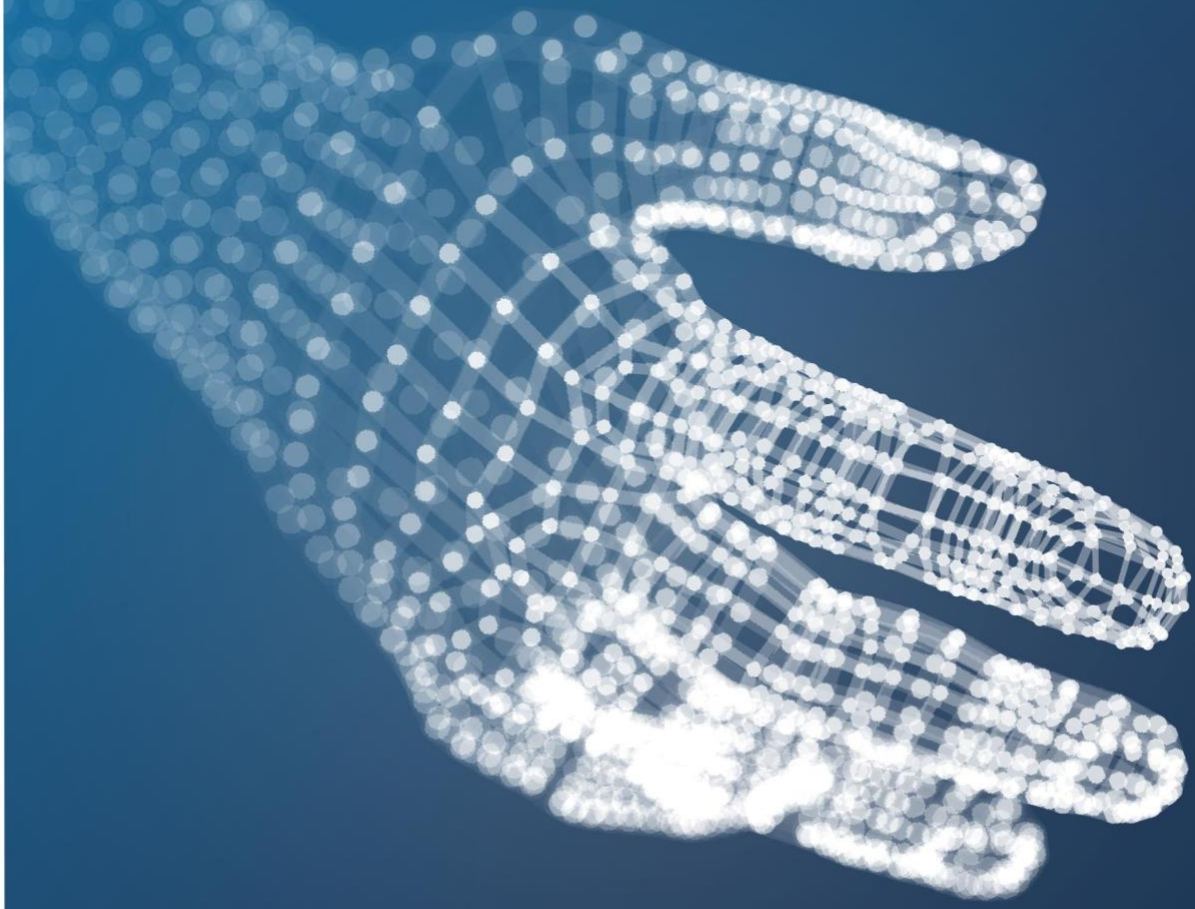
N3uron

Industrial IoT connectivity solutions



REFERENCE MANUAL

DB

INJECTOR

www.n3uron.com

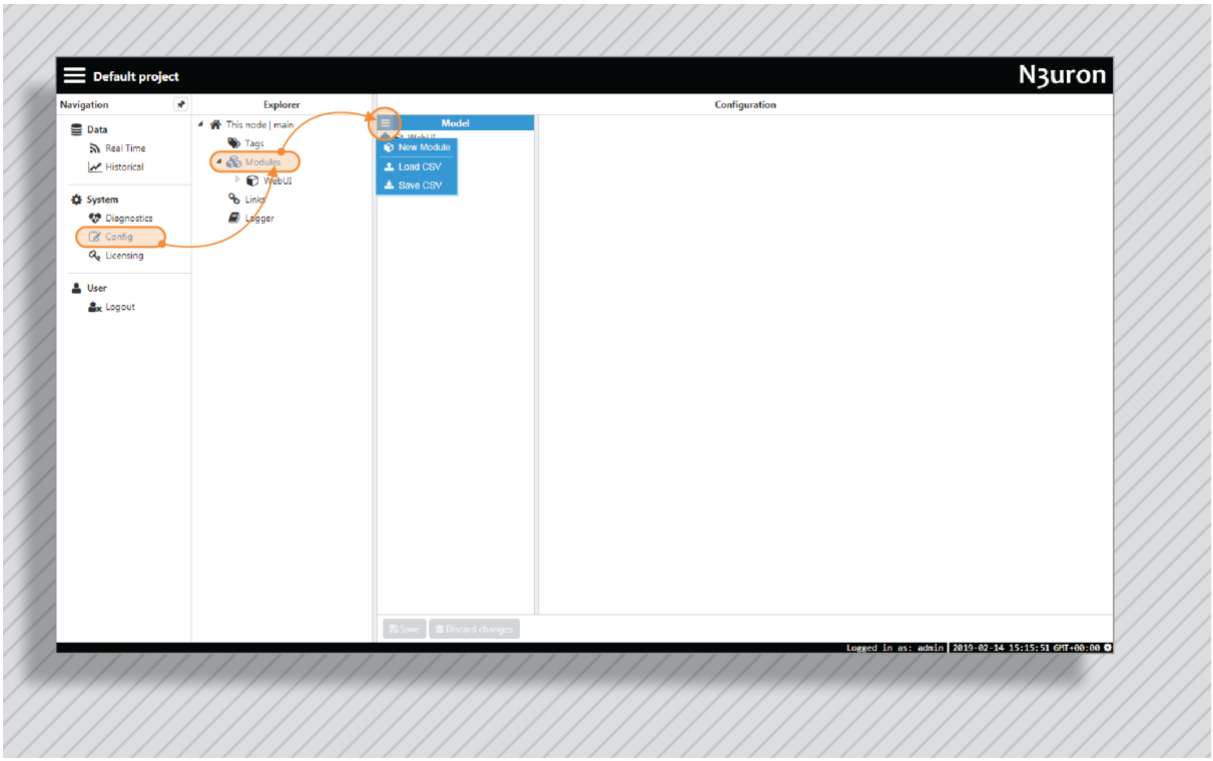
Index

Introduction

DB Injector is used to store tag updates in SQL server tables. It also allows users to partition this data into daily tables, as well as filter tags by their tag paths.

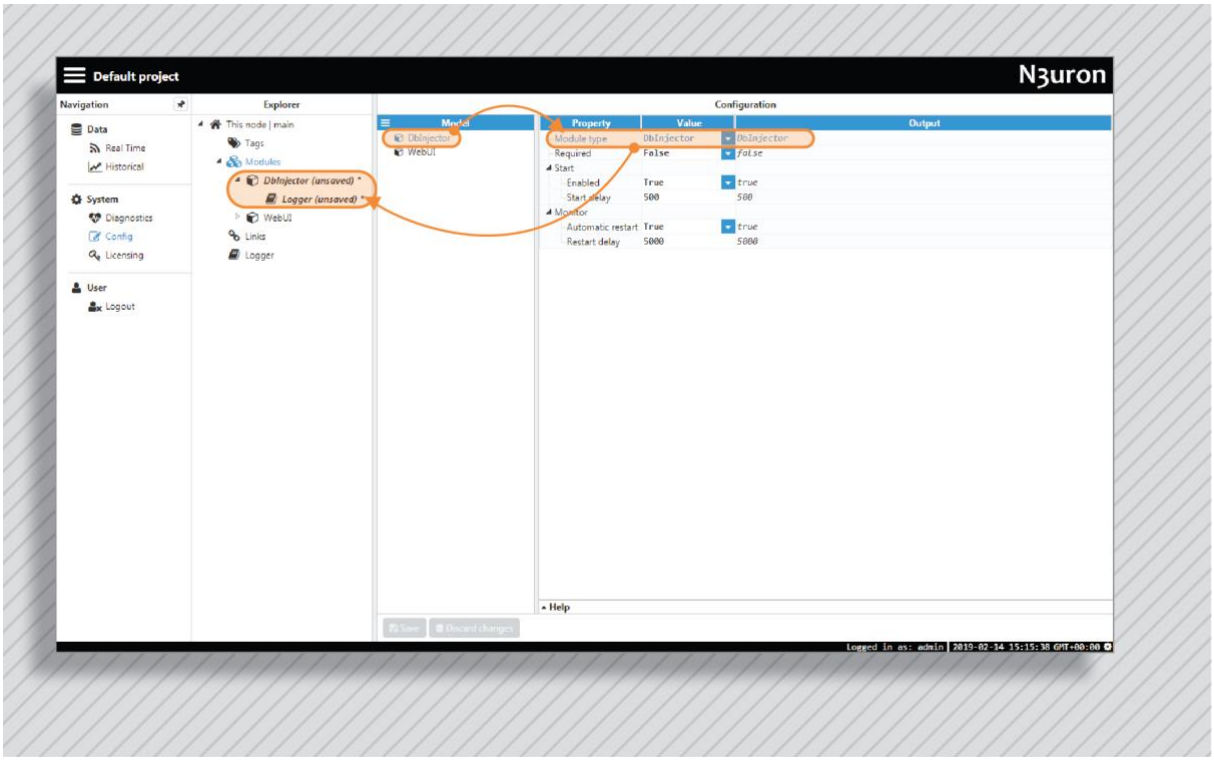
Creating Module Instances

- Open **N3uron** and navigate to the “Config” panel.
- Click on “Modules”, then create a new module. This instance can be given any name (except names with special characters like ‘.’, ‘/’, etc.), although users are recommended to name instances in a similar way to the name of the module being instantiated. In this example, it has been given the name **DB Injector**.



Creating a new module instance.

By setting the module type to **DB Injector**, the new instance will automatically become a **DB Injector** instance. Once saved, **DB Injector** should appear in bold in the module list because there are unsaved changes.



Setting the instance type

In addition to configuring the instance, each module has a Logger which needs to be configured separately. The default settings will be sufficient for this, but users will need to actively open the Logger configuration settings and save the default values in order for the configuration file to be generated.

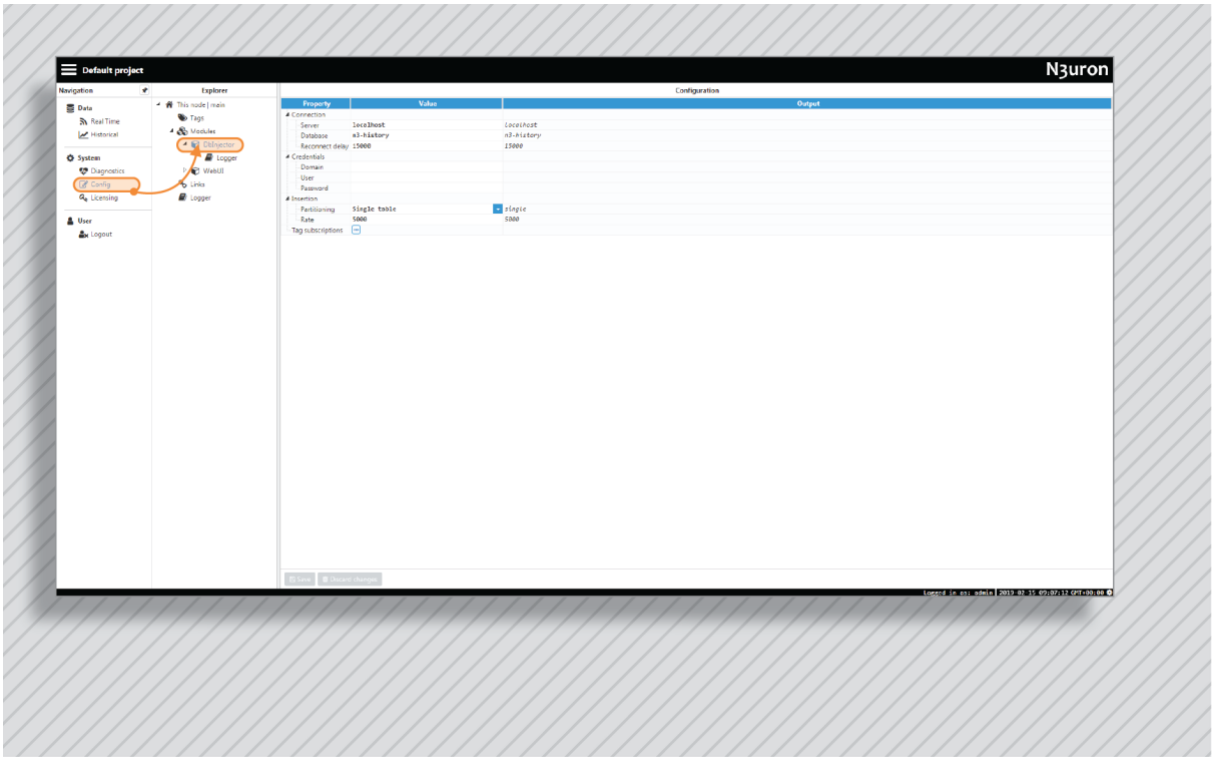
Additionally, each instance can be configured with the following options:

- **Required:** When enabled, all links will be paused when the module is offline to avoid data loss. If not enabled, this module will have no effect on links when offline.
- **Start:** This section controls how the module behaves when the **N3uron** service is started (which also includes service restarts).
    - **Enabled:** If true, the module will start when the **N3uron** service starts. Alternatively, the module must be started manually.
    - **Start delay:** When automatic start is enabled, this setting is used to control how much delay there should be between starting the **N3uron** service and starting the module. This value is displayed in milliseconds.

- **Monitor:** This section is used to monitor the status of each module, as well as to enable it to automatically restart if it goes offline.
    - **Automatic restart:** If true, whenever the module goes offline (except when manually stopped by the user), the module will automatically restart.
    - **Restart delay:** Determines the delay applied before restarting the module after it has gone offline.

Configuration

DB Injector offers the following options:



DB Injector configuration

- **Connection**: These options are for configuring the connection to the destination database:
  - **Server:** hostname or IP of the server where the database is located.
  - **Database:** name of the database where the data will be stored. The database must be created in advance before attempting to connect and must permit CREATE and INSERT operations for any credentials applied to the **DB Injector**.
  - **Reconnect delay:** Delay before attempting a new connection to the server, displayed in milliseconds. Valid range is 1000 to 9000000 milliseconds.

- **Credentials**: These are the credentials used to authenticate the target database:
  - **Domain:** If supplied, **DB Injector** will attempt to use the domain to authenticate the database.
  - **Username:** The username created for authentication. All users must be granted INSERT and CREATE permissions.
  - **Password:** The password created for authentication.

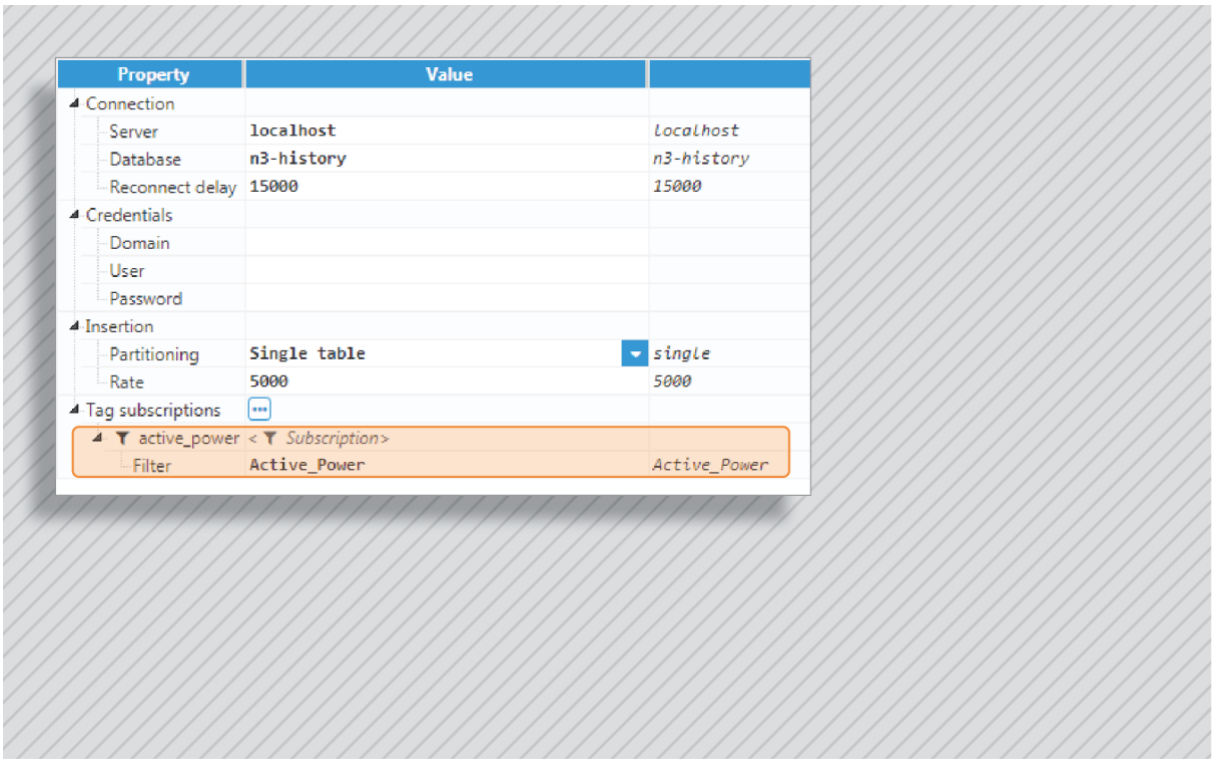
- **Credentials**: These options are used to control the way data is added to the database:
  - **Partitioning:** This defines how data will be saved to the database. This can be set to **Single table**, meaning that all data will be stored to a table labelled data. Alternatively, it can be set to

Daily tables, meaning that data will be partitioned into multiple tables, each table displaying one day of data. Tables will use the following naming convention: YYYYMMDD-data.

- **Rate:** The rate at which data is added to the database, in milliseconds. Data will be buffered to memory before being added. The minimum value is 1000 milliseconds.

Tag subscriptions

Tag subscriptions are used to filter which tags will be saved to the database. If there are no subscriptions, no tags will be saved to the database. The configuration of a tag subscriptions can be seen in the following screenshot:



Tag subscription configuration

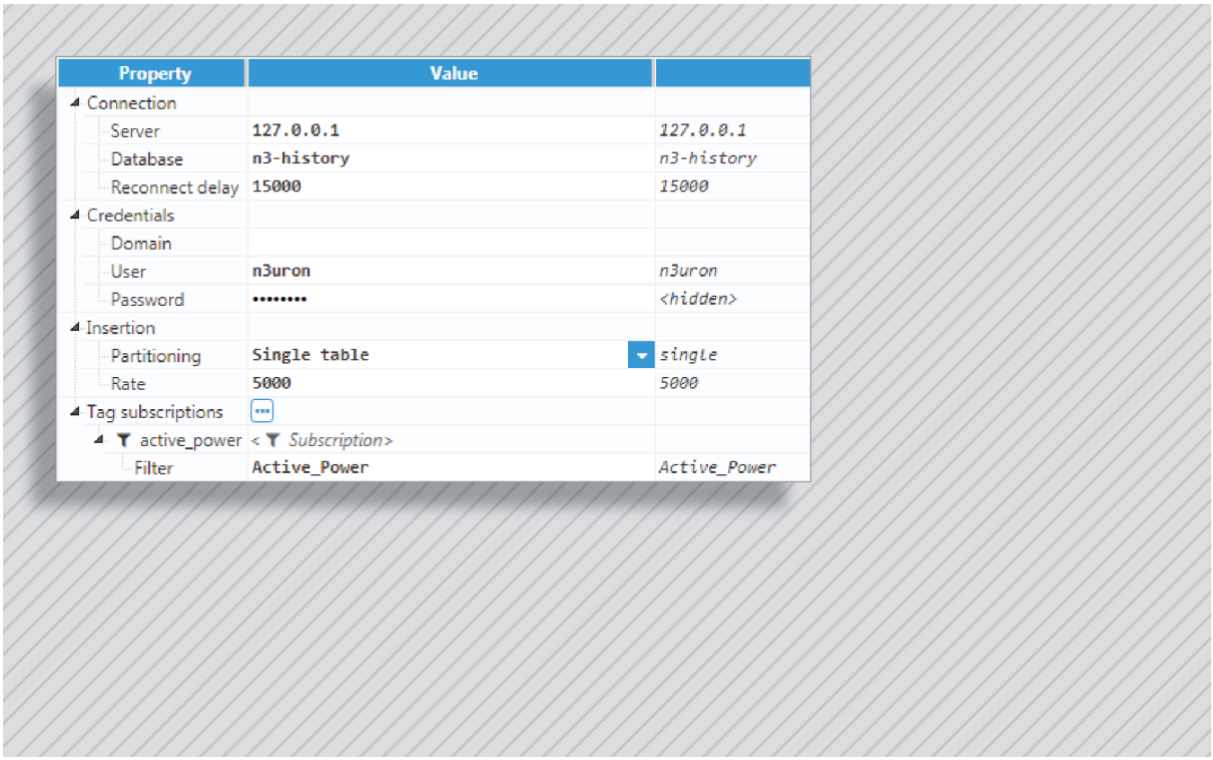
- **Filter:** This is a Regular Expression used to filter tags based on their path. It can use almost any regular expression feature (atomic groups and lookbehind are not supported). This filter is used to check for any matches, although it does not need to match the full tag path. For example:

```
Tag: /Inverter1/Active_Power
Tag: /Inverter2/Active_Power
Tag3: /Inverter3/Active_Power

Filter: Active_Power
Match-> Tag1, Tag2, Tag3

Filter: Invert
Match -> Tag1, Tag2, Tag3

Filter: /Inverter1/Active_Power
Match -> Tag1
```

Examples

SQL Server

- This example displays how to connect a **DB Injector** to a SQL Server using standard authentication, and the default port (1433).
- The first step is to create a database where the data will be stored. In this example, the database is labelled n3-history.
- Create a user in the database with INSERT and CREATE permissions, which will be used to create the necessary tables and enter the data. In this example, the user is “n3uron”.
- In order to save tags to the database, there needs to be at least one tag subscription. In this example, any tag with the name “Active_Power” will be saved.
- The following configuration should be used:

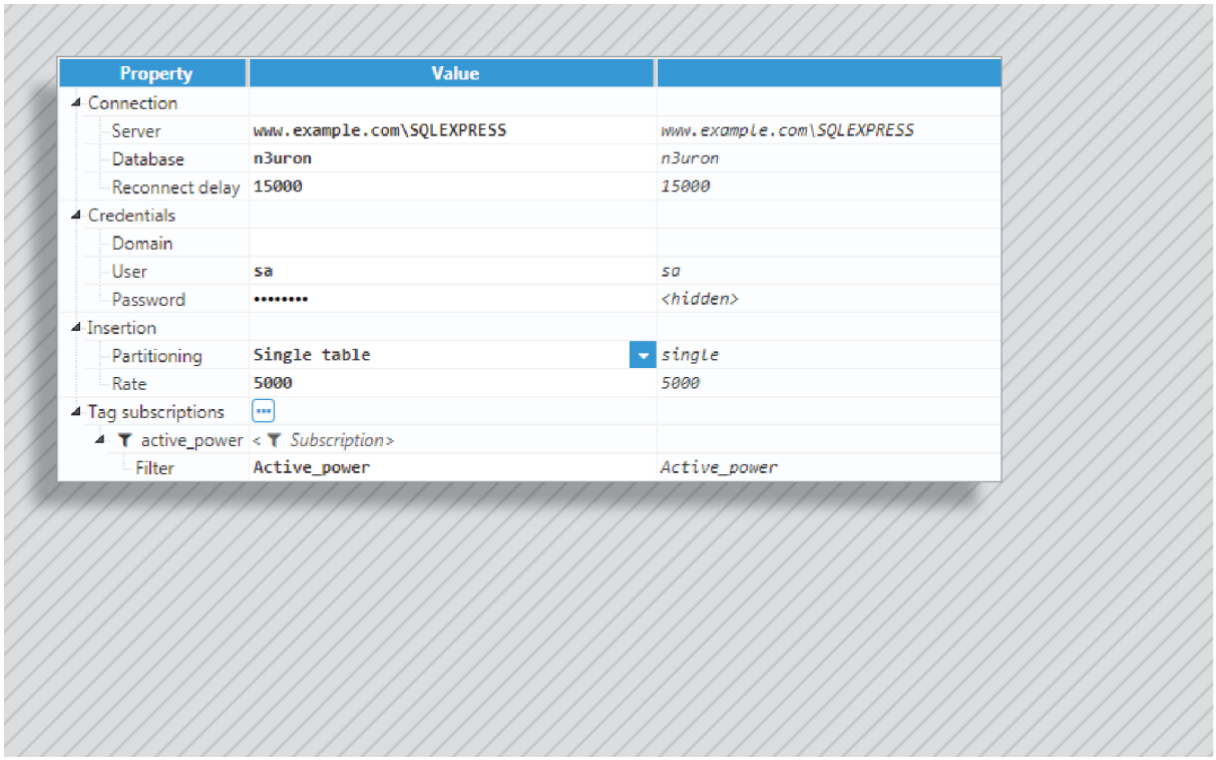


SQL Server configuration example

SQL Server Express

This example shows how to connect a **DB Injector** to a SQL Server Express instance. The connection will be established using dynamic ports and instance name. This requires the SQL Server Browser service to be running and the UDP port to be allowed through the firewall (the default port is 1434).

- The first step is to manually create the destination database for the data. In this example, the database is called “N3uron”.
- Next, a database user with INSERT and CREATE permissions should be created. In this example, the user is named “sa”.
- There needs to be at least one subscription in order to save tags to the database. In this example, the subscribed tags are any tag with “Active_Power” mentioned somewhere in its path.
- The following configuration should be used:



SQL Server Express configuration example

N3uron

Industrial IoT connectivity solutions

REFERENCE MANUAL

DB

INJECTOR