Data Hub Communication Service

Information on how to set up a TagProvider connection with Data Hub Communication.

Reference **Modules** **UNS** **UI** **Services** Data Hub

The DataHub Communication Service is a distributed data aggregation and communication solution designed for large-scale industrial applications. It enables multiple independent applications to publish and consume real-time data, device states, and alarm events through an optimized, subscription-based architecture.

In summary, this TagProvider Connection enables multiple runtime solutions to share and access tag, device, server, info, and alarm namespaces.

It uses a TCP-based connection, listening on port 30001 by default. Each runtime configures its access through an alias and an optional access key.

Configuration
- Connection Parameters

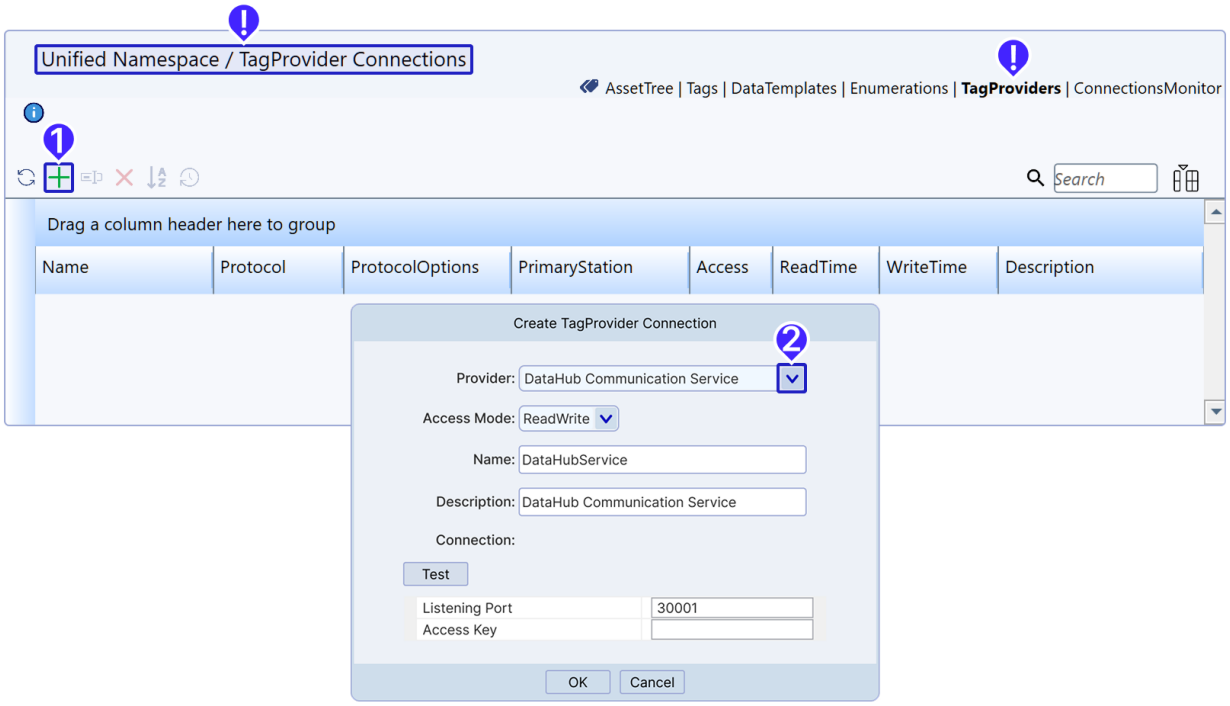
Deployment
- Workflow
- Service configuration
- Clients configuration
- Runtime Startup
- Testing

Architecture example
- Overview
- Getting data from a Historian through DataHub in TrendChart
- Getting data from an Alarm through DataHub in AlarmViewer
- Getting the list of connected solutions to the Datahub
- Configuring databases

Troubleshooting

Configuration

To set up a new TagProvider connection navigate to **Unified Namespace/TagProvider Connections** and press the add button to open the New TagProvider Connection dialog. You can also create your TagProvider Connection directly from the AssetTree. To do this, navigate to **Unified Namespace/Assets Tree** and press  to open the New TagProvider Connection dialog. In the configuration dialog, select the **DataHub Communication Service** on the Provider field and insert the required connection parameters, such as the Listening port and AccessKey. Test the connection by pressing the "Test" button. After that, press the Ok button to finish the provider configuration. Creating the connection adds no folder to the Asset Tree. To use its data with Asset(), open **Unified Namespace / Assets Tree**, select a folder (create one first if needed) and click **Insert data from TagProvider** on the toolbar: the folder linked to the TagProvider is created inside the selected folder. A linked folder cannot sit at the root of the Asset Tree.



Connection Parameters

Field	Description
Provider	Specifies the system the TagProvider will interface with. In this case, it interfaces with the Hub for Server Communication.
Access Mode	Determines permissions for interacting with tags. **Read** allows data reading. **Write** permits data writing. **ReadWrite** enables both reading and writing.
Name	Identifies the tag provider connection. Differentiates multiple connections within the system.
Description	Provides details about the connection. Helps users understand its purpose.
Connection	Shows the parameters used for establishing the connection.
Listening Port	Specifies the port for communication. Directs network traffic through the assigned port.
AccessKey	Manages authentication for the connection. Protects access based on security requirements.

Deployment

Workflow

The DataHub solution deployment follows this:

- **Service Configuration:** Consume data from the specified port, providing real-time access to tags, tag historian, devices, and alarms within Client solutions.
- **Client Configuration:** Publish data to a designated IP and port, allowing Service solutions to access this data.
  For detailed information see Publish to DataHub Service.
- **Runtime Startup**: Both the server and clients must be running. Clients connect and begin publishing data to the DataHub Service.
- **Testing**: On the solution with the configured DataHub Service, navigate to **Unified Namespace/Connection Monitor** and check if the Client data is available.

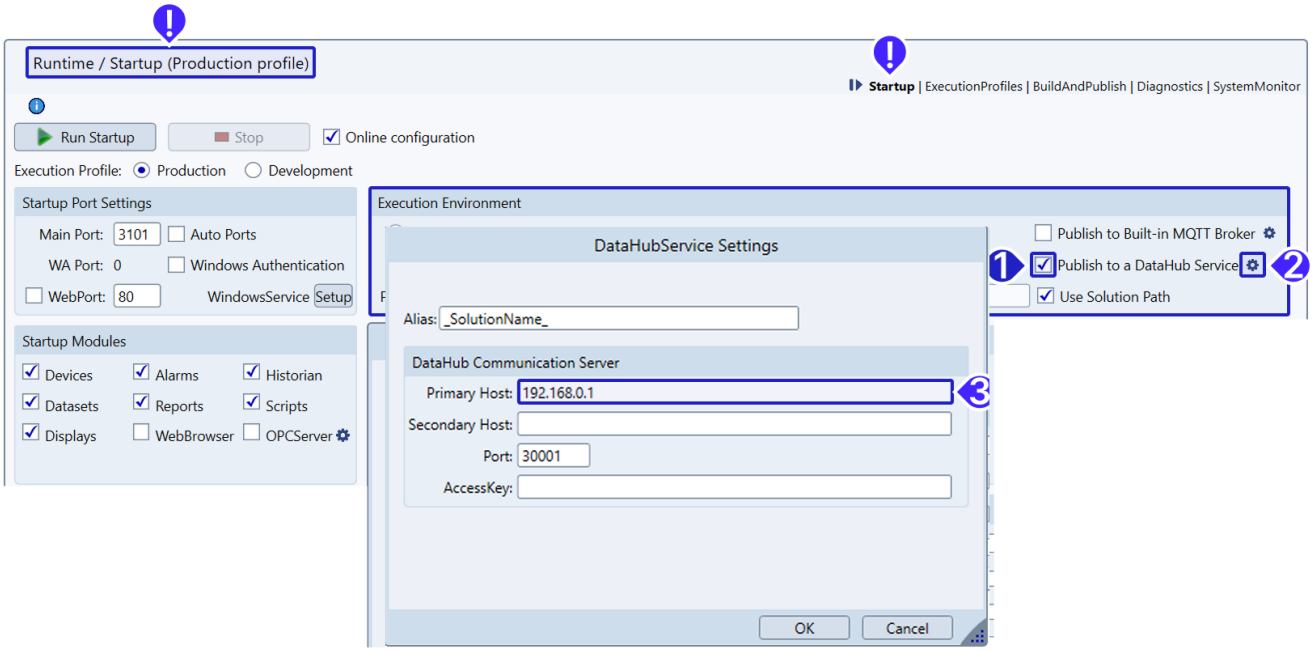
Service configuration

This initial setup allows multiple clients to connect to the DataHub Service. The chosen port stays open so clients can connect at any time, and the service can then access objects in the client solution.
To set it up, go to **Unified Namespace / TagProvider Connections** and create a new connection with the DataHub Service provider, making sure to enter the correct port. The configuration is shown previously in this page.

Clients configuration

This configuration allows each client to send data to the DataHub Service instance you set up earlier. To enable this:

1. Go to **Runtime / Startup** and check the **Publish to a DataHub Service** checkbox in the **Execution Environment**.
2. Click the gear icon to open the **DataHubService Settings** dialog.
3. Assign a name (**Alias**) for the solution, this is how the DataHub Service will identify it.
4. Enter the IP address of the DataHub Service instance (configured in the first step). This ensures the client knows where to send data and can establish a network connection. For example, use 192.168.1.100 for local, x.y.78.210 for cloud, or z.w.0.5 for VPN.

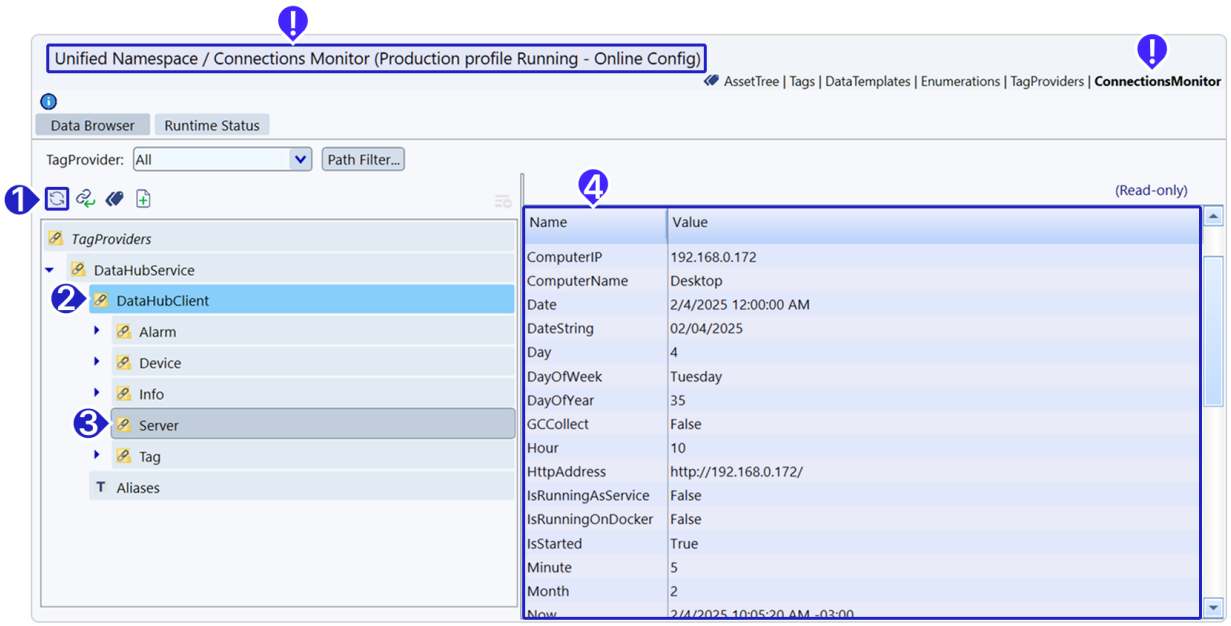


Runtime Startup

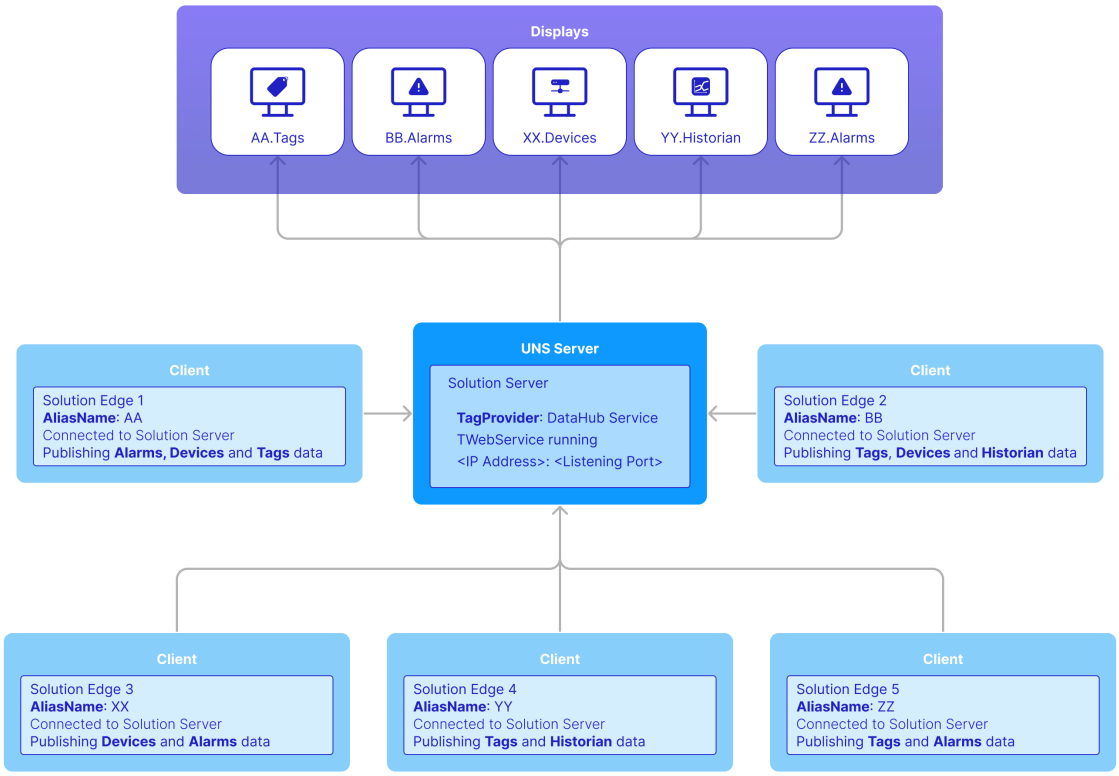
Both the server (DataHub Service) and all configured clients must be running simultaneously for data exchange to begin. Once running, clients will automatically initiate data transmission based on their configured settings.

Testing

The testing stage involves verifying that the DataHub Service receives data from all configured clients correctly. This is done by accessing the **Unified Namespace / Connection Monitor** and refreshing the connections tree, which allows for real-time data transmission. If client data appears in the monitoring interface, it confirms that the system is operating correctly and that communication between clients and the DataHub Service is functional.



Architecture example



Overview

- 1 User Interface solution
- 5 EDGE solution

The UI solution will be only to consume data from EDGE and show the data to users in Displays. The displays can be accessed from multiple places, but it will be only 1 solution.

The EDGE solutions handle tasks such as device management, alarm generation, and data storage in the TagHistorian.

In this architecture EDGE solution 1 can send Alarms, Devices and Tags values to the UI. EDGE solution 2 can send Tags, Devices and Historian to the UI. EDGE solution 3 can send Devices and Alarms to the UI...

When creating the TagProvider (within the UI), it stays in listening mode on the default port 30001. This brings a significant advantage: if you need to add a new EDGE solution to the architecture, you only need to update the display. The objects from the new solution tree will appear automatically without requiring additional configurations when the EDGE solution start publishing data.

In summary, the solution with the TagProvider(UI) can focus solely on the display part, saving resources, while all data processing can take place across multiple EDGE solutions. This provides a significant advantage as the solution becomes decentralized and scalable over time. The only limitation will be processing power and memory, which can be upgraded along with the architecture by simply adding new solutions.

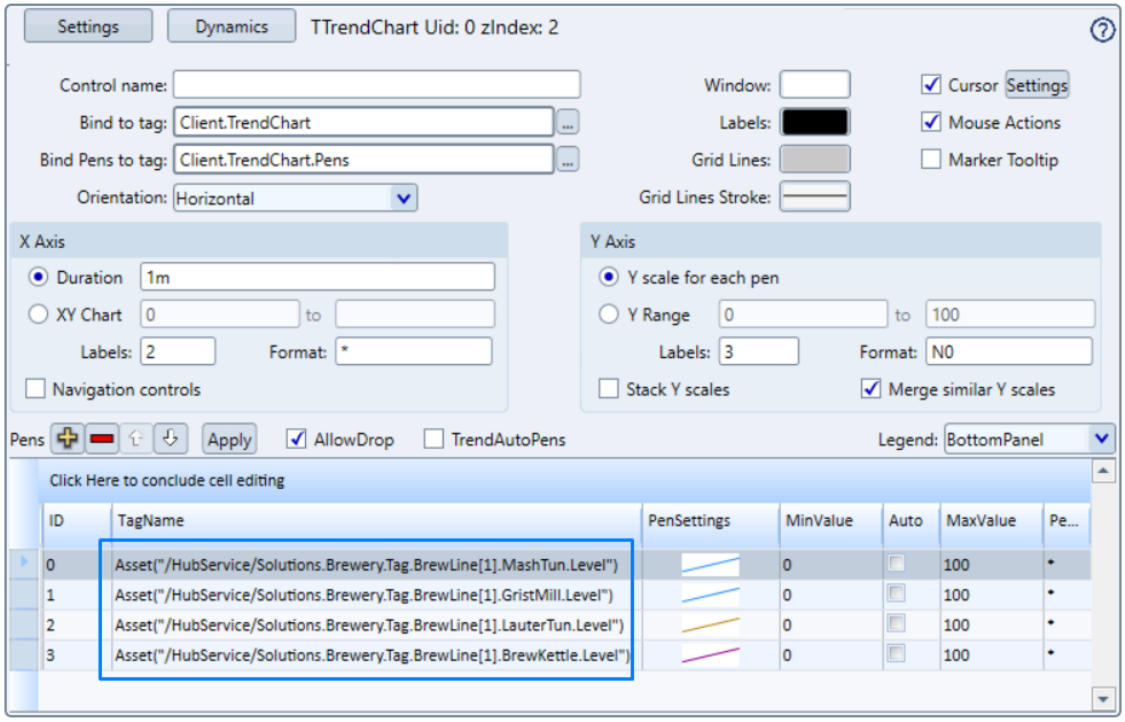
Where:

EDGE (Client): Multiple solutions responsible for the heavy processing, devices, alarms, tags, etc.
UI 'User Interface' (Service): Solution used solely to display information on the screen.

Getting data from a Historian through DataHub in TrendChart

Here you just need to add the Asset path to the tag, when adding a new pen to the Trend you can browse the tag remotely.

With this tool, you can access the EDGE solution Historian and plot current and historian data on the TrendChart within the UI solution.

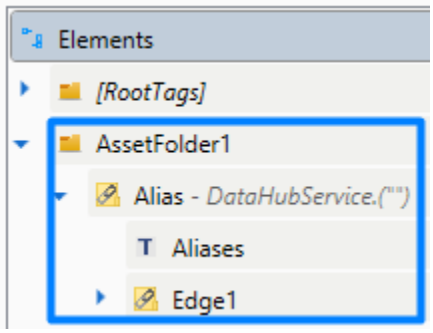


Getting data from an Alarm through DataHub in AlarmViewer

The solution offers full integration with EDGE alarms. Whenever an alarm is acknowledged or its status changes in either environment (EDGE or UI), the update occurs automatically in both, ensuring reliable information.

Two configuration items are important for DataHub:

Asset: You must follow this syntax "/<AssetPath>.<AliasName>" for example: /AssetFolder1/Alias.Edge1. You can check the path in Unified Namespace > Asset Tree. This is the setting that establishes the connection with the **AlarmOnline** edge.



SQL Table: Will be the Table name that was set in **Alarms / Global Setting / Alarm Logging Database / Table Name** in the edge solution.

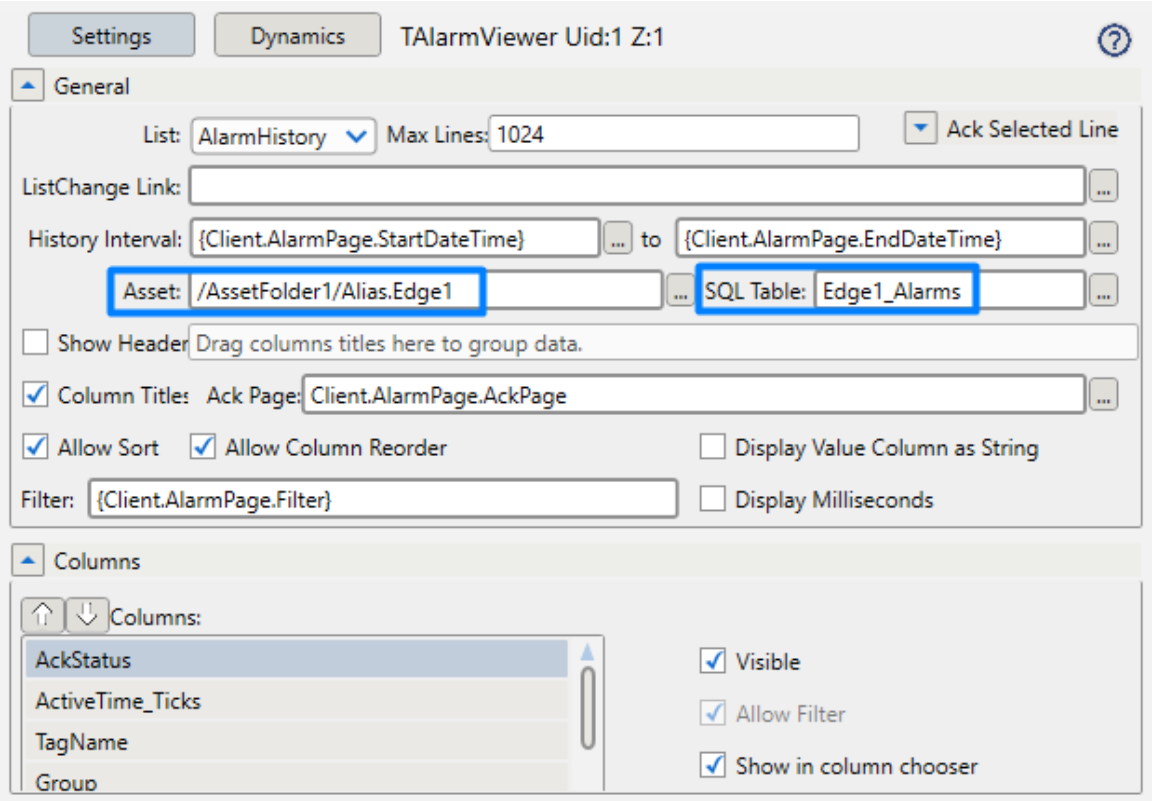
To be more clear about the **SQL Table**, this is a 2 step configuration:

1. Configure all edges to point to the same AlarmHistorian database. This setting is applied in **Datasets > DBs > AlarmHistorian**. By doing so, all edges will use a shared database connection, ensuring that the UI retrieves historian values from a single, consistent source for every edge.
2. Each edge will create its own table, and the table name can be defined in **Alarms > Global Setting > Alarm Logging Database > Table Name** within the edge solution. This value corresponds to the one configured in the **SQL Table** field.

Tip

In '**Alarms / Global Setting / Alarm Logging Database / Table Name**' if you want to combine the solution name with Alarms you can use this dynamic setting: *_SolutionName__*Alarms

You can always use an dynamic object with {}, for example: {Asset("/AssetFolder1/Alias.Edge1.Alarm.GlobalSettings.HistoryTableName")} or {Tag.Text1}



With those setting you will be getting the alarms from the Edge (Client) solution, you can ack any alarm and it will be updated either in the UI, and in the Client solution.

Getting the list of connected solutions to the Datahub

In a DataGrid you can set the Data Source as a tag with DataTable datatype (in this example it will be @Tag.TableGrid), then you create a button that run the following code:

```
DataTable table = await TK.GetChildrenElementsAsync("Asset(\"/HubService/Solutions\")");
        if (table != null)
        {
                foreach (DataRow row in table.Rows)
                {
                        if (TK.ToBool(row["IsAttribute"]))
                                row.Delete();
                }
        }
        table.AcceptChanges();
        @Tag.TableGrid = table;
```

In this example HubService is a folder of the Asset Tree, and Solutions is the folder inside it that is linked to the root of the Data Hub TagProvider. Replace `/HubService/Solutions` with the **asset path up to (but not including) the project folder**. You can confirm the correct path in **Tools > Object Browser**: navigate to the asset and copy the path only up to the parent level of the projects.

Tip: You can either create different screens for each solution, or you can create just one display and create a button to switch between views. Let's say that in the same AlarmViewer you want to switch between solution A and solution B. You can set the "Remote Solution Asset" as '`/HubService/Solutions.{Tag.CurrentSolution}`' and changing the solution name it will change the alarms it gets.

Configuring databases

When multiple solutions publish to the same database, add a '*SolutionName*' column (type VARCHAR) to the '*TagsDictionary*' and '*Alarms*' tables.

The created tables must be empty. Therefore, if any of these tables already contain data, you must delete the existing records before adding the new columns.

Execution steps:

- Run the solution that connects to the database — this will create the required tables in the database.
- Stop the solution.
- Execute the script provided below.
- Restart the solution.

> **Tip:**
> If the project runs with only the Alarms and Historian modules enabled, no data will be inserted into the database, eliminating the need to delete existing records.

Example query for a PostgreSQL database:

```
DELETE FROM public."TagsDictionary";
DELETE FROM public."Alarms";
DELETE FROM public."<TableName>_BIT";
DELETE FROM public."<TableName>_FLOAT";
DELETE FROM public."<TableName>_NTEXT";
DELETE FROM public."<TableName>_REAL";

ALTER TABLE public."TagsDictionary"
ADD COLUMN "SolutionName" VARCHAR(1024);

ALTER TABLE public."Alarms"
ADD COLUMN "SolutionName" VARCHAR(1024);
```

> Note:
> For databases other than PostgreSQL, adjust the SQL syntax to match the respective database engine's requirements.

Troubleshooting

Enabling and checking logs on both the DataHub Service and client machines can provide insight into potential misconfigurations or errors. It ensures that the deployed solution is functioning as expected and that all configured components are successfully exchanging data in real time. To check the TraceWindow, open it via Tools in the Solution Designer, access it through Run Tools in Runtime/Diagnostics, or click its icon in the Status tab of the Startup popup.

In this section...
