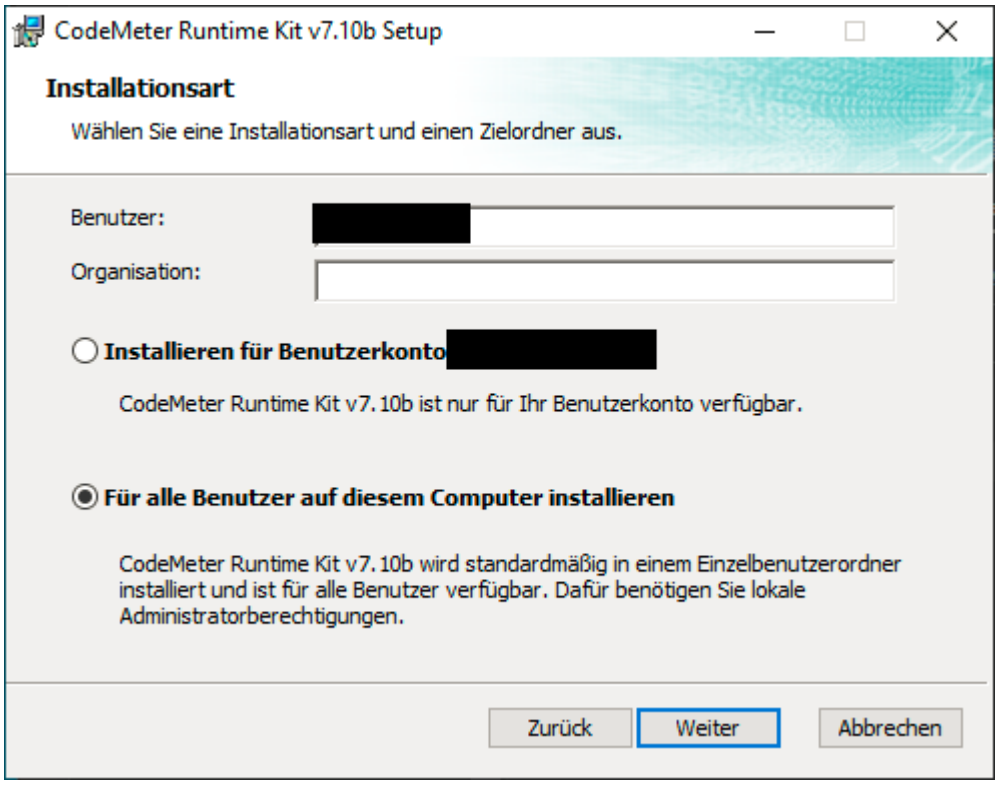
4. Optional: The iTE-SI_Collector requires the CodeMeter Runtime. If this is not available on your computer, it will be installed by a separate installer that runs automatically. Otherwise this section will be skiped.

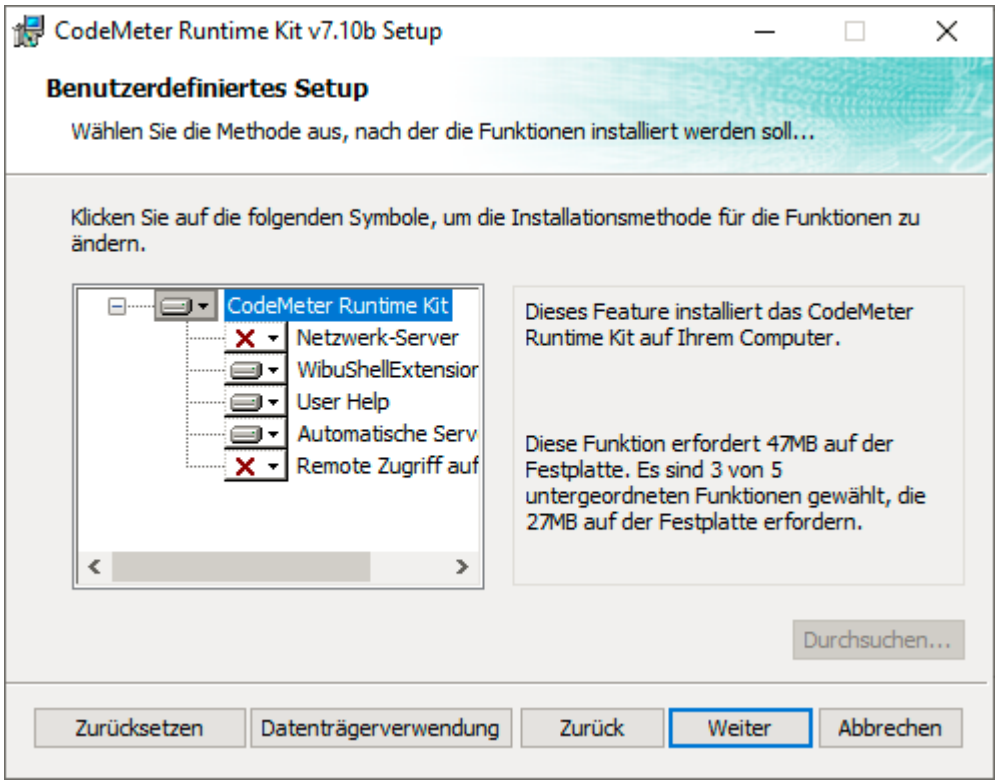


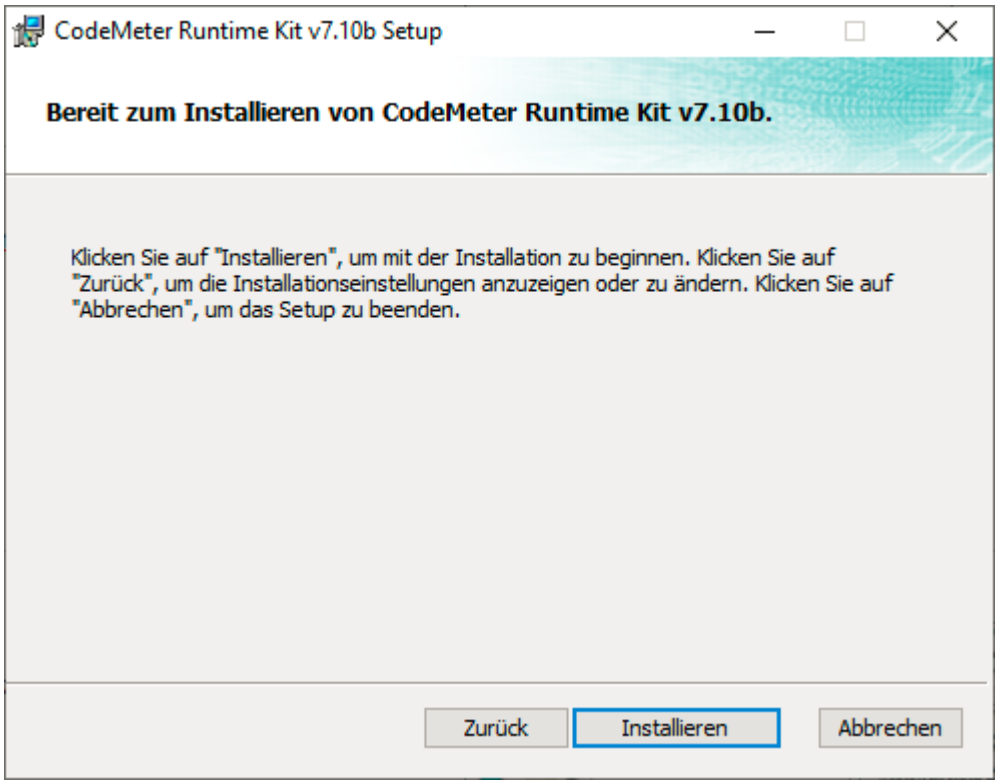
On the second page of the CodeMeter Runtime installer you have to read and accept an end user license agreement.

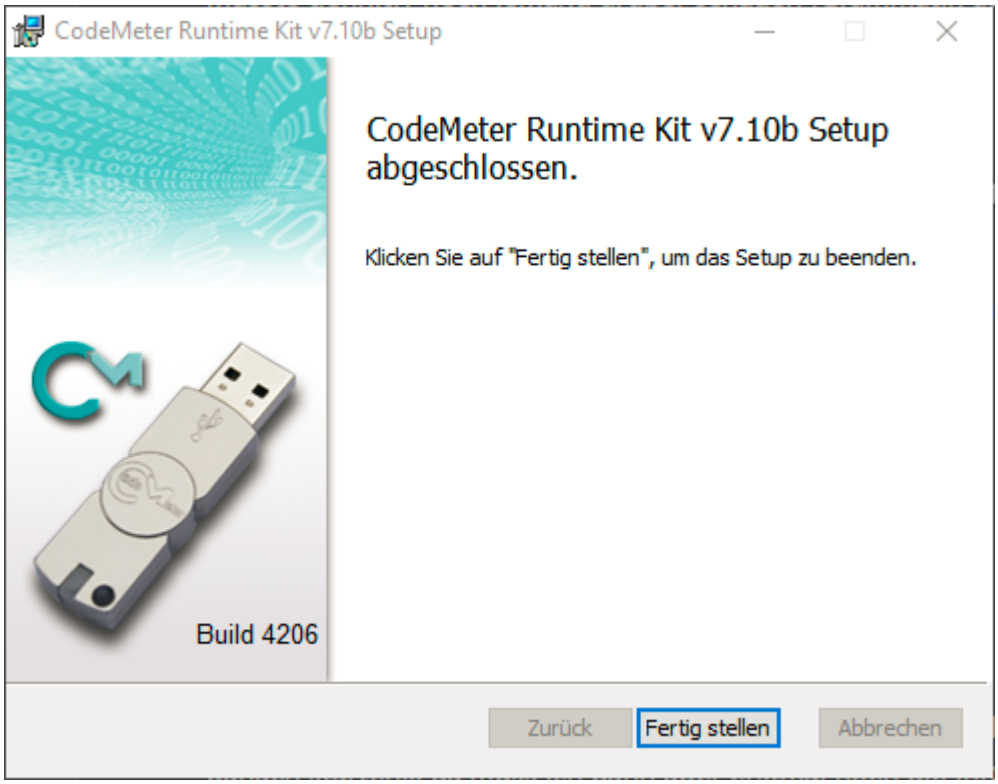


After that just press Next a few times, Install and Finish.



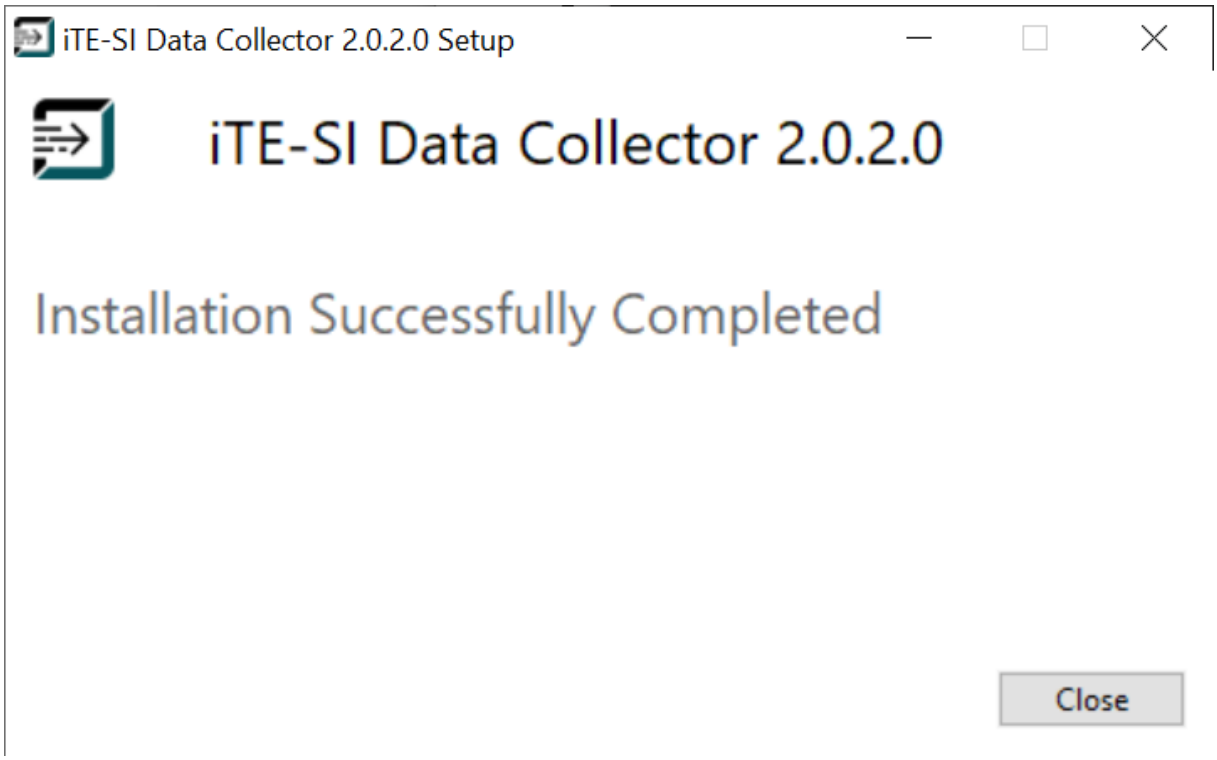






After the installation of the CodeMeter Runtime, the installation of the collector continues.

5. The installation has been completed successfully. Press Close.



6. The Collector can now be started via start menu.

4.2.2 Docker

The Docker image can be downloaded here: IIoT Building Blocks Downloads

1. Loading the image

Info

x.x.x.x must be replaced by the current version!

Info

If rancher-desktop is used with containerd, docker must be replaced by nerdctl.

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

2. Start container:

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

The collector can be configured using environment variables. For example:

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

For more on configuration, see: Collector configuration

To save the collector settings persistently, the configuration folder in the container must be mapped to a folder in the host system.

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

4.3 Configuration

4.3.1 Command Line Argument

The command-line arguments use default values that can be configured with environment variables, but can be overridden via CLI.

run Command

Parameter	Beschreibung	Default
--console-log-level	Restricts the log level of the console to the level	COLLECTOR_CONSOLE_LEVEL; 'info'
--log-dir	Sets the directory where the log files are stored. The collector creates the directory if it does not already exist. The collector must have write permission on the folder. Relative paths are allowed and are relative to the 'Working directory'.	COLLECTOR_LOG_DIR; $working-dir/logs
--working-dir	The execution directory of the collector. The collector creates files according to the directory.	Linux: COLLECTOR_WORK_DIR;'.' Docker: COLLECTOR_WORK_DIR;'/opt/ite-si/collector/tmp/' Windows: COLLECTOR_WORK_DIR;'%APPDATA%\iTE-SI\Collector\'
--config-dir	The configuration directory of the collector. The collector creates the directory and empty configuration files if it does not exist.	Linux/Windows: COLLECTOR_CONFIG_DIR;$working-dir Docker: COLLECTOR_CONFIG_DIR;'/opt/ite-si/collector/etc/'

migrate-v1 Command

Migrates a Collector v1 configuration file (config.toml) into a equivalent Collector v2 config files. Further information can be obtained using the following line in the shell.

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

4.3.2 Environment Variables

Variable	Beschreibung
COLLECTOR_CONSOLE_LOG_LEVEL	Specifies the default value for the console log level when the collector is started.
COLLECTOR_LOG_DIR	Specifies the default value for the log directory.
COLLECTOR_WORK_DIR	Specifies the default execution directory.
COLLECTOR_CONFIG_DIR	Specifies the default directory for the configuration files (service.toml, objects.toml)

4.3.3 Configuration file

The collector uses two configuration files: service.toml and objects.json. The configuration files are created if they do not exist. The 'service.toml' should be edited manually, the objects.json should **not** be edited by the user. This file is modified on the fly.

service.toml

Sample Configuration

```
version = 1

[general]
hideConsole = false

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

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

Description of the fields

Section	Field	Type	Optional	Description	Default
	version			Internal format of the json file. Used for automatic upgrade in case of format changes.	1
general	hideConsole	bool		Hides the window in Windows.	false
http	port	uint16		The REST API is started under this port.	8001
http	token	string	true	HTTP bearer token for authentication of the REST API. Can be omitted in order to.	
http.certificate			true	HTTP Zertifikat zur Verwendung von ssl/tls in der REST API.	
http.certificate	certificateFile	string		Path to the HTTPs certificate in PEM format. Relative to the configuration directory.	
http.certificate	keyFile	string		Path to the HTTPs private key in PEM format. Relative to the configuration directory. File must not be password protected.	

4.4 Release Notes

4.4.1 v2.4.0 (05.04.2023)

Features

- Added global JSON output settings (MQTT output symbols)
- Automatically generate certificates for OPCUAConnection

Fixes

- Handling influx urls with appropriate set path segment, e.g. https://myhost/influxdb
- Connect correctly to MQTT broker using TLS
- HTTP REST fixes for OPCUAEventListener symbol (query, patch)
- Minor issues during shutdown procedure

4.4.2 v2.3.0 (01.12.2023)

Features

- Added name query parameter to /connections and /symbols endpoints
- Implemented new Symboltype OPCUAEventListener

Fixes

- introducing (default) HTTP connect timeout 5 seconds to speed up influx failure detection
- conversion of OPC UA boolean values to internal data structure
- handle mass update for OPCUASubscription and deadband gracefully
- do not lookup localhost hostname for resolved OPC UA url
- use IPv4 Addresses when replacing the OPC UA url
- Influx1Connection did not respect failover timeout when creating the database
- respecting the debugEventloop flag correctly
- reconfiguring the influx connection URL do not prevent the influx connection to fail to restart
- prevent crash on browsing UA server capabilities on python asyncua server

4.4.3 v2.2.6 (22.08.2023)

4.4.4 Fixes

- Handle interaction with older Collector App versions gracefully without appearing offline

- Prevent iTE-SI_Collector migrate-v1 subcommand from crashing
- Restart OPCUAConnection when authentication or certificate settings have changed
- Prevent crash on invalid configured certificate for OPCUAConnections

4.4.5 v2.2.5 (02.03.2023)

Fixes

- Add missing license status to connection objects

4.4.6 v2.2.4 (15.02.2023)

Features

- Added [log] flushOnMessage in service.toml to enforce logging of all messages
- Added [general] debugEventloop to service.toml for improved debugging capabilities if necessary
- Improved type information retrieval (especially for Siemens PLC) custom datatypes

Bugfixes

- Improved REST API performance which prevents Collector-App to falsely assume collector has gone down
- OPCUASubscription respects failover timeout now correctly
- OPCUASubscription sends relative and absolute deadband options correctly to the OPC UA server
- Fixed a bug where connection statistics were not shown in the collector app

4.4.7 v2.2.3 (17.10.2022)

Bugfixes

- Fixed a crash in Collector when the statistics were written into an influx db but connection was down

4.4.8 v2.2.2 (2022-09-22)

Bugfixes

- Crash on Windows Collector when connection is closed by OPC UA server during encrypted connection

- Log file output improved on invalid shutdown/crash

4.4.9 v2.2.1 (2022-09-09)

Fixes

- Installation could fail when a newer version of vcredist.exe was already installed

4.4.10 v2.2.0 (03.08.2022)

Collector supports join operations on Influx1Measurement and MQTTPublishTopicObject.

Info

Collector v2.2.0 can be configured completely with Collector App v2.1.X.

Features

- Configured multiple upstreams in Influx1Measurement and MQTTPublishTopicObject are valid
- Join operates by combining the timestamps on a best-effort basis

Bugfixes

- Internal library upgrades
- Prevent Data Collector hanging during shutdown in rare cases
- Add missing metadata 'Host' for OPC UA BulkRead symbols
- Linear transformation was not applied in MQTTPublishTopicObject

4.4.11 v2.1.0 (14.06.2022)

Introduction of OPC UA BulkRead input symbol and MQTT connection and two MQTT output symbols that write data points in json format to the broker.

Info

Collector App 2.1.X can configure the new symbols. Collector App 2.0.X cannot display or configure new types.

Features

- OPC UA BulkRead symbol
  - OPC UA Server uses trigger mechanism to identify new data point
  - Parallel read of multiple OPC UA nodes into a structure
  - Optional handshake with OPC UA server: collector has read the new datapoint(s), collector has written the new datapoint(s) to the output (Influx and MQTT supported)
- MQTT for writing data points to the MQTT broker
  - Data points are sent using JSON
  - Configuration of MQTT Topics
  - Data points can be sent plainly (MQTTPublishTopicPlain) or as json object including timestamp, status, value and other metadata (MQTTPublishTopicObject)

Bugfixes

- Fixed file structure issues with docker image

4.4.12 v2.0.1 (20.09.2021)

Features

- Upgraded OPC UA stack to support string-based NodeIds in type descriptions

Bugfixes

- Fixed sampling interval issues in OPCUASubscription
- Bumped dependent libraries
- Fixed links information displayed in Collector-App

4.4.13 v2.0.0 (24.06.2021)

- Initial release of version 2.0.

5. iTE OPCUA Browser

5.1 General

The OPCUA Browser allows the Collector app to browse the OPC UA address space. The user can select different OPC UA nodes to be collected.

5.2 Installation

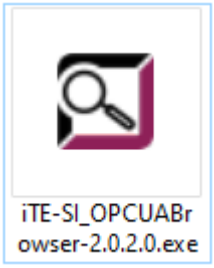
5.2.1 Windows

Download Installer

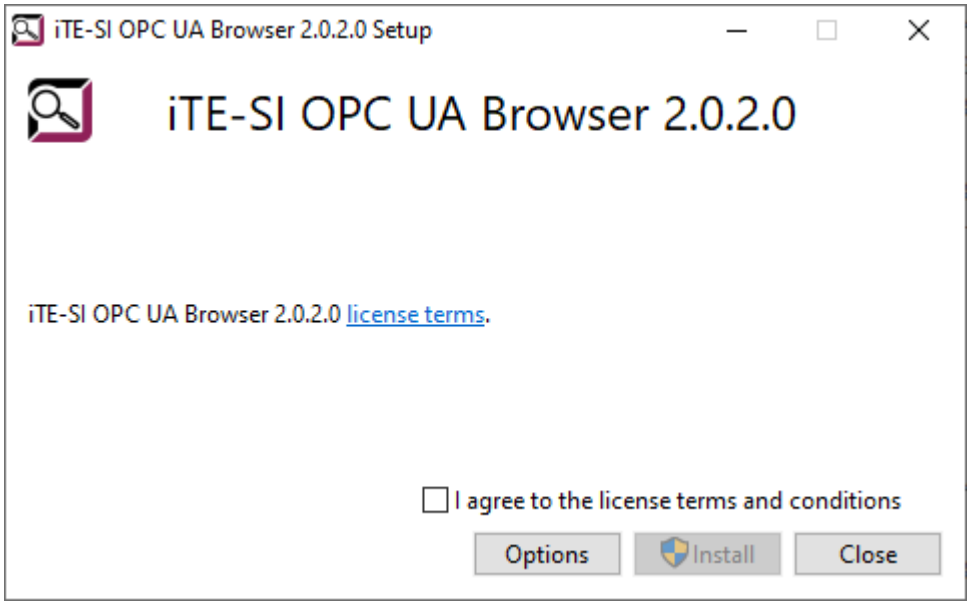
The Windows Installer can be downloaded here: IIoT Building Blocks Downloads

Installieren

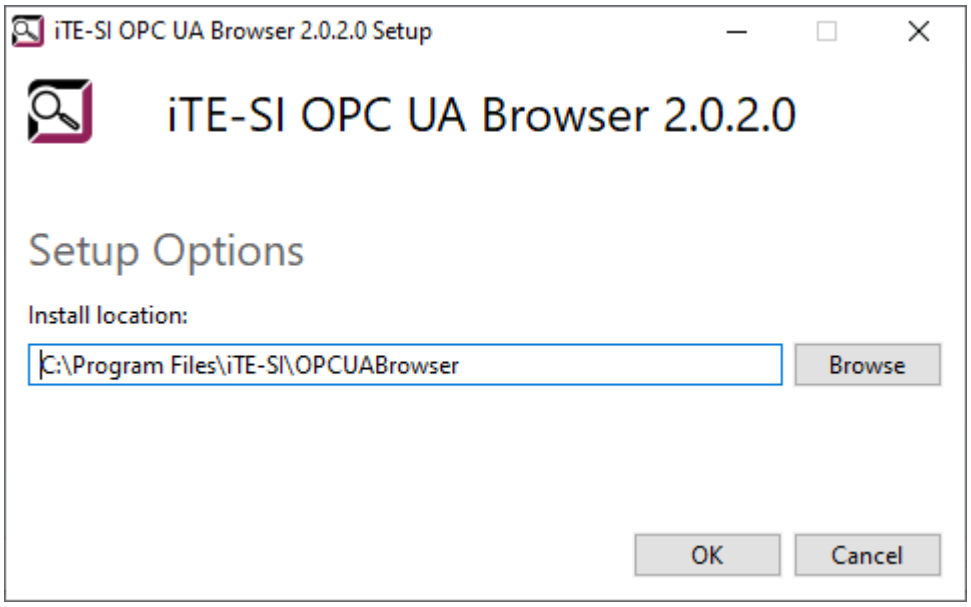
1. Invoke the installer:



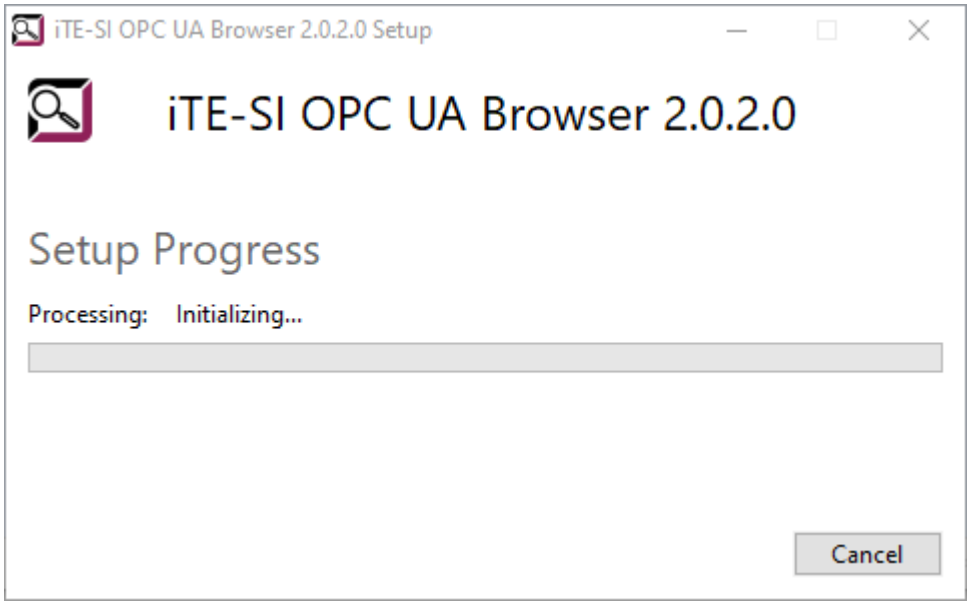
2. Read the End User License Agreement, accept it if necessary and press `Install`. Otherwise press Cancel to abort the installation.



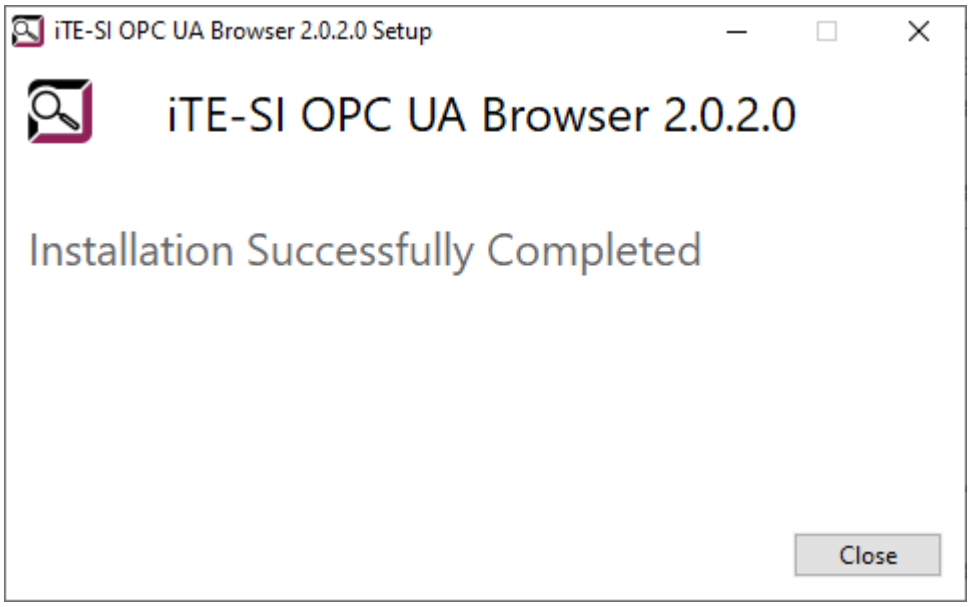
Optional: Select the installation directory in Options.



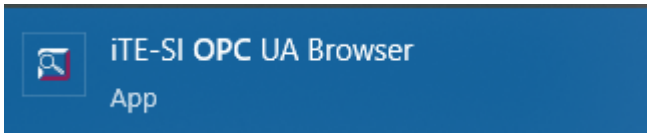
3. Wait a second...



4. The installation has been completed successfully. Press Close.



5. The OPCUA Browser can now be started via start menu.



5.2.2 Docker

The Docker image can be downloaded here: IIoT Building Blocks Downloads

1. Loading the image

Info

x.x.x.x must be replaced by the current version!

Info

If rancher-desktop is used with containerd, docker must be replaced by nerdctl.

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

2. Start container:

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

The browser has differnet configuration options: Browser Konfiguration

To save the browser settings persistently, the configuration folder in the container must be mapped to a folder in the host system.

```
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 Configuration

The OPCUA Browser uses the configuraiton file config.json. The file is located in the docker container at `/opt/ite-si/browser/config.json` and on Windows in `%APPDATA%\iTE-SI\OPCUABrowser/config.json`.

Configuration options:

Field	Type	Description	Default
Http.Port	int	REST Port	8080
MaxNodesPerRead	int	Restricts the number of nodes read in a single OPC UA read request.	Server response
MaxNodesPerBrowse	int	Restricts the number of nodes browsed in a single OPC UA browse request.	Server response
UseSecurity	bool	Default setting for using a secured connection. Can be overwritten in requests	false
ClientPrivateKey	string	The filepath to the OPC UA client key file in DER form.	
SecurityPolicyUri	string	The default security policy URI; use http://opcfoundation.org/UA/SecurityPolicy#None or http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256 if client certificates are set	http://opcfoundation.org/UA/SecurityPolicy#None
MessageSecurityMode	string	The default message security mode.	None
ServerAuthToken	string	The (Bearer) authentification token for the REST API	
ServerCertificateFile	string	The filepath to the server certificate for the REST API in PEM form	
ServerPrivateKeyFile	string	The filepath to the server certificate for the REST API in PEM form	

5.4 Release Notes

5.4.1 v2.2.0 (2024-04-05)

Features

- Automatic creation of certificate for OPC UA connection
- implemented browsing for event fields (OPCUAEventListener)
- Add MaxBrowseDepth parameter for limiting the search

Fixes

- Fix conversion of ByteStrings and NodeIds on json conversion
- Invalid NodeId conversion with type NODEID_TYPE_BYTESTRING

5.4.2 v2.1.1 (2023-09-11)

Features

- Add TestTimeoutInMs configuration for controlling reachability test

Fixes

- correct handling of browser service test (/ or /browse/v1) depending on App configuration
- re-enabled hiding in tray for browser console window

5.4.3 v2.1.0 (2023-08-22)

Features

- Added options "QueryValues, "QueryTypenames","QueryDatatypes" in config.json to prevent attributes being read which may affect some OPC UA servers
- Added optional debugging mode which can be activated using "DebugMode" in config.json

Fixes

- Improve interoperability with older Collector-App using wrong URLs

5.4.4 v2.0.4 (2022-09-27)

Fixes

- Connection with client certificates failed using current Collector App

5.4.5 v2.0.3 (2022-09-09)

Fixes

- Installation could fail when a newer version of vcredist.exe was already installed

5.4.6 v2.0.2 (2021-11-26)

Features

- The installer exe is signed

5.4.7 v2.0.1 (2021-10-12)

Fixes

- Installation could fail with vcredist.exe issues

5.4.8 v2.0.0 (2021-06-24)

- Initial release of version 2.0.