PostgreSQL Database Connector

PostgreSQL databases

- Name: **PostgreSQL**
- Version: 1.0.0.0
- Protocol: Proprietary
- Interface: TCP/IP
- Runtime: .NET 4.8 (Windows)
- Configuration:
  - Devices / Protocols

Configuration PostgreSQL Via Tag Historian
Configuration Example

Overview

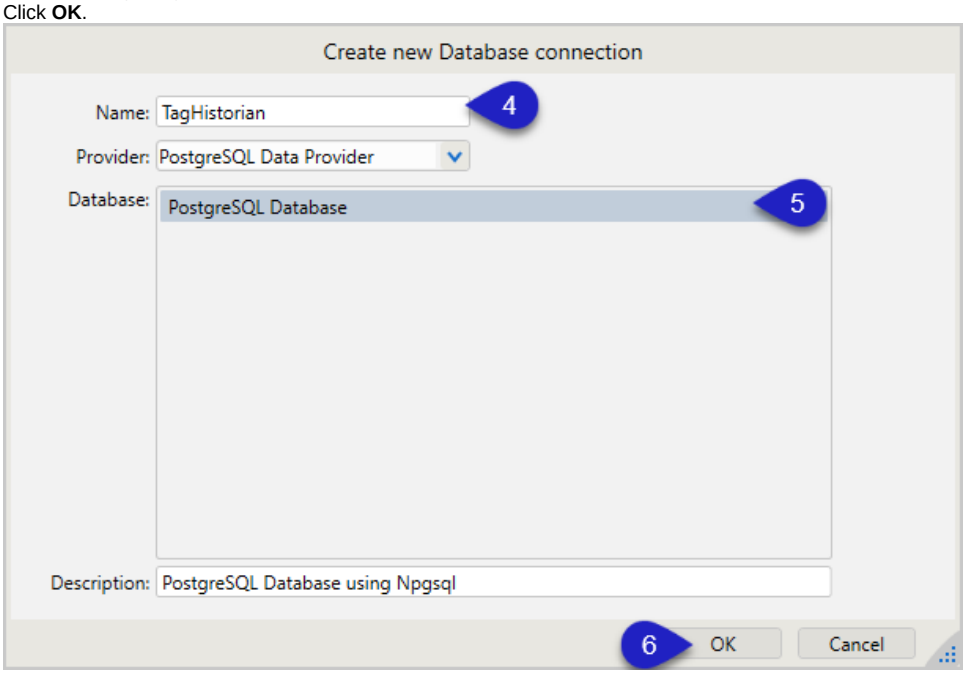
You can use a PostgreSQL database with the Tag Historian feature. The steps to achieve this involve:

1. Rename Databases.
2. Create and Test the PostgreSQL Connection.
3. Configure Tags and Tasks.
4. Write Specific Queries for Select, Insert, Update, and Delete Operations.

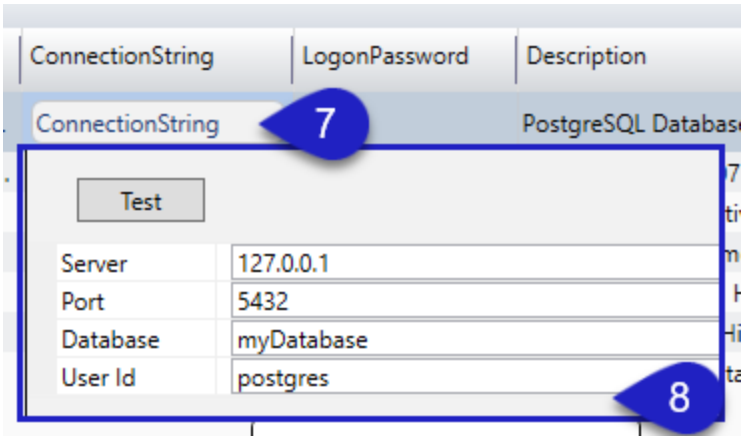
Configuration PostgreSQL Via Tag Historian

Follow the steps below to configure PostgreSQL using Tag Historian.

1. Access **Datasets / DBs.**
2. Rename the SQLite database **TagHistorian** to **TagHistorian2** or any other name. You should rename it because the system will use the **Tag Historian** name to configure the Postgre SQL.
3. In **DBs**, click the **plus icon** to create a new Database Connection.
4. In the **Name** field, enter **TagHistorian**.
5. Choose **PostgreSQL Data Provider** as the **Provider** for the database.
6. Click **OK**.



7. In the data grid, click the **Connection String** column of the newly created row.
8. Configure the connection. For the **Server** field, enter the IP address of the PostgreSQL server. In addition, you need to provide the Port, Database name, and User ID. After filling in all fields, click **Test** to ensure the connection with the PostgreSQL server is working correctly.



Configuration Example

This example presents necessary steps to connect and start using a PostgreSQL DB. To run the example, it is necessary to created objects in other modules of the platform. Below you find a list of modules used and the necessary configurations.

In **Datasets / Queries** you need to create the queries below. Access Datasets Queries Reference to learn how to configure queries.

- **QuerySelect**
- **QueryInsert**
- **QueryDelete**
- **QueryUpdate**

In **Unified Namespace / Tags**, we created some tags to save values in the database. To learn how to create and configure tags, access Unified Namespace. The created tags are described below:

- **Integer01, Integer02, and Integer03** are used to send values to the PostgreSQL database.
- **TriggerUpdate, TriggerSelect, TriggerInsert, and TriggerDelete** are used to trigger script tasks created for the example.
- **QuerySelect** is used to receive the result of the Select query.

In **Historian / Historian Tags**, configure the Integer01, Integer02, and Integer03 tags to **Table1**. Remember that the PostgreSQL connection must be configured to work with Tag Historian, which is standard.

In addition to the above configuration, you need to access **Scripts / Tasks** and create tasks for Select, Delete, Update, and Insert, configuring the Trigger column with the marks created earlier. Below, you will find all required scripts:

- **Select Query**

```
@Tag.QuerySelect = @Dataset.Quey.QuerySelect.SelectCommand();
@Info.Trace("Select OK:" + @Tag.QuerySelect);
```

- **Insert Query**

```
int i = @Dataset.Query.QueryInsert.ExecuteCommand();
@Info.Trace("Insert OK:" + i);
```

- **Update Query**

```
int i = @Dataset.Query.QueryUpdate.ExecuteCommand();
@Info.Trace("Update OK:" + i);
```

- **Delete query:**

```
int i = @Dataset.Query.QueryDelete.ExecuteCommand();
@Info.Trace("Delete OK:" + i);
```

After finishing the configuration and creating the scripts, you can run the project and provide different values to the Tags created. Thus, if you access the PostgreSQL Server, you can access the created **Table1.** You can also check that all values you provided are within **Table1** due to the execution of previously created queries.

blocked URL

In this section...
