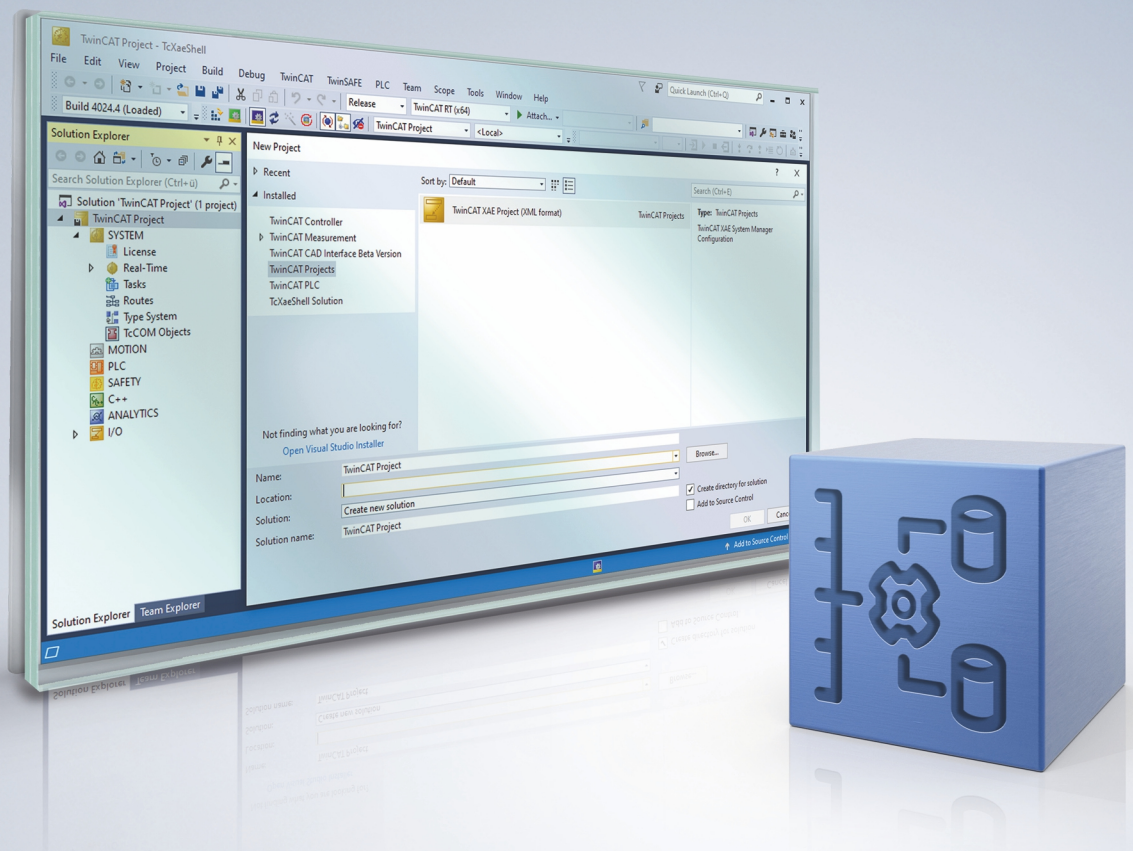
BECKHOFF New Automation Technology

Manual | EN

TF3520

TwinCAT 3 | Analytics Storage Provider



2025-06-11 | Version: 1.6.0

Table of contents

1 Foreword 5
1.1 Notes on the documentation 5
1.2 For your safety 5
1.3 Notes on information security 7
2 Overview 8
3 Installation 10
3.1 System requirements 10
3.2 Installation 10
3.3 Installation under TwinCAT4026 13
3.4 Licensing 13
3.5 Installing the TwinCAT/BSD 15
4 Analytics Workflow - First Steps 17
4.1 Recording data from the machine 17
4.2 Communication 20
4.3 Historicize data 21
4.4 Importing/converting Analytics Files 28
4.5 Analyse data 30
4.6 24h Analytics application 34
5 Technical introduction 43
6 Configuration 45
6.1 Configurator 45
6.1.1 Generic Configurations 45
6.1.2 Additional Configurations 46
6.1.3 Importing/converting Analytics Files 55
6.2 Manager 56
6.2.1 Manager ("Recorder") 57
6.2.2 Data handling with Rule Engine 62
6.3 Working with Historical Data 95
6.4 Console Configurator/Client 99
6.4.1 Configurator 100
6.4.2 Client 101
6.4.3 Batch files for control 104
7 PLC API 109
7.1 Function blocks 109
7.1.1 Topic Architecture 109
7.1.2 FB_ALY_StorageProvider 129
7.2 Data types 140
7.2.1 ST_ALY_SP_Config 140
7.2.2 ST_ConnectionSettings 141
7.2.3 E_CancelType 141
7.2.4 E_ControlMode 142
7.2.5 E_DataStorageType 142
7.2.6 E_RawDataFormat 142

7.2.7 E_RecordingState ... 143
7.2.8 E_RecordMode ... 143
7.2.9 E_ReloadType ... 144
7.2.10 E_RingBufferMode ... 144
7.2.11 E_SetGetHistoricalDataState ... 144
7.2.12 E_StorageState ... 145
7.2.13 E_SymbolMode ... 145
7.2.14 E_PipelineCmdType ... 146

8 Samples ... 147
8.1 PLC Client ... 147

9 Appendix ... 155
9.1 Glossary ... 155
9.2 FAQ - frequently asked questions and answers ... 157
9.3 Third-party components ... 158

10 Support and Service ... 159

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
The documentation and the following notes and explanations must be complied with when installing and commissioning the components.
The trained specialists must always use the current valid documentation.

The trained specialists must ensure that the application and use of the products described is in line with all safety requirements, including all relevant laws, regulations, guidelines, and standards.

Disclaimer

The documentation has been compiled with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without notice.
Claims to modify products that have already been supplied may not be made on the basis of the data, diagrams, and descriptions in this documentation.

Trademarks

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar®, and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of the designations or trademarks contained in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Copyright

Third-party trademarks

Trademarks of third parties may be used in this documentation. You can find the trademark notices here: https://www.beckhoff.com/trademarks.

1.2 For your safety

Safety regulations

Read the following explanations for your safety.
Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation, and drive technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

⚠ DANGER

Hazard with high risk of death or serious injury.

⚠ WARNING

Hazard with medium risk of death or serious injury.

⚠ CAUTION

There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE

The environment, equipment, or data may be damaged.

Information on handling the product

This information includes, for example:
recommendations for action, assistance or further information on the product.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

2 Overview

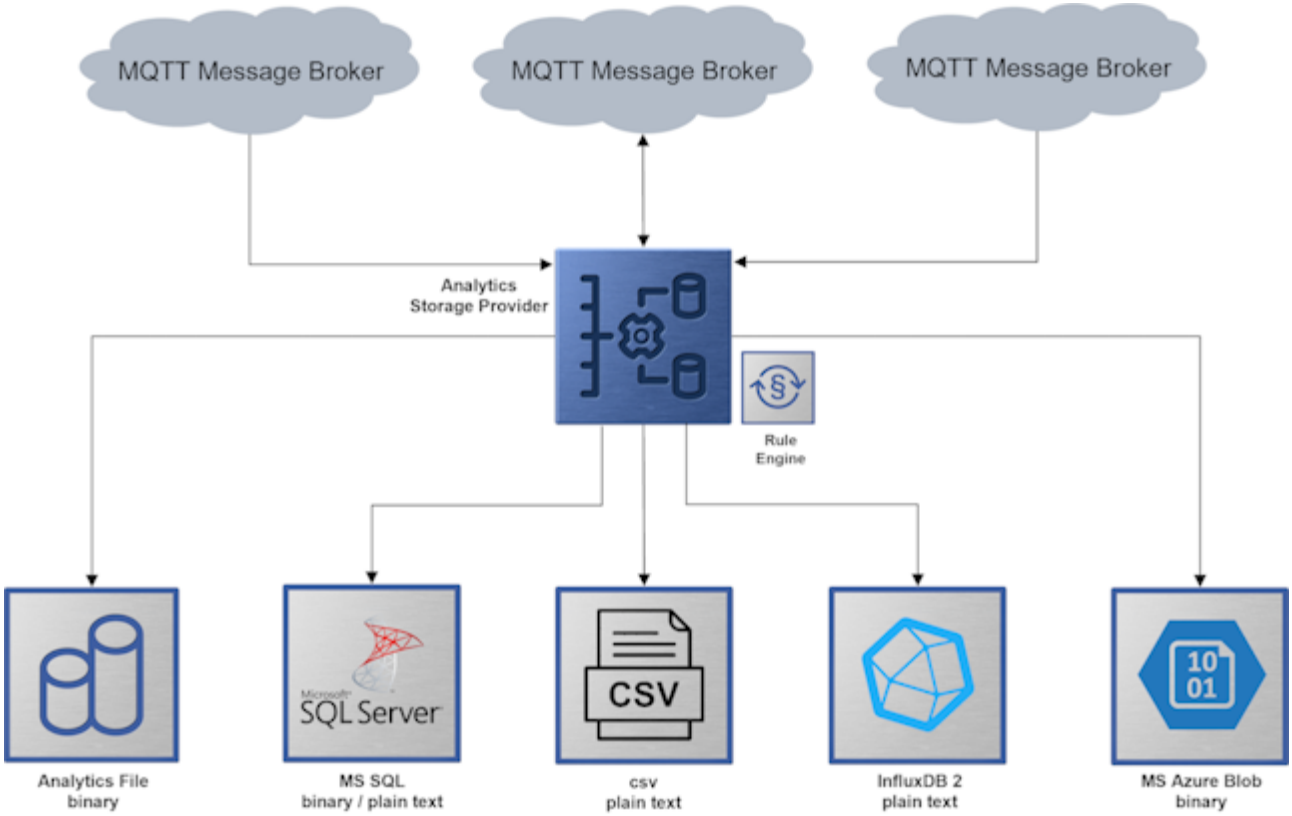
With the TwinCAT Analytics Storage Provider, Beckhoff offers a way to continuously store high-resolution data in a binary format or in plain text. The decisive factor is that the user does not have to worry about data storage. The storage provider takes care of this automatically. The configuration is done with a few clicks in engineering. Complex SQL commands are not necessary. Classic databases can be used, but also binary blob stores.

Components

- TwinCAT Analytics Storage Provider service: A Windows service that manages the communication.
- TwinCAT Analytics Storage Provider PLC Library: A TwinCAT 3 PLC library with functions for controlling the storage provider from a PLC application.
- TwinCAT Analytics Storage Provider Client: A console application with functions for controlling the Storage Provider via the command line.
- TwinCAT Analytics Storage Provider Configurator: An engineering application for configuring the various data sinks and data sources.
- TwinCAT Analytics Storage Provider Manager: An engineering application for managing the recorded data and controlling the Storage Provider.

Principle of operation

The Analytics Storage Provider receives and sends data via MQTT communication protocol. For this purpose, it is connected to a native MQTT message broker in the network and on the other side to the corresponding data sink.



Supported databases/storage

- TwinCAT Analytics Binary File [▶ 48]
- Microsoft SQL (binary format / plain text [▶ 49])
- CSV file [▶ 51]
- InfluxDB [▶ 52]

- Microsoft Azure Blob [▶ 53]

3 Installation

3.1 System requirements

The requirements of the Service and the PLC library of the Analytics Storage Provider can be found in the following tables. It is also possible to install both on one system as well.

Technical data Service	TF3520 TwinCAT 3 Analytics Storage Provider
Target System	Windows 10, TwinCAT/BSD
.NET Framework	.Net 4.5.1 or higher
Min. TwinCAT version	3.1.4022.25
Min. TwinCAT level	TC1000 | TwinCAT 3 ADS

Technical data Library	TF3520 TwinCAT 3 Analytics Storage Provider
Target System	Windows 10, TwinCAT/BSD
Min. TwinCAT version	3.1.4022.29
Min. TwinCAT level	TC1200 | TwinCAT 3 PLC

3.2 Installation

Setup installation (TwinCAT 3.1 Build 4024)

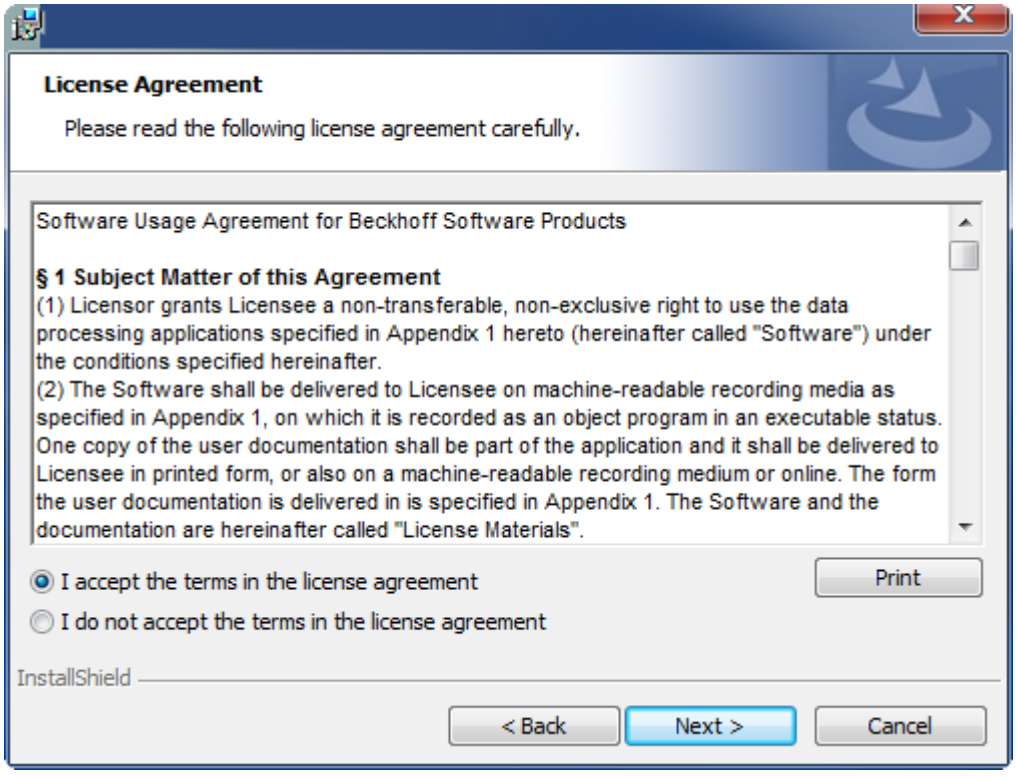
The following section describes how to install the TwinCAT 3 function for Windows-based operating systems.

✓ The TwinCAT 3 function setup file was downloaded from the Beckhoff website.

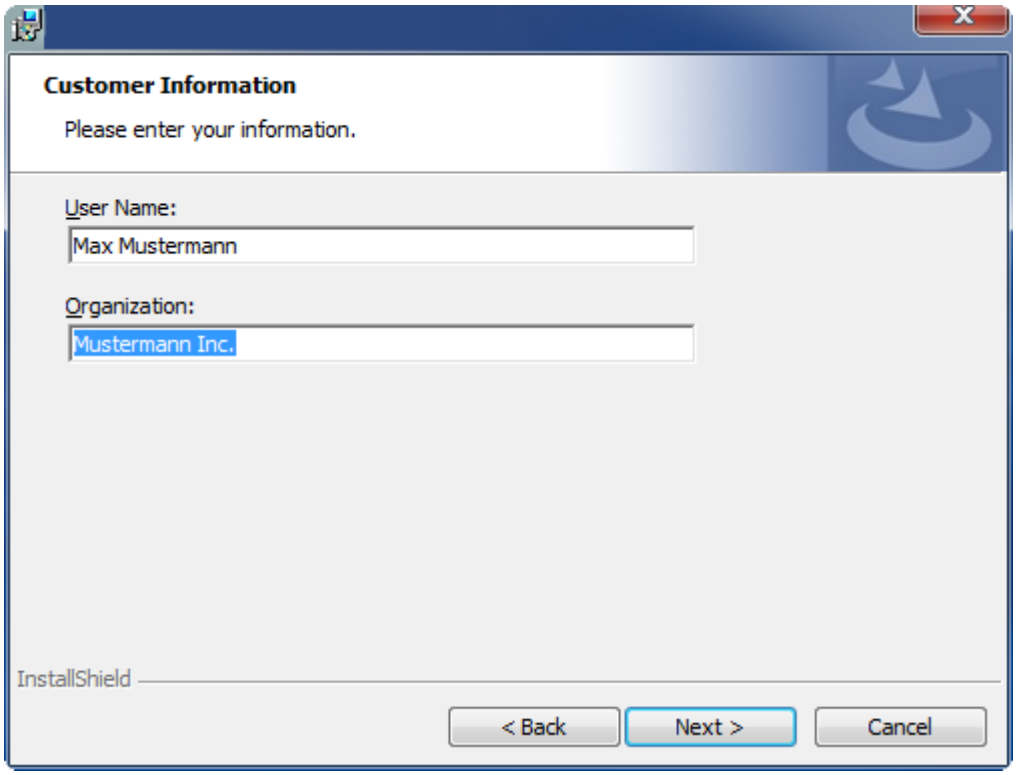
1. Run the setup file as administrator. To do this, select the **Run As Admin** command in the context menu of the file.

   ⇨ The installation dialog opens.

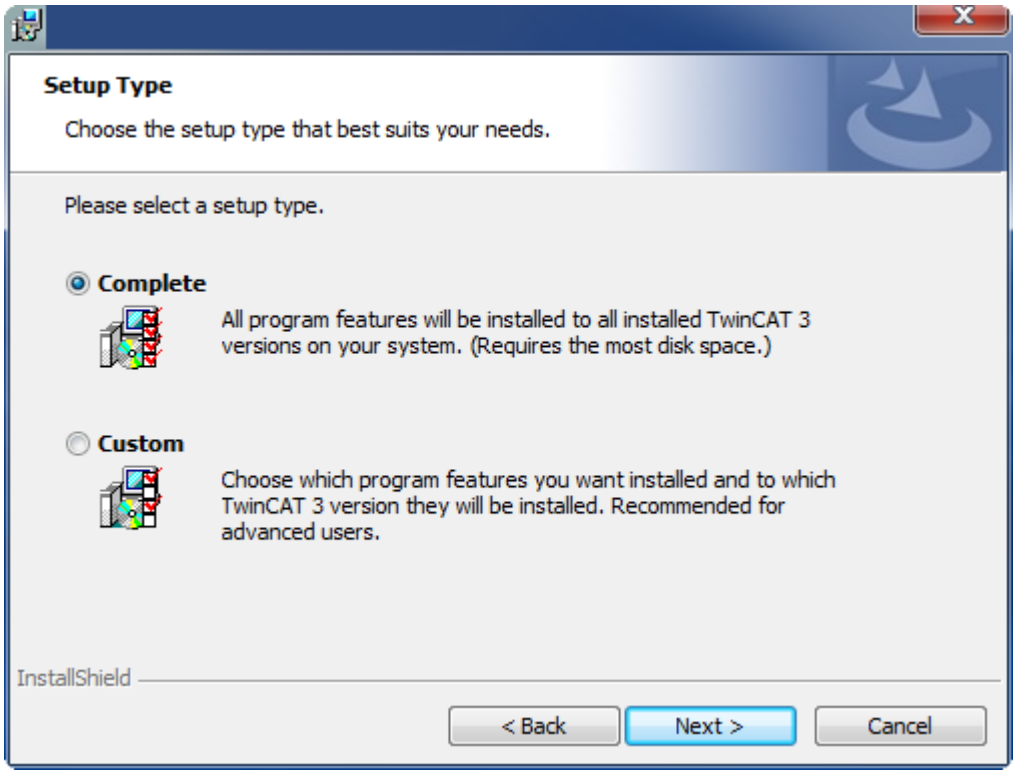
2. Accept the end user licensing agreement and click **Next**.



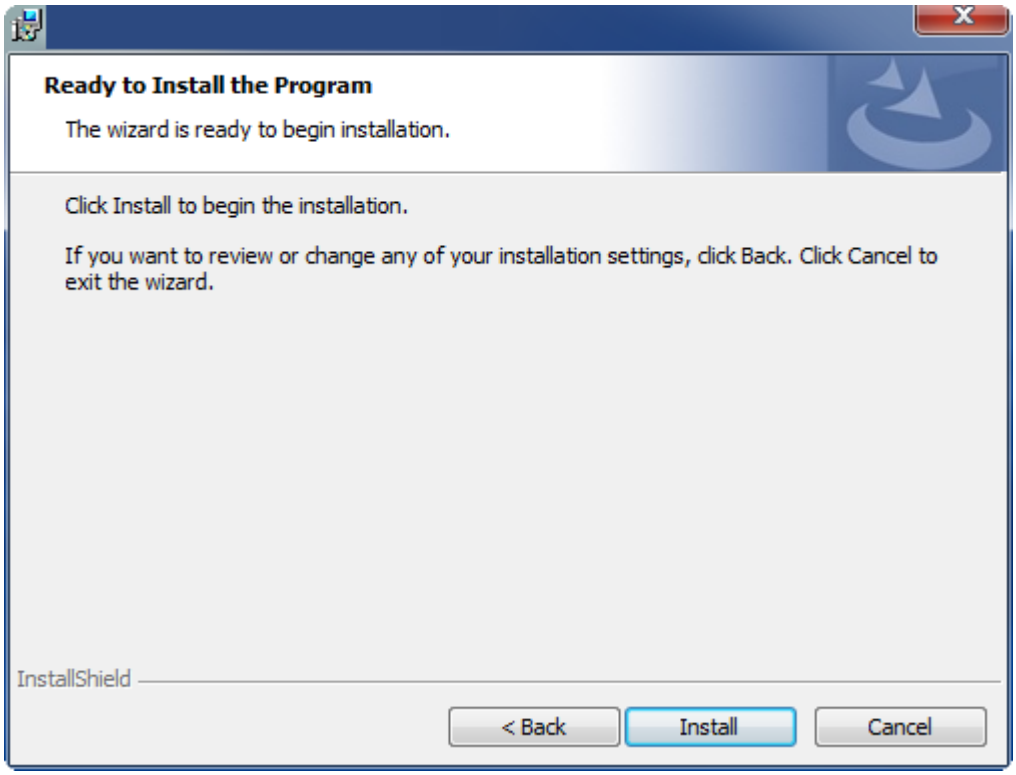
3. Enter your user data.



4. If you want to install the full version of the TwinCAT 3 function, select **Complete** as the installation type. If you want to install the TwinCAT 3 function components separately, select **Custom**.

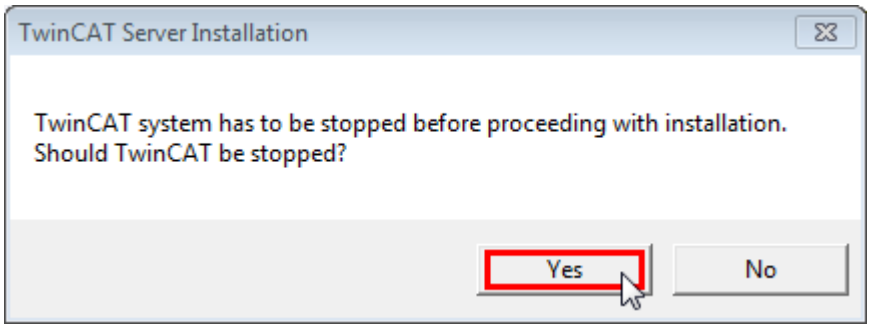


5. Click **Next**, then **Install** to start the installation.

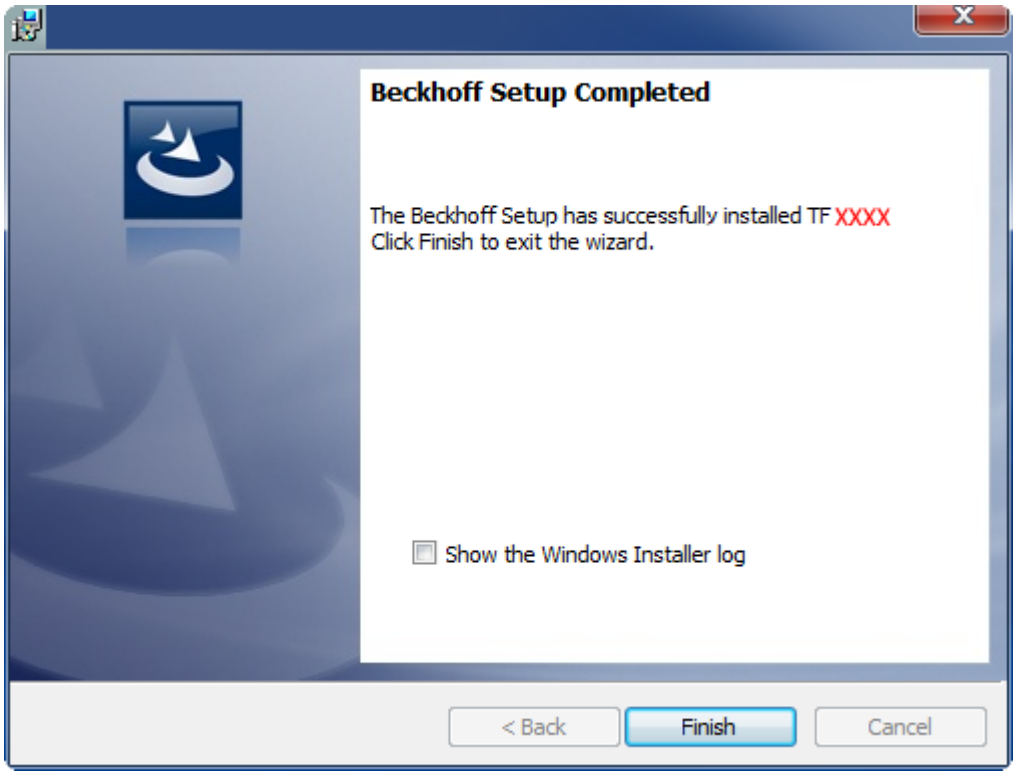


⇨ A dialog box informs you that the TwinCAT system must be stopped to proceed with the installation.

6. Confirm the dialog with **Yes.**



7. Click **Finish** to exit the setup.



⇨ The TwinCAT 3 function has been installed successfully.

3.3 Installation under TwinCAT4026

TwinCAT Package Manager

If you are using TwinCAT 3.1 Build 4026 (and higher) on the Microsoft Windows operating system, you can install this function via the TwinCAT Package Manager, see Installation documentation.

Normally you install the function via the corresponding workload; however, you can also install the packages contained in the workload individually. This documentation briefly describes the installation process via the workload.

Command line program TcPkg

You can use the TcPkg **C**ommand **L**ine **I**nterface (CLI) to display the available workloads on the system:

```
tcpkg list TF3520
```

You can use the following command to install the Workload of the TF3520 TC3 Analytics Storage Provider function.

```
tcpkg install TF3520.AnalyticsStorageProvider.XAE
tcpkg install TF3520.AnalyticsStorageProvider.XAR
```

TwinCAT Package Manager UI

You can use the **U**ser **I**nterface (UI) to display all available workloads and install them if required.
To do this, follow the corresponding instructions in the interface.

3.4 Licensing

The TwinCAT 3 function can be activated as a full version or as a 7-day test version. Both license types can be activated via the TwinCAT 3 development environment (XAE).

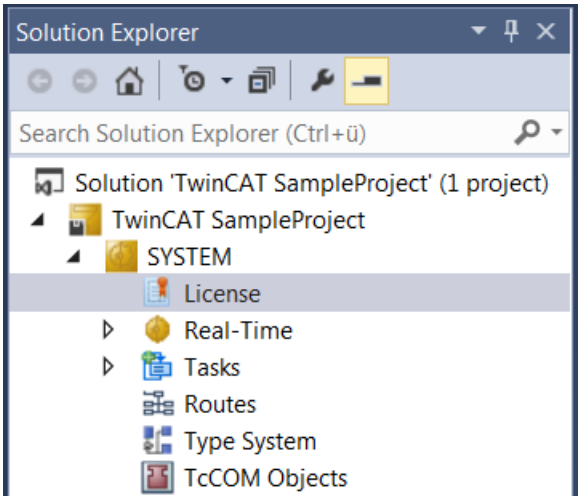
Licensing the full version of a TwinCAT 3 Function

A description of the procedure to license a full version can be found in the Beckhoff Information System in the documentation "TwinCAT 3 Licensing".

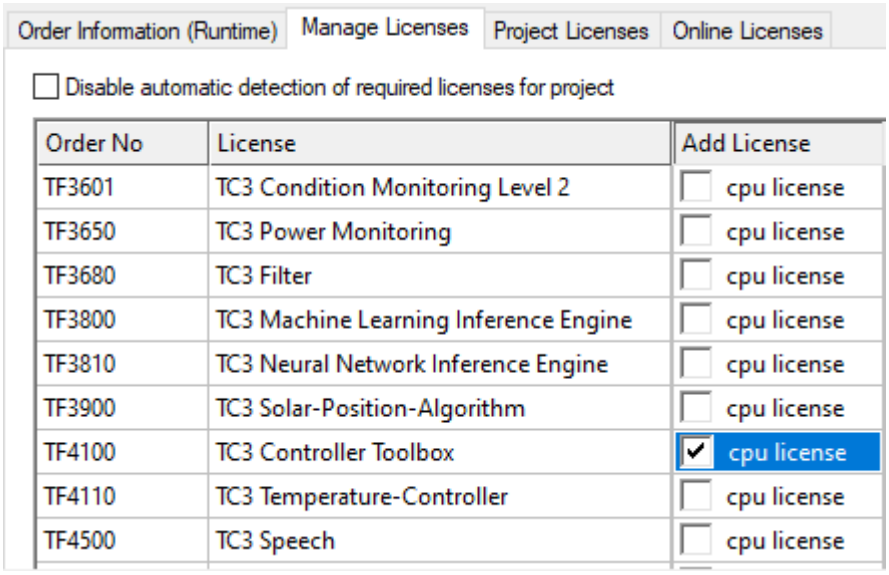
Licensing the 7-day test version of a TwinCAT 3 Function

A 7-day test version cannot be enabled for a TwinCAT 3 license dongle.

1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the toolbar.
   ⇨ The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.
4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.



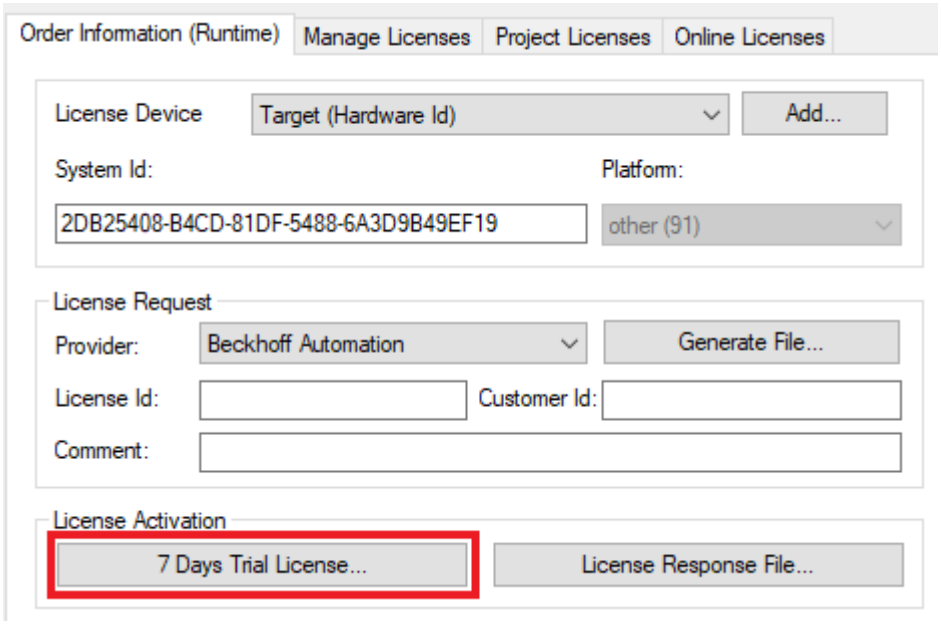
   ⇨ The TwinCAT 3 license manager opens.
5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. "TF4100 TC3 Controller Toolbox").



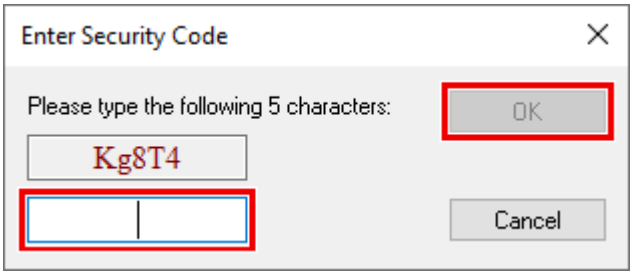
Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	cpu license
TF3650	TC3 Power Monitoring	cpu license
TF3680	TC3 Filter	cpu license
TF3800	TC3 Machine Learning Inference Engine	cpu license
TF3810	TC3 Neural Network Inference Engine	cpu license
TF3900	TC3 Solar-Position-Algorithm	cpu license
TF4100	TC3 Controller Toolbox	cpu license
TF4110	TC3 Temperature-Controller	cpu license
TF4500	TC3 Speech	cpu license

6. Open the **Order Information (Runtime)** tab.
   ⇨ In the tabular overview of licenses, the previously selected license is displayed with the status “missing”.

7. Click **7-Day Trial License...** to activate the 7-day trial license.



⇨ A dialog box opens, prompting you to enter the security code displayed in the dialog.



8. Enter the code exactly as it is displayed and confirm the entry.
9. Confirm the subsequent dialog, which indicates the successful activation.
   ⇨ In the tabular overview of licenses, the license status now indicates the expiry date of the license.
10. Restart the TwinCAT system.

⇨ The 7-day trial version is enabled.

3.5 Installing the TwinCAT/BSD

The TwinCAT 3 Analytics Storage Provider Server is available as a package for TwinCAT/BSD in the package repository. Under the package name "TF3520 Analytics Storage Provider" it can be installed via the following command:

```
doas pkg install TF3520-Analytics-Storage-Provider
```

Further information about the Package Server can be found in the Embedded PC section of the TwinCAT/BSD manual.

After a system restart or restart of TwinCAT, the TwinCAT 3 Analytics Storage Provider Server is also started and can be configured by a client via MQTT.

MQTT port enabling

To use the Analytics Storage Provider and Console Configurator/Client [▶ 99] under TwinCAT/BSD, the corresponding MQTT port must be enabled for communication. For more info see: Port enabling under TwinCAT/BSD

After installation, the Client.dll for the console is located under the path */usr/local/etc/TwinCAT/Functions/TF3520-Analytics-Storage-Provider/Client*.

The Analytics Storage Provider service can be started by the following command if the license is activated:

```
doas service TcAnalyticsStorageProvider start
```

4 Analytics Workflow - First Steps

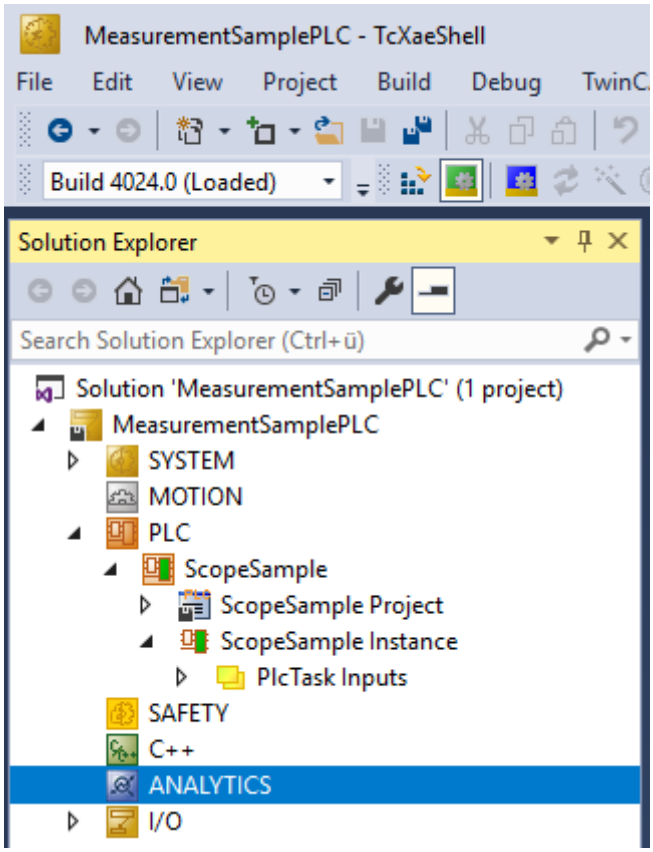
This step by step documentation presents the complete TwinCAT Analytics workflow. From the data acquisition over the communication and historizing up to the evaluation and analysis of the data and to the presentation of the data in web-based dashboard.

4.1 Recording data from the machine

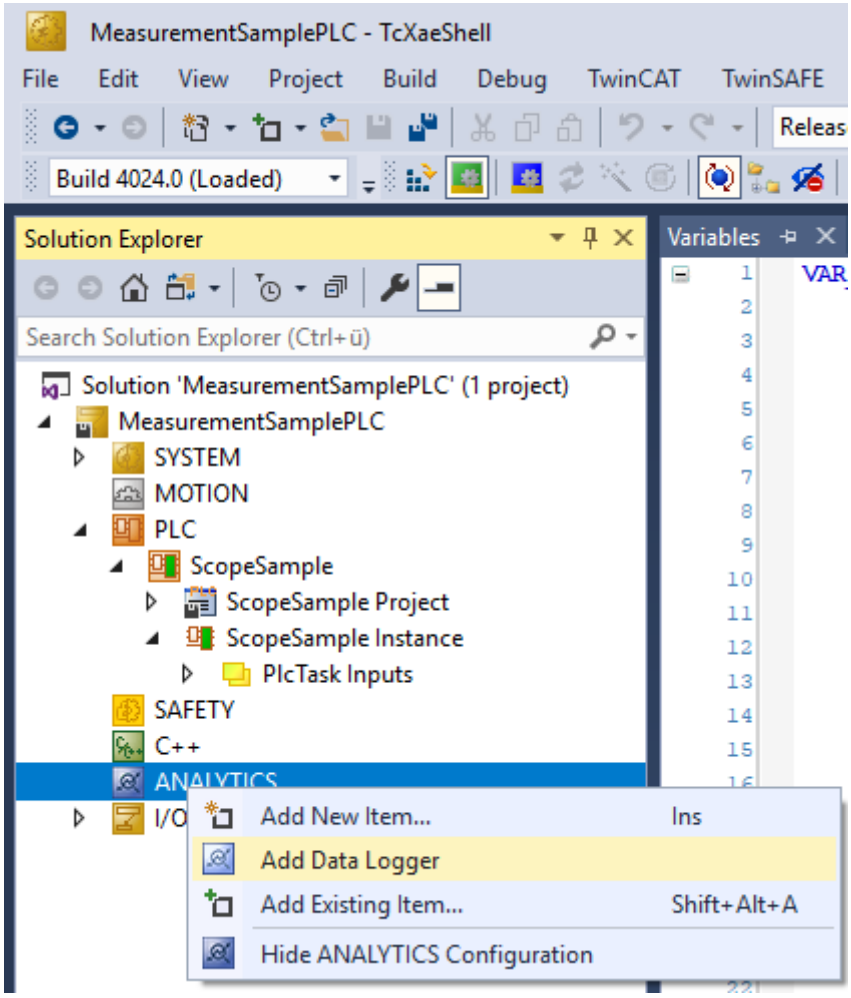
On the machine side is the Analytics Logger the recorder of process data from the machine image, PLC, NC and so on. The Logger is working in the real-time context of TwinCAT.

The TwinCAT Analytics Logger is installed with TwinCAT XAE and XAR. The Logger can act as MQTT Client to communicate the recorded data to a native MQTT Message Broker or store the data in the same data format in a local binary file. By the usage as MQTT Client the Logger is able to bypass short disconnects to the Message Broker with a ring buffer functionality. You can configure a ring buffer as well for the local binary file storage.

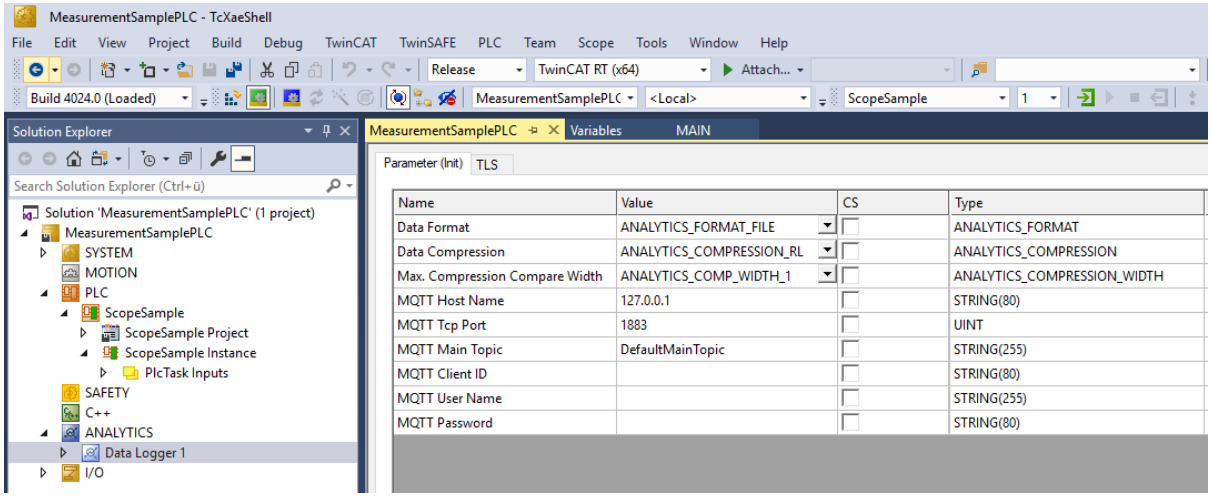
- To configure the Analytics Logger you have to navigate in your existing TwinCAT Project to the Analytics tree node



- Right click on this node and click on “Add Data Logger” to add one new instance to your configuration



- For configuring the base settings, please double click on the new tree item



You can make your specific Analytics Logger settings

-Data Format: Binary file or MQTT stream

-FILE format: Analytics Logger stores the data in local binary files and all other settings are not necessary anymore. The files will be stored in C:\TwinCAT\3.1\Boot\Analytics.

-BINARY: Data will be sent to the configured MQTT Message Broker. You can have multiple Logger in one TwinCAT project to communicate data to different MQTT Message Broker.

-Data Compression: on (default) or off

-Max Compression: mode of the compression

-MQTT host name

-MQTT Tcp port

-MQTT main topic for own hierarchical levels to keep the identification easy

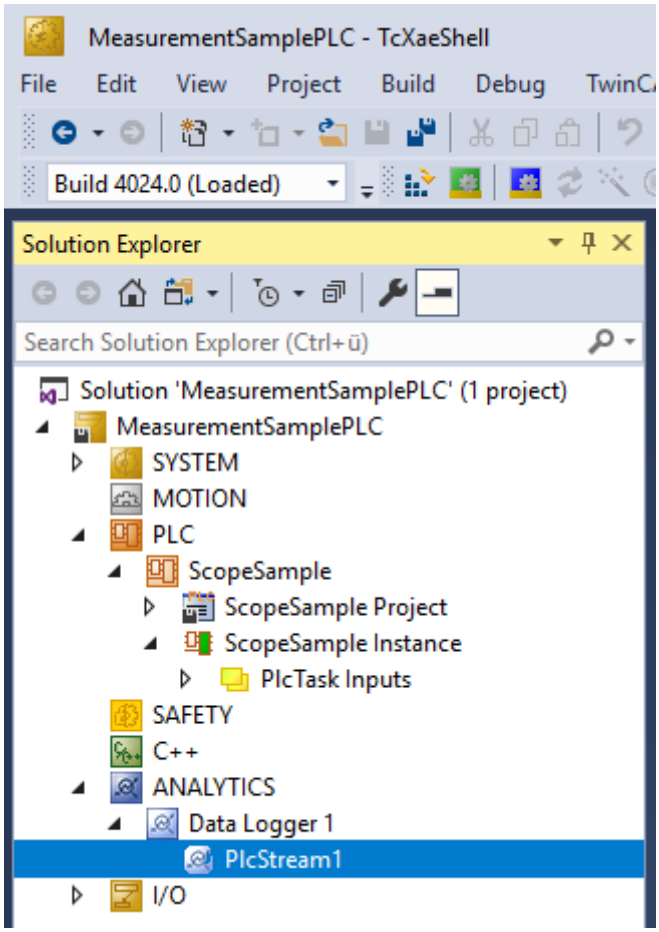
-MQTT Client ID should be unique in the network

-MQTT username

-MQTT password to make authentication at the message broker

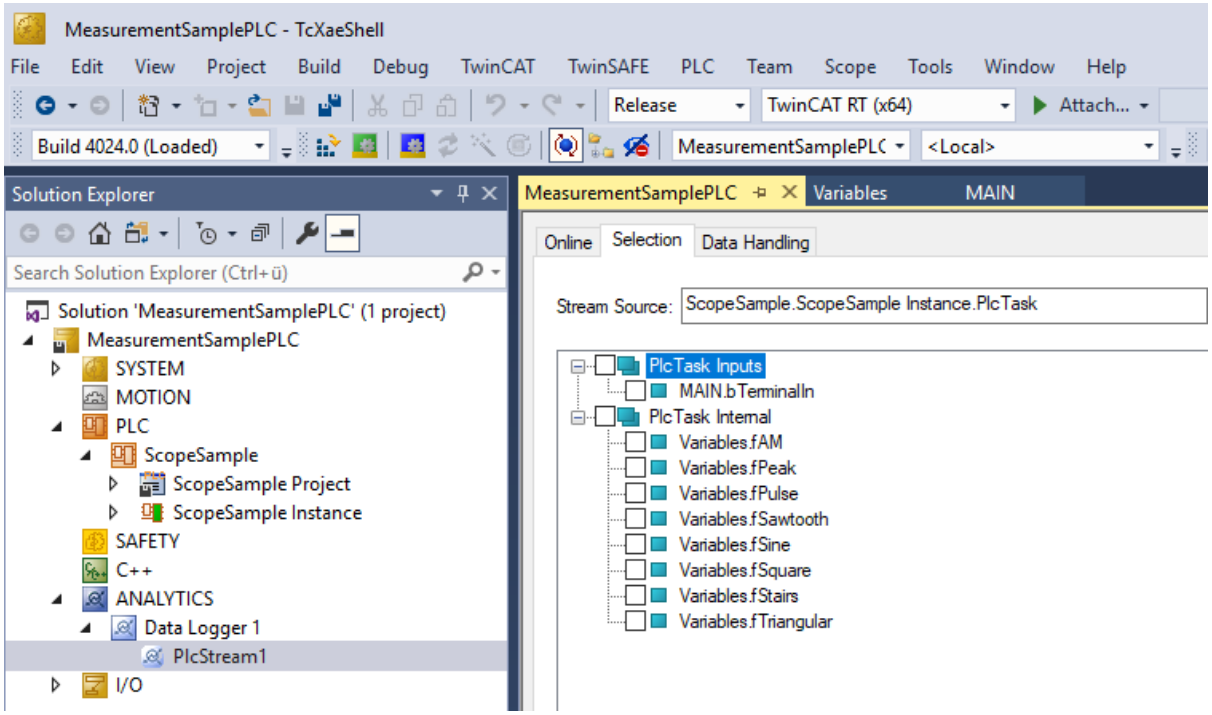
-At the TLS (Transport Layer Security) tab, security settings can be configured. TLS is a secure communication channel between client and server. By the usage of certificates, the TCP port 8883 is exclusively reserved for MQTT over TLS. Analytics Logger is supporting the modes CA Certificates, CA Certificates & Client Certificate and Preshared Key (PSK) mode.

- If variables in your PLC application are marked in the declaration with the attribute {attribute 'TcAnalytics'} they will be shown automatically as a stream below the Data Logger tree node.

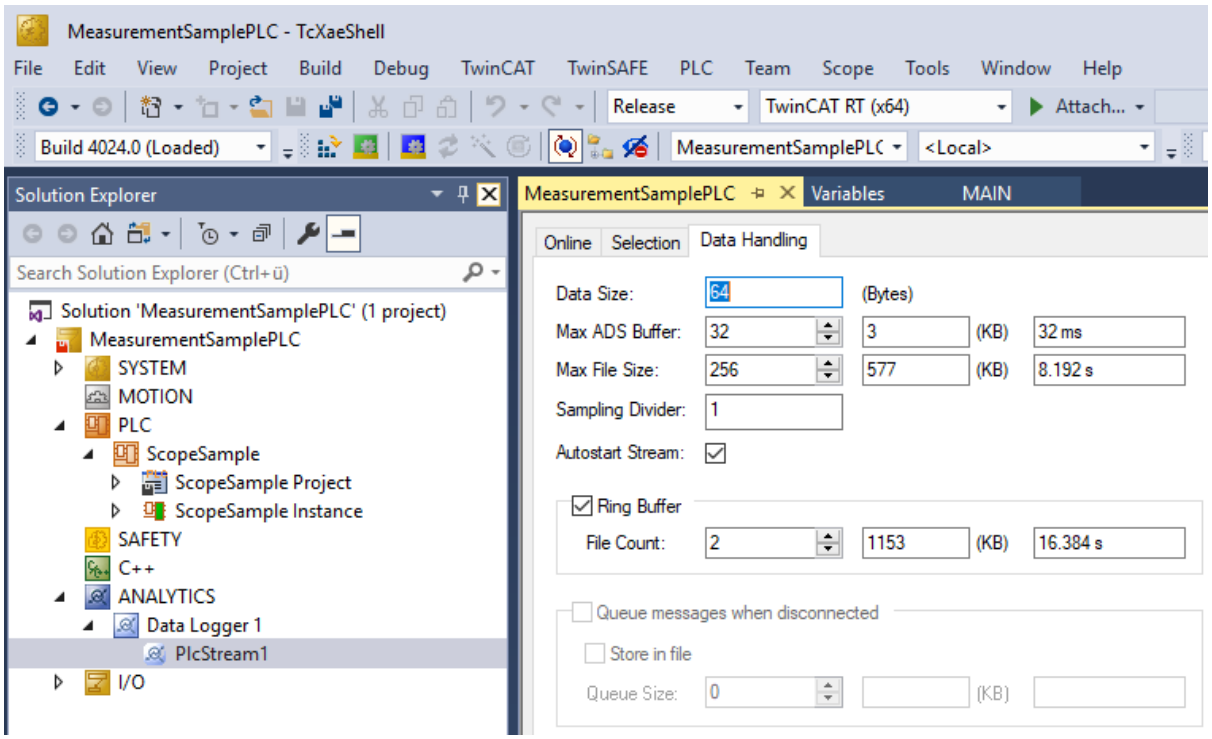


An additional device stream will be shown if your configuration provides an EtherCAT Process Image.

- In the stream a Selection tab is available to choose the variables that should be recorded

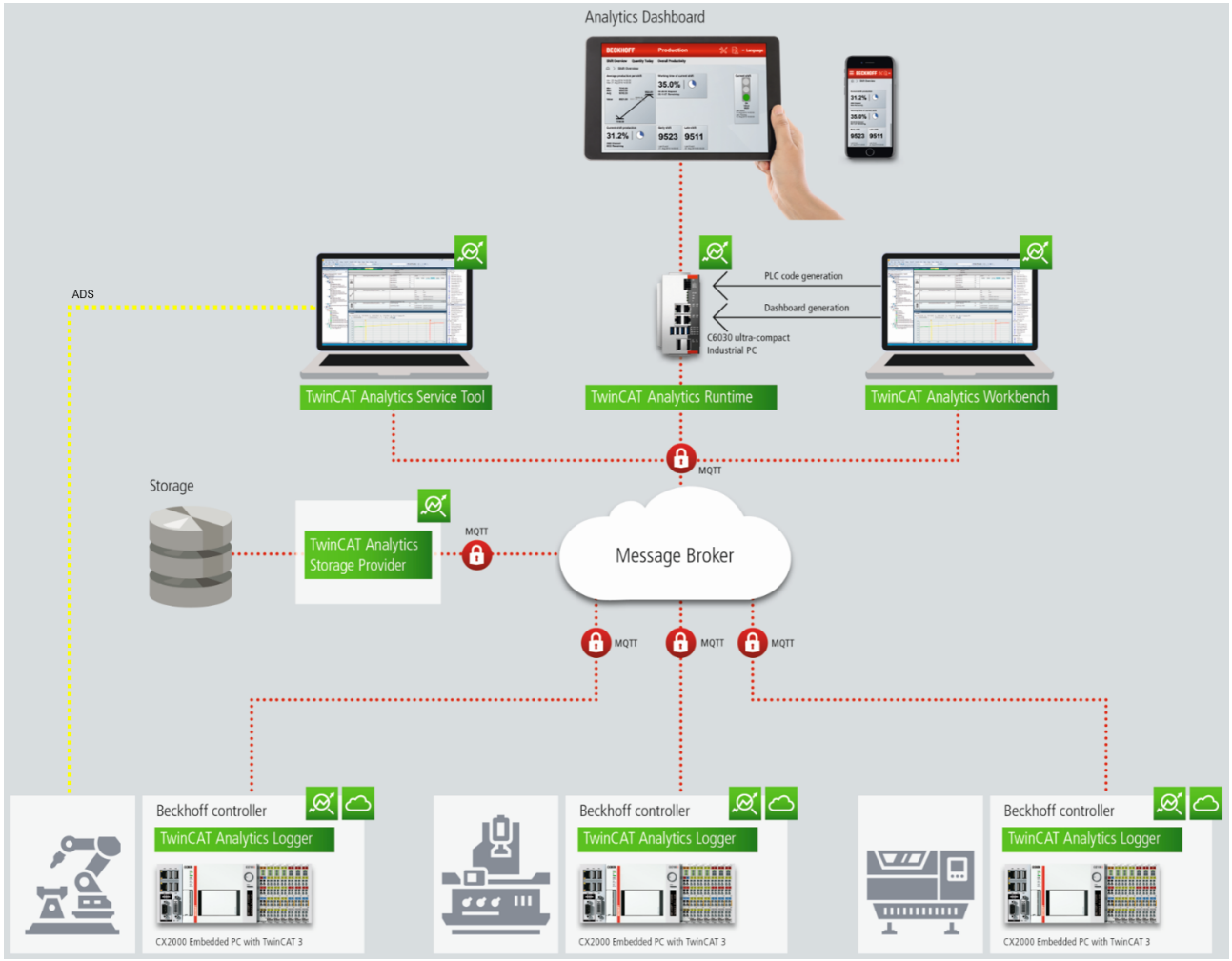


- Finally it is possible to change the package size for the frames or to configure the ring buffer for disconnects and file in the Data Handling tab.



4.2 Communication

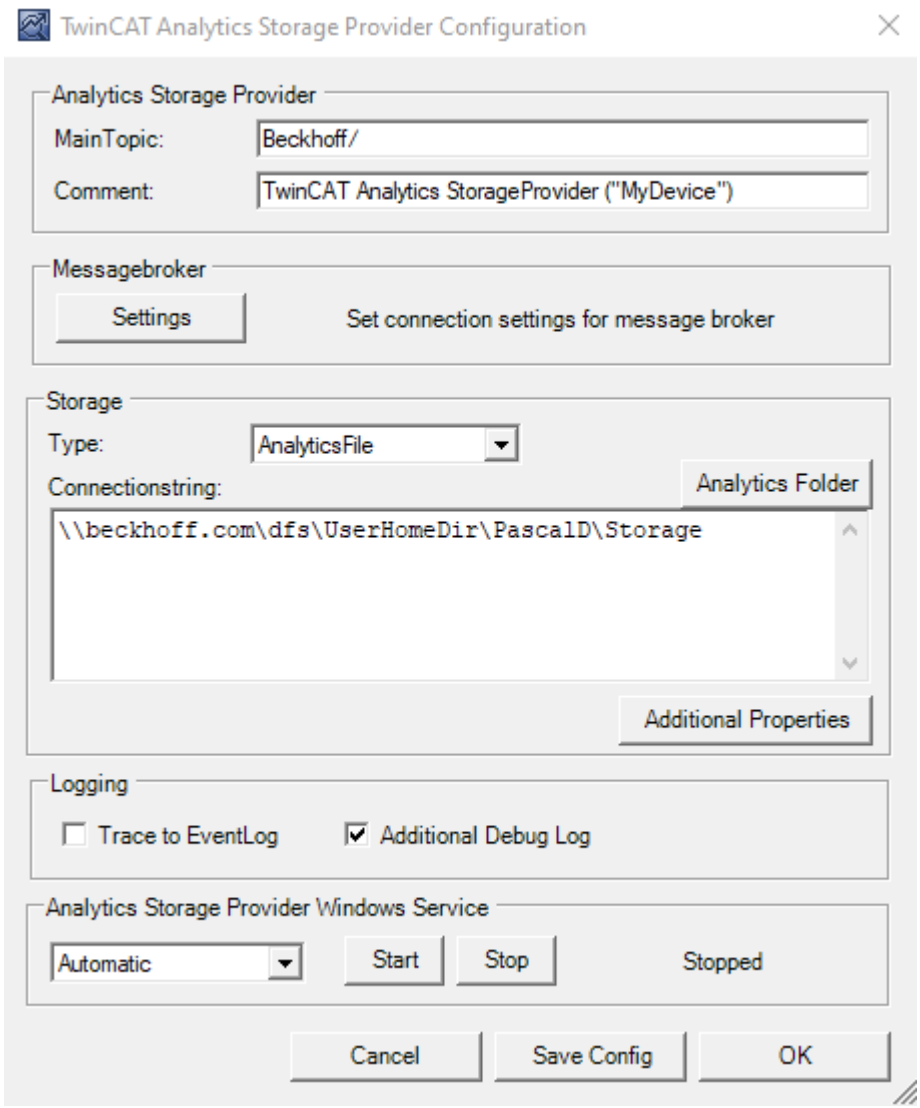
Currently, the Analytics workflow is fully mappable via MQTT. The engineering tools can also access the data of the machines via ADS and carry out analyzes.



If you choose for the IoT communication protocol MQTT you have to setup a native MQTT Message Broker somewhere in the network (VM in a cloud system is also possible). This Message Broker provides a decoupling of the different applications in the Analytics Workflow.

4.3 Historicize data

After the TwinCAT Analytics Storage Provider has been installed, the service running in the background can be configured. You will find the TwinCAT Analytics.StorageProvider.Configurator application in the folder *C:\TwinCAT\Functions\TF3520-Analytics-StorageProvider\Tools*.

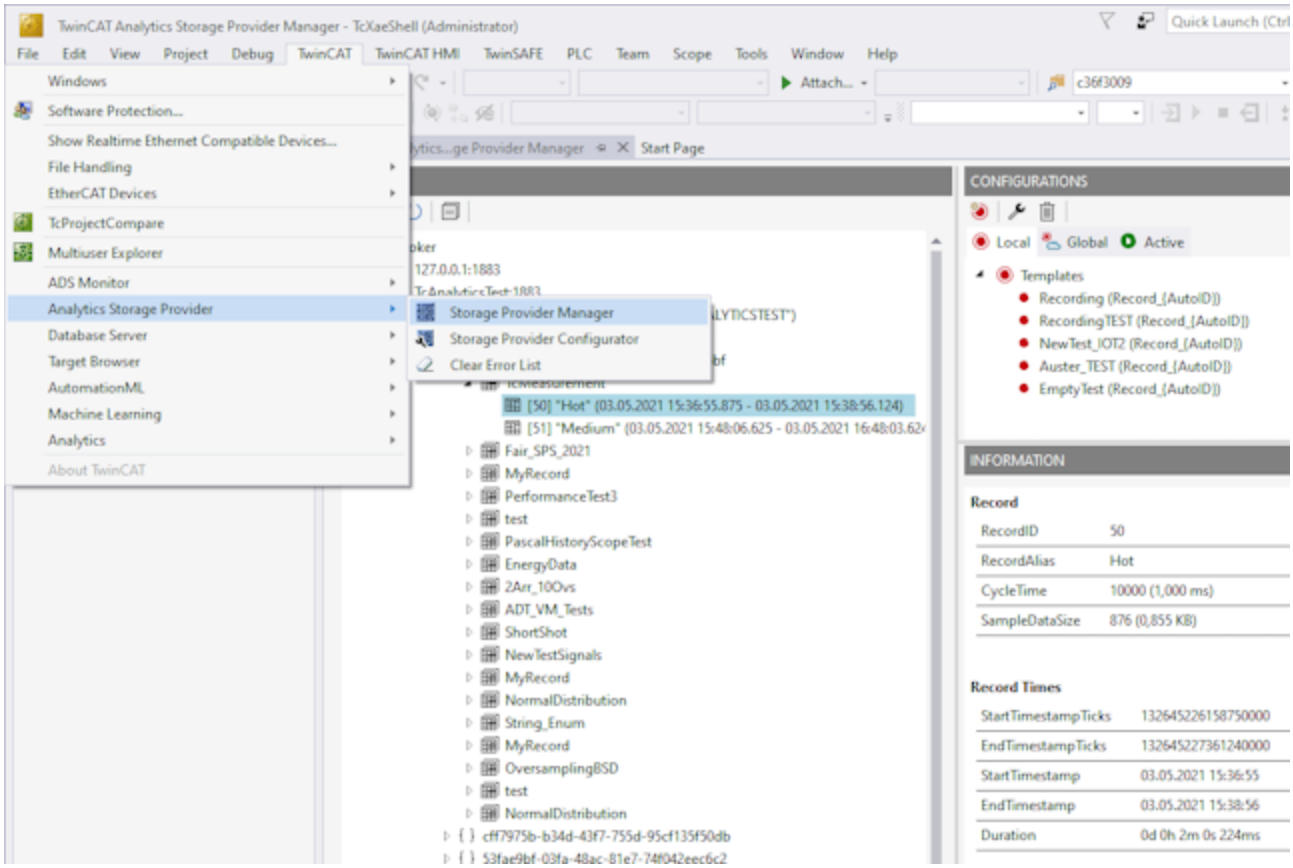


The main part of the topic can be defined in the configuration as well as the comment, which is used for identification if more than one Storage Provider is registered with the message broker.

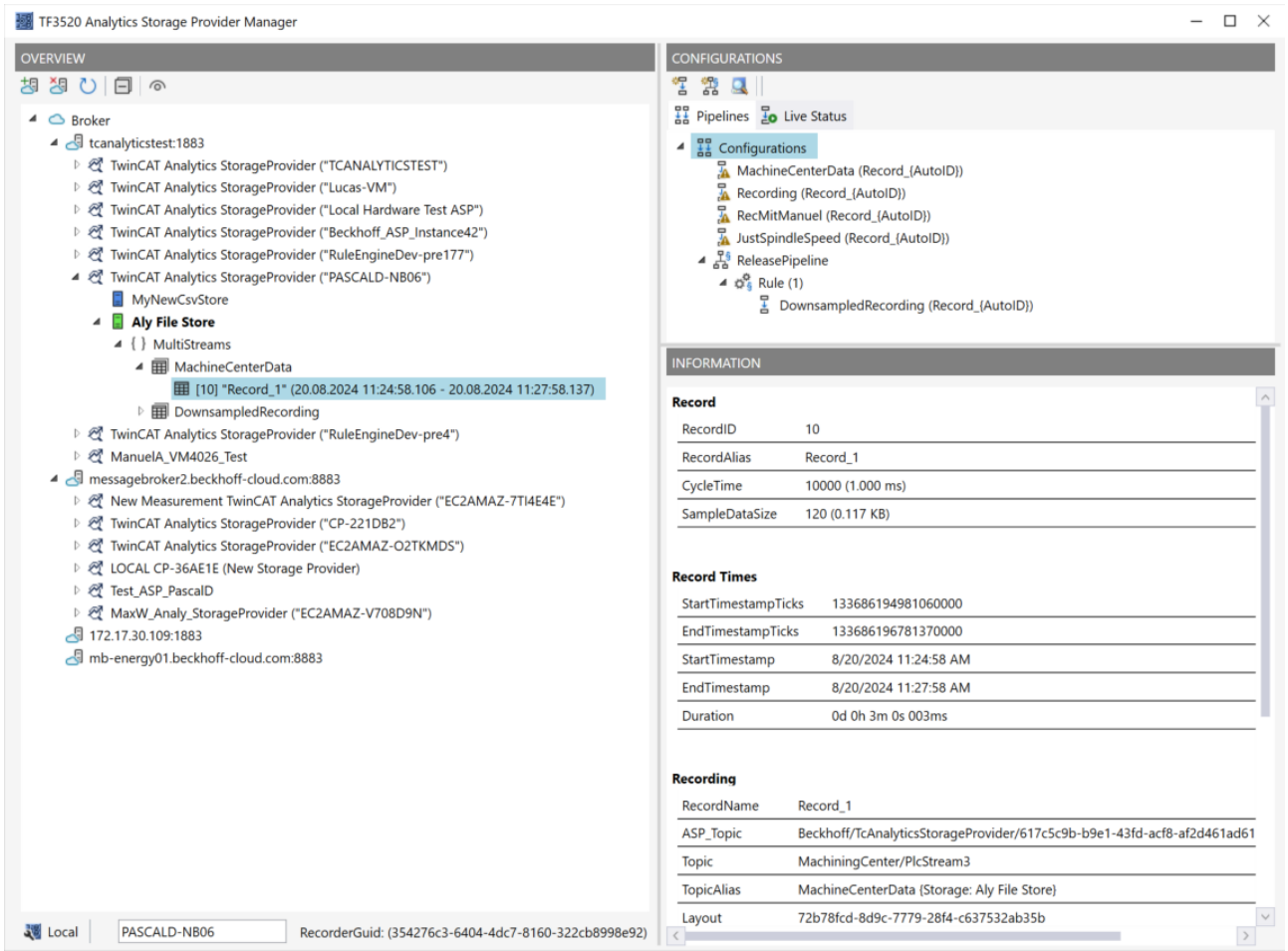
You can make the message broker settings and decide on a storage type:

- Analytics File (binary file)
- CSV file
- Microsoft SQL (binary / plain text)
- InlfuxDB (plain text)
- Microsoft Azure Blob (Azure Cloud required)

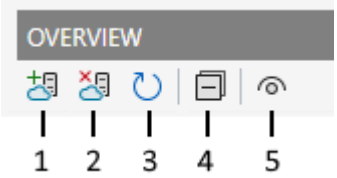
At last you can save the configuration and start the service. The next step is to configure the specific recording. For this you should select the **Storage Provider Manager** in your development environment.



With the Storage Provider Recorder recording definitions can be created, started and managed. In addition, it is possible to manage the data memories of individual Analytics Storage Providers. All important properties of the found Analytics Storage Providers and historized data are clearly displayed.



Toolbar Manager window ("OVERVIEW")

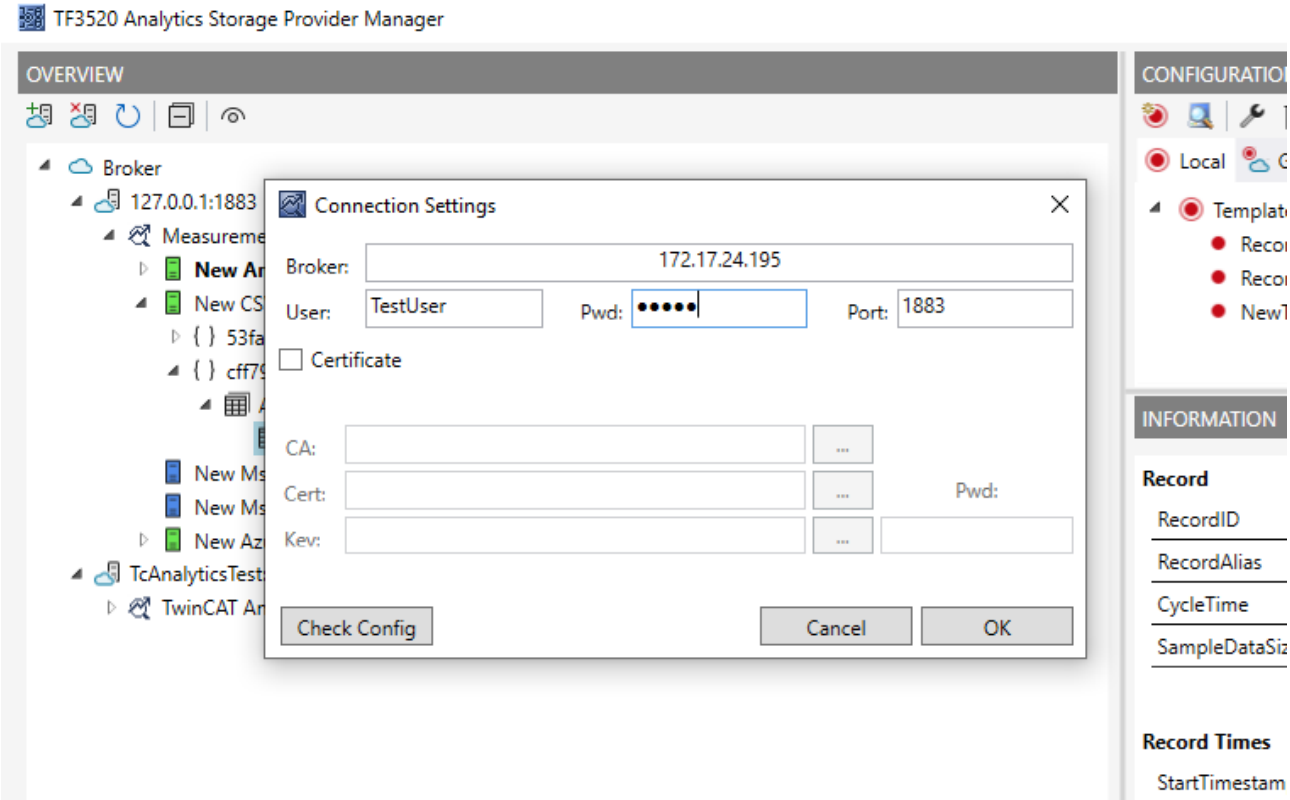


1	Add new broker
2	Remove selected broker
3	Refresh display
4	Collapse all nodes
5	View switch between dark/light mode

Function Manager window ("OVERVIEW")

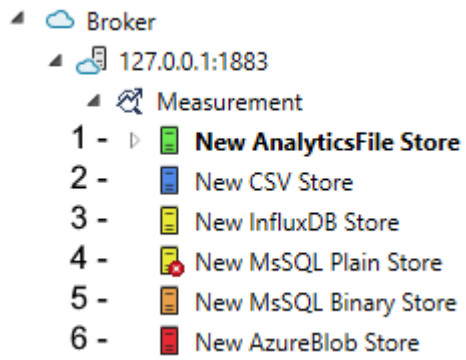
First assign a **RecorderAlias**. This helps to group the started recordings and to find its self started ones again.

After that, one or more brokers can be set up. This is done via the already known input mask for MQTT connection properties.



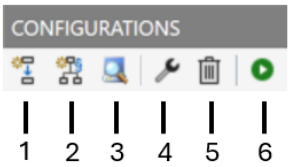
Once a connection to the broker could be established, all Analytics Storage Providers connected to it will be listed.

"Storage" status



1	Storage Online
2	Storage Offline
3	Storage starts
4	Storage starts with error. Still trying to start it
5	Storage is shut down
6	Storage is in the error state

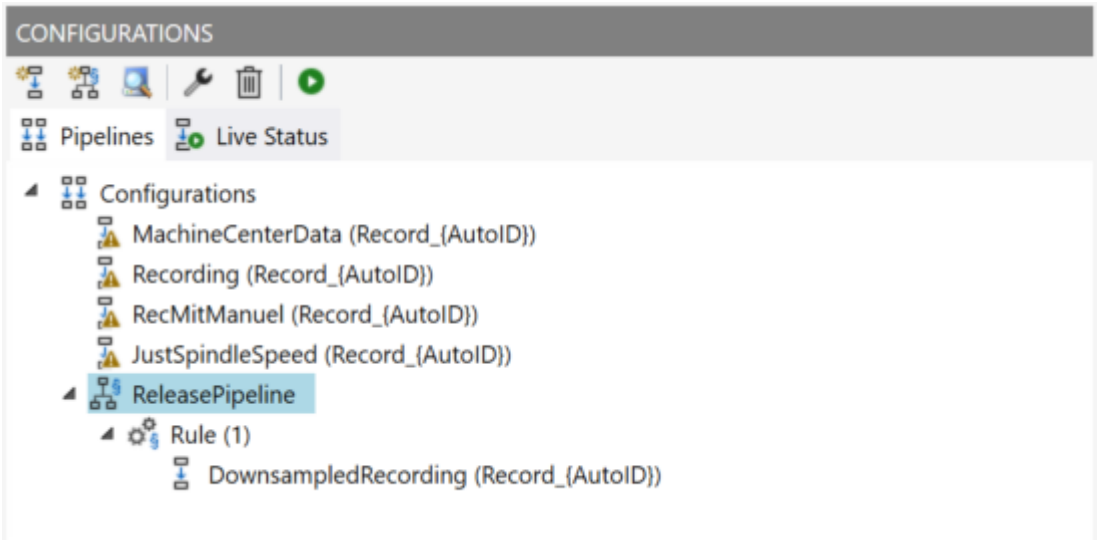
Toolbar Manager window ("CONFIGURATIONS")



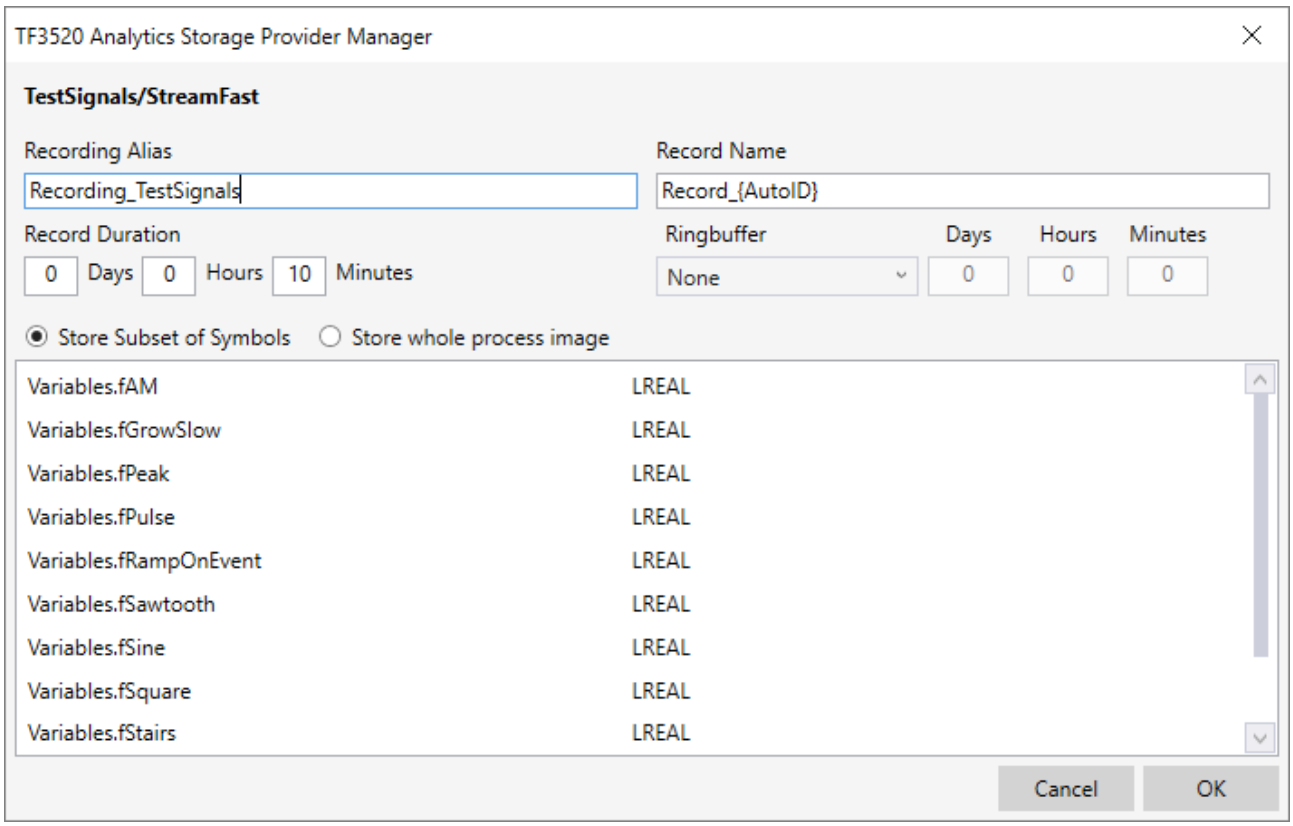
1	Create a new pipeline
2	Create a new pipeline with Rule Engine
3	Open Target Browser for connecting simple pipelines
4	Edit a selected pipeline
5	Delete a selected pipeline
6	Start a selected pipeline

Function Manager window ("CONFIGURATIONS")

The window is divided into two tabs. Pipelines and Live Status. Under Pipelines you will find the configurations of your pipelines. You can define new pipelines from here. Edit existing. Delete or start.



To create a new simple pipeline, click the "Create new pipeline" button. The following dialog opens.



You can now drag and drop the symbols you want to record from the Target Browser into the dialog. You also assign a Recording Alias and a Record Name.

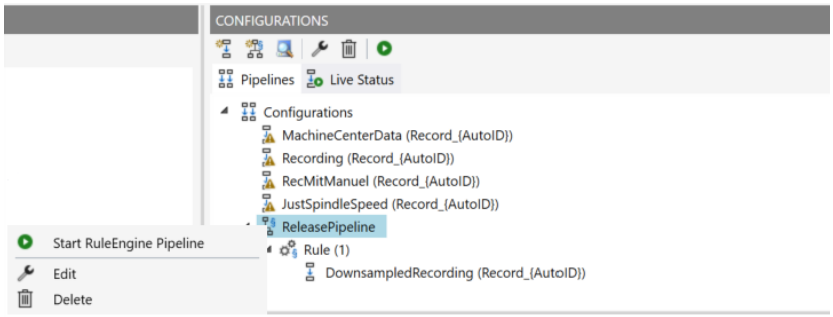
Various placeholders are available for the Record Name:

"{AutoID}"	
"{Topic}"	
"{SystemID}"	
"{Layout}"	
"{CycleTime}"	
"{SampleSize}"	
"{RecordStart}"	

You can also configure recording names and a duration (otherwise the recording will run endlessly until it is stopped manually). A ring buffer can be set according to storage space or time.

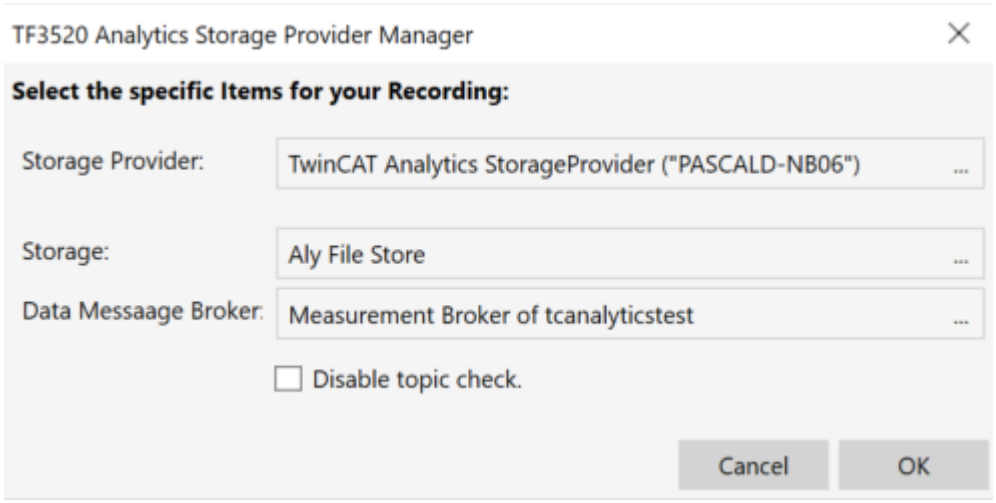
The entries are confirmed with **OK** and a new local recording definition is created.

It is now possible to start this definition directly via the toolbar or the context menu.



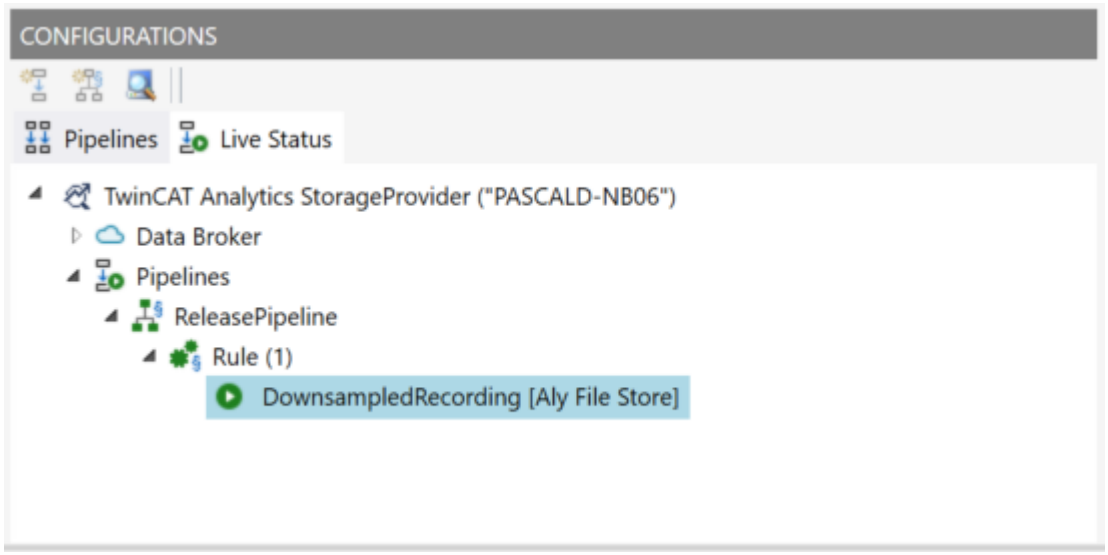
However, it is also possible to make the definition globally accessible. This can be done via the context menu with the entry **Publish Recording**.

The following dialog then opens:



Here you can now select the desired Analytics Storage Provider via which the definition is to be published. In addition, the definition is assigned a Storage and a Data Broker of the selected Analytics Storage Provider. After the selection, the recording definition is confirmed with **OK** and published to the selected Analytics Storage Provider. This means that it can be found by any Storage Provider Manager that is connected to the MQTT Broker.

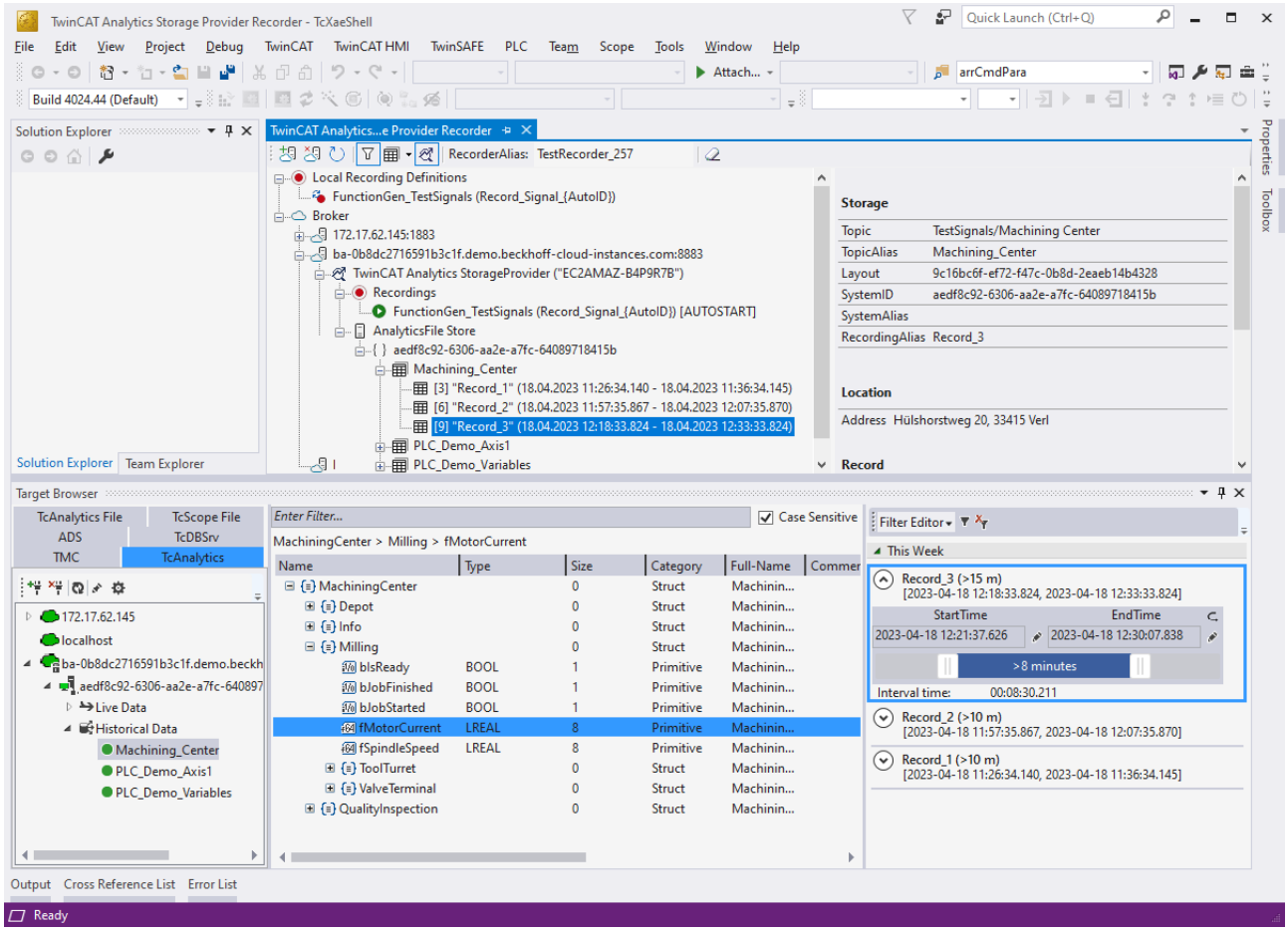
After starting a pipeline, the view automatically jumps to the second tab, the Live Status.



All active recordings from all users are listed here. The recordings can be ended in this tab and it is also possible to jump to the resulting record.

Use historized data

After and also during recording, you can select the historical data as input for your analysis in Target Browser. In the Target Browser, you will find a new control on the right side for the historical data. There you can select the timespan for your data.



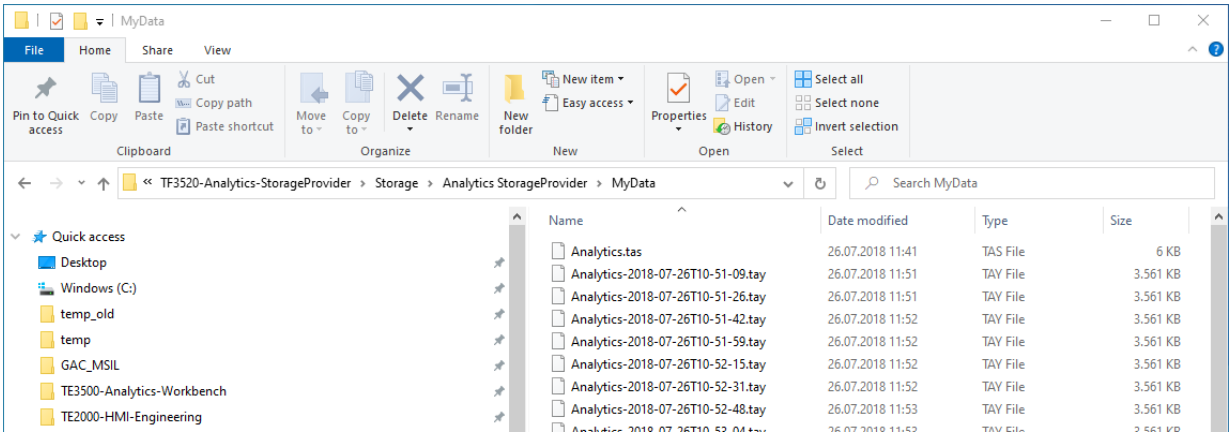
4.4 Importing/converting Analytics Files

In the following it is assumed that you have installed TwinCAT under "C:/TwinCAT". Otherwise, you must adjust the specified paths accordingly.

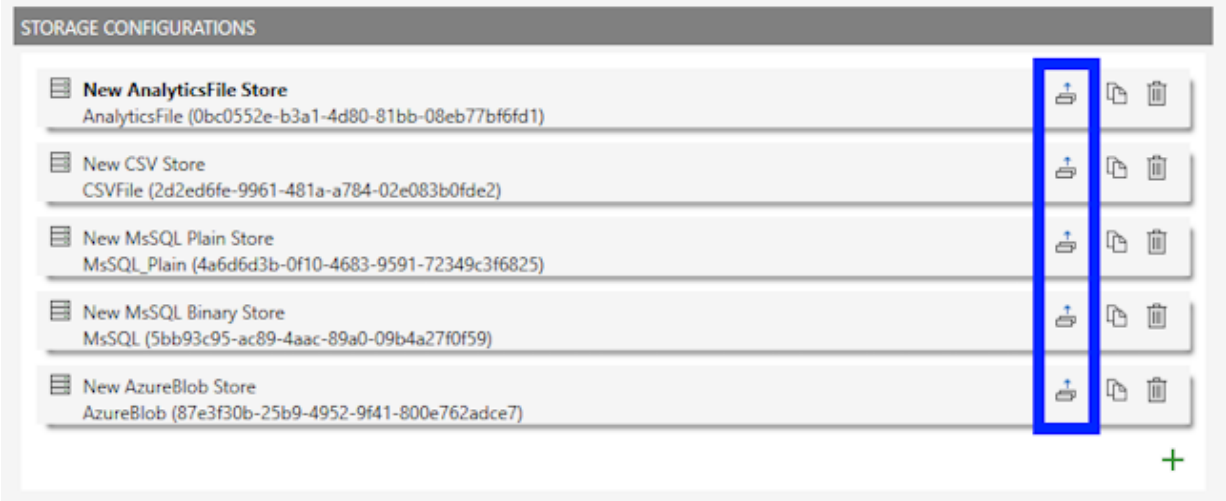
You can import recordings from the Analytics Logger stored in Analytics File Format (*Analytics.tas, Analytics-<Date>.tay*) into the Storage Provider. In general, you can convert data saved by the Storage Provider as an Analytics File into other formats. Analytics File is always the source format.

To do this, perform the following steps:

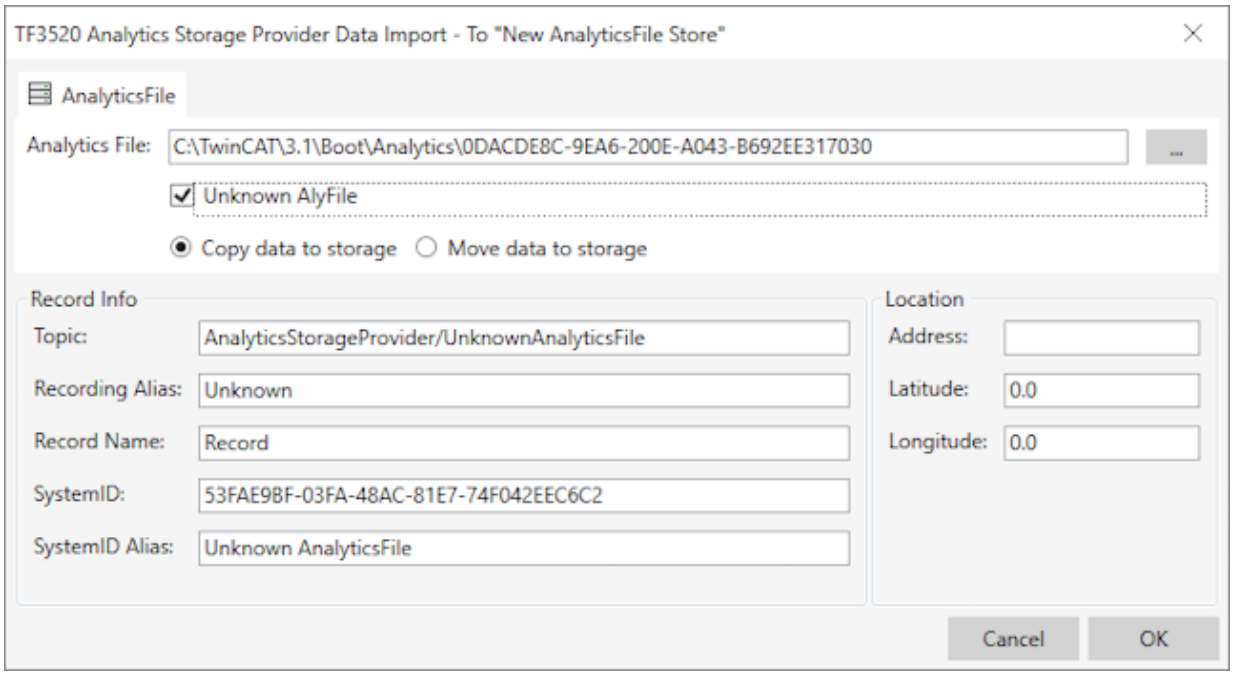
1. Save the folder with the Analytics Files in your Storage Provider location
   By default, here: *C:\TwinCAT\Functions\TF3520-Analytics-StorageProvider\Storage\Analytics StorageProvider* (create the folders manually if they do not exist)



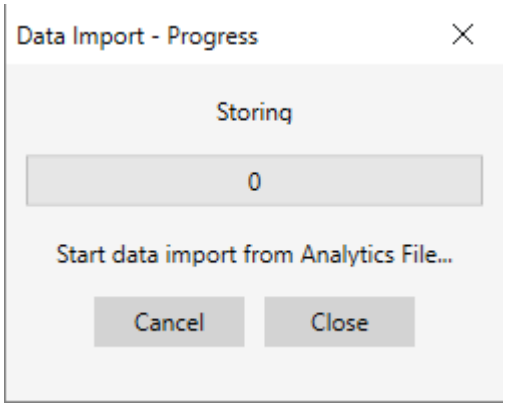
2. Open *TwinCAT.Analytics.StorageProvider.Configurator.exe*. The program can be found under the path *C:\TwinCAT\Functions\TF3520-Analytics-StorageProvider\Tools*
3. Select the Storage into which the Analytics File is to be imported and press the **DataImport** button.



4. Then select the path to the Analytics File in the DataImport dialog and enter all other known parameters for this recording.



5. Press **OK** and the data import begins.



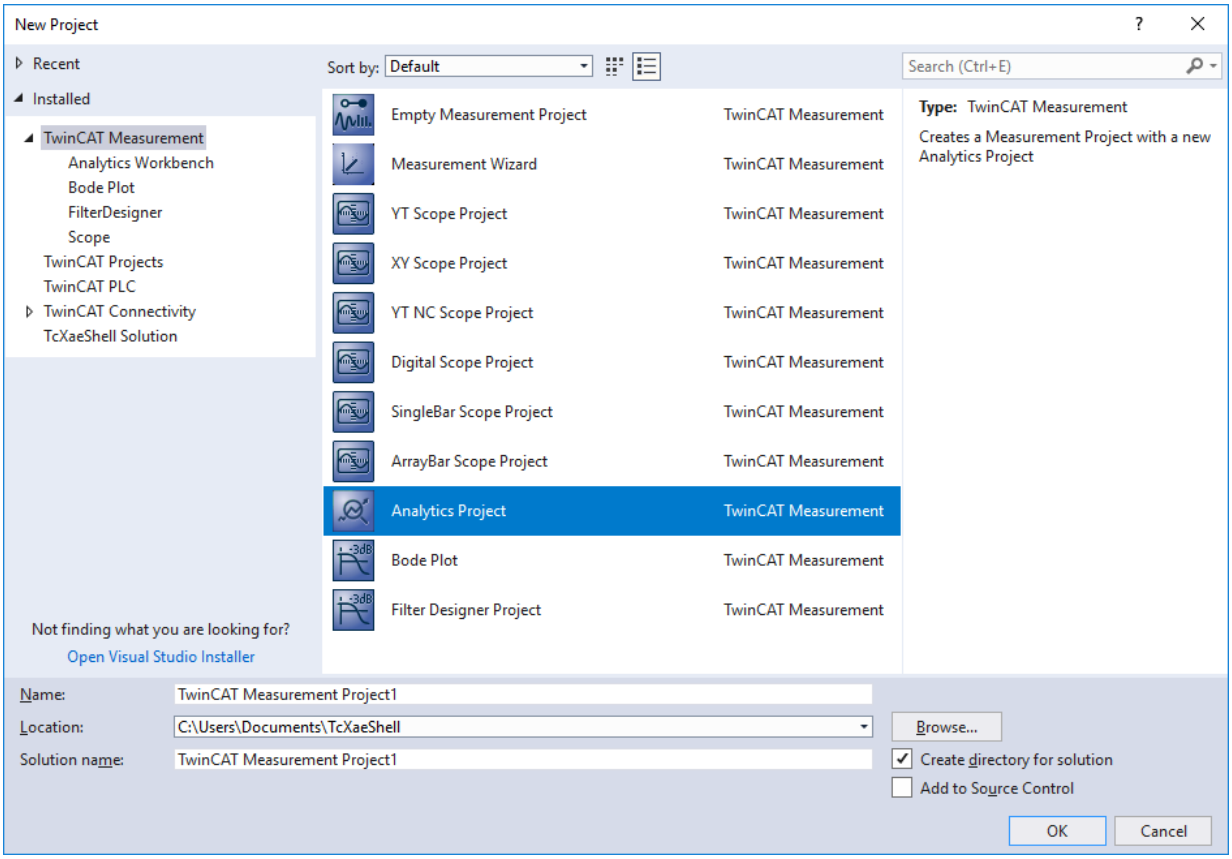
⇨ Now you can see your imported data in the TwinCAT Target Browser [▶ 95].

⇨ You may have to wait a short time or restart your Storage Provider.

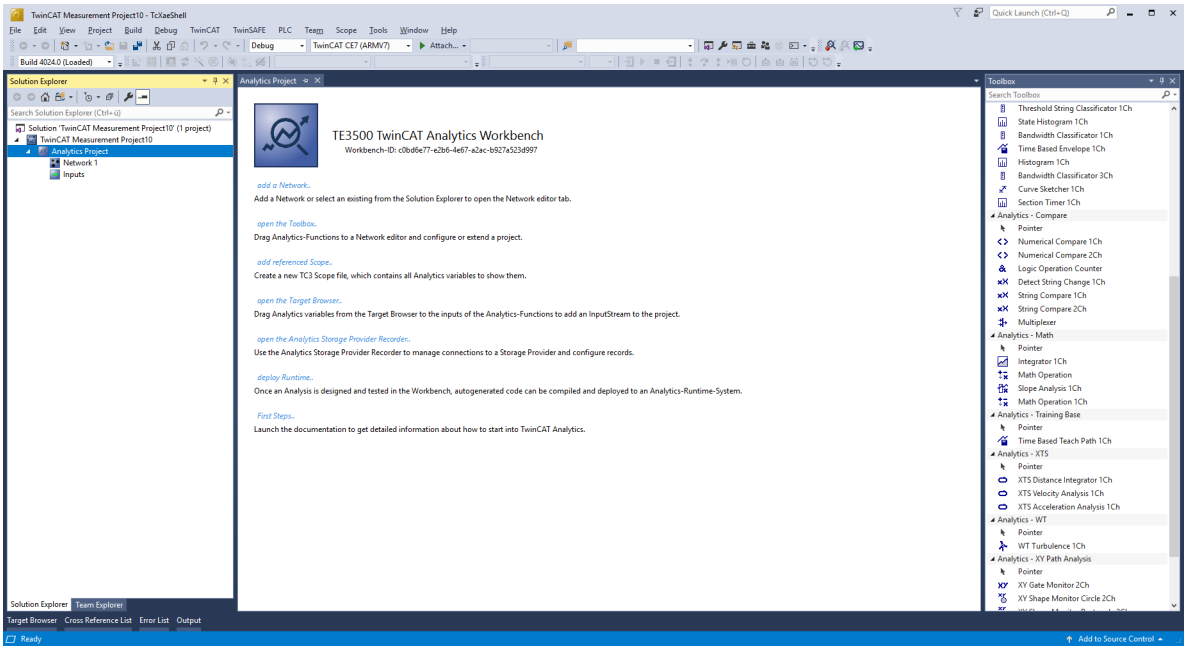
4.5 Analyse data

✓ Open your TwinCAT Engineering environment to start the data analysis.

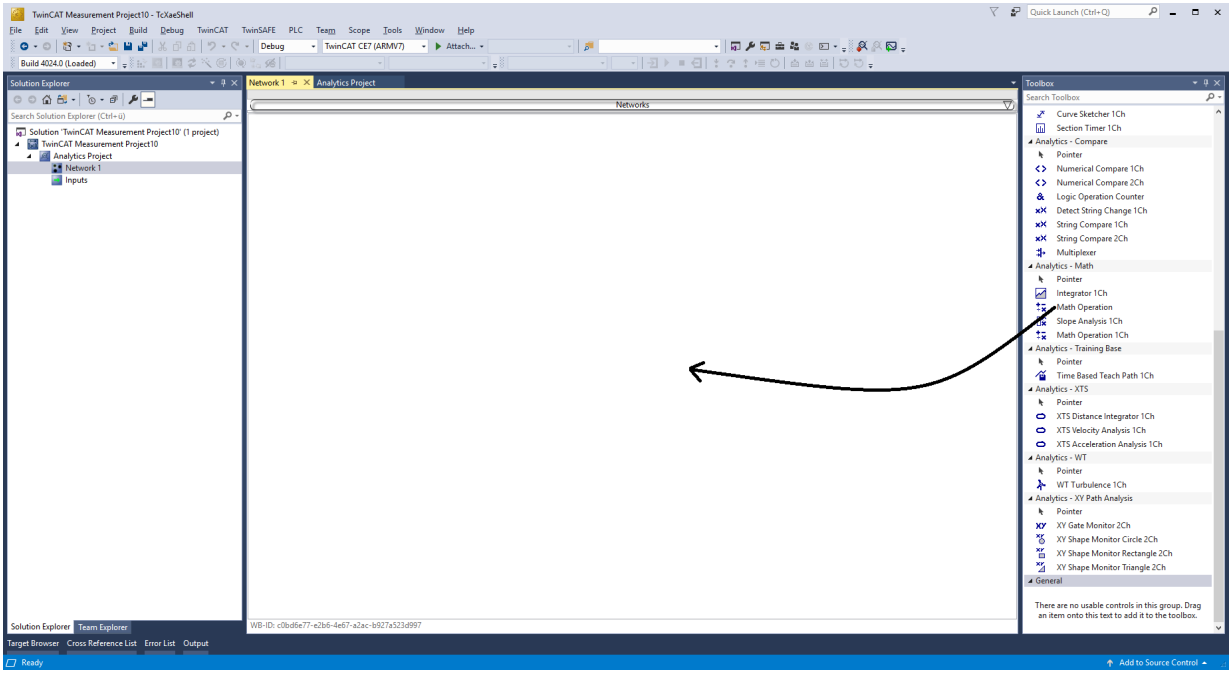
1. Open **Visual Studio® > File > New > Project…**
2. Select the **Analytics project template** from **TwinCAT Measurement**.



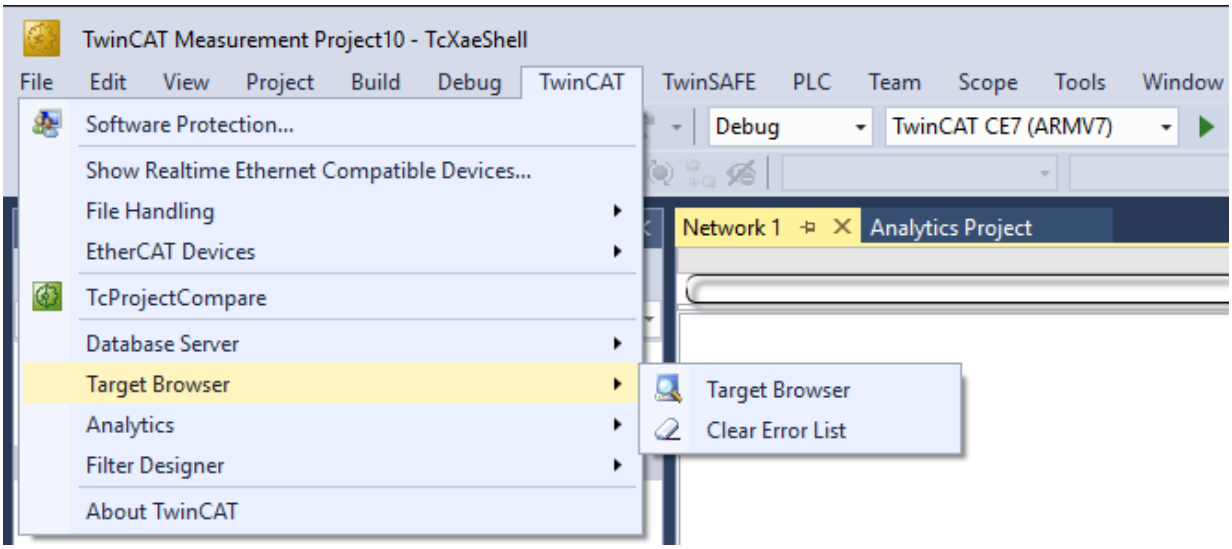
⇨ The new project is displayed in the Solution Explorer. After clicking the **Analytics Project** tree node element a start window opens where you can select your first action. From here you can add a network, open the **Toolbox**, open the **Target Browser** or open the **Analytics Storage Provider Recorder**. In the following steps you will perform all these actions.



3. It makes sense to open the **Toolbox** of Visual Studio® first. There you will find all the algorithms supported by TwinCAT Analytics. Algorithms need to be grouped and organized into networks. Right-click **Analytics Project** to add a new network, or add a network using the start page. The first network is always generated by default.

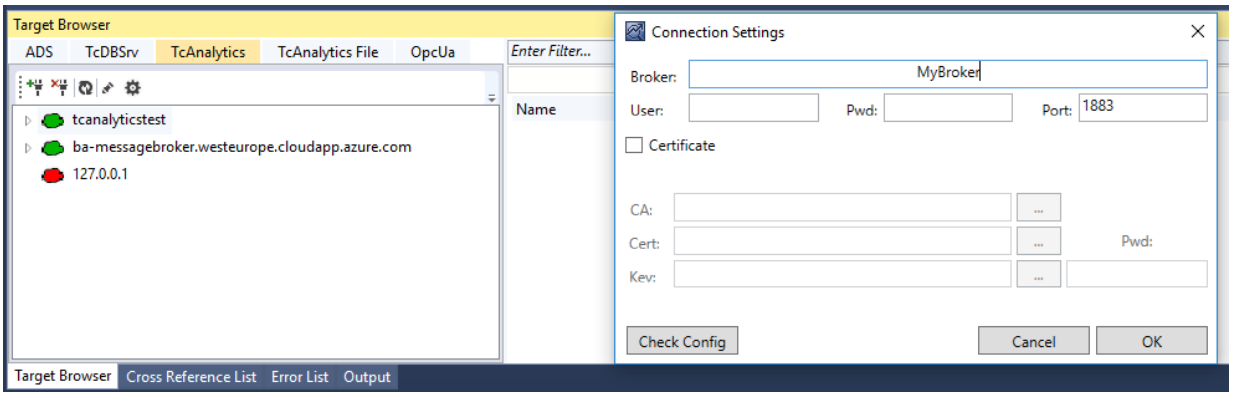


4. When you click on the network, an editor opens. Now you can drag and drop the desired algorithm into the editor interface.
5. After selecting the algorithm, you need to connect input variables to the modules (algorithm). To do this, open the **Target Browser**.
   TwinCAT > Target Browser > Target Browser

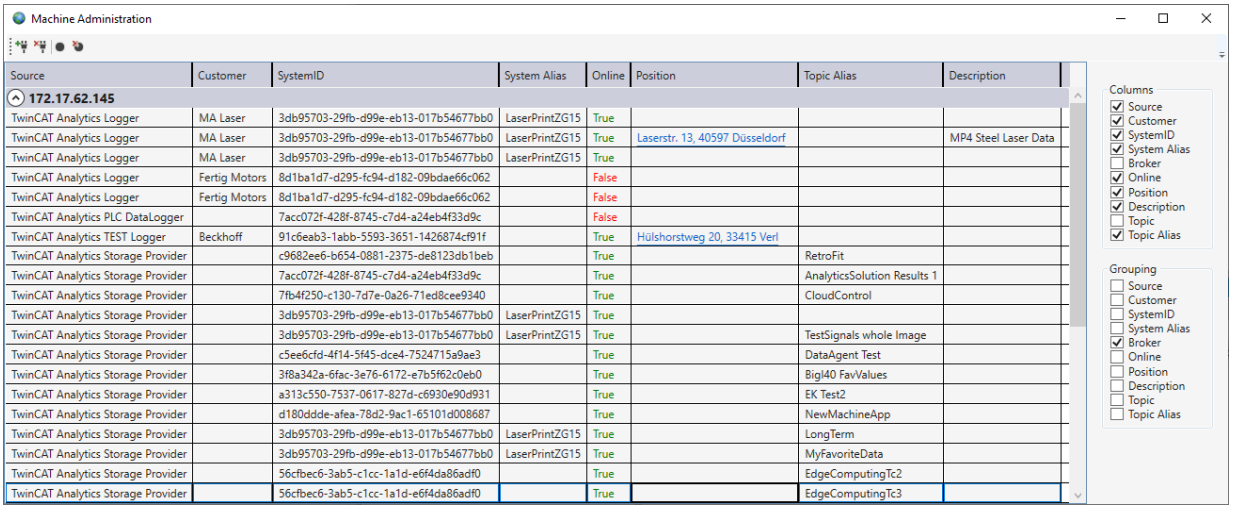


6. Now select the **TcAnalytics** or **TcAnalyticsFile** tab in the Target Browser. Continue with the tab **TcAnalytics** (MQTT).

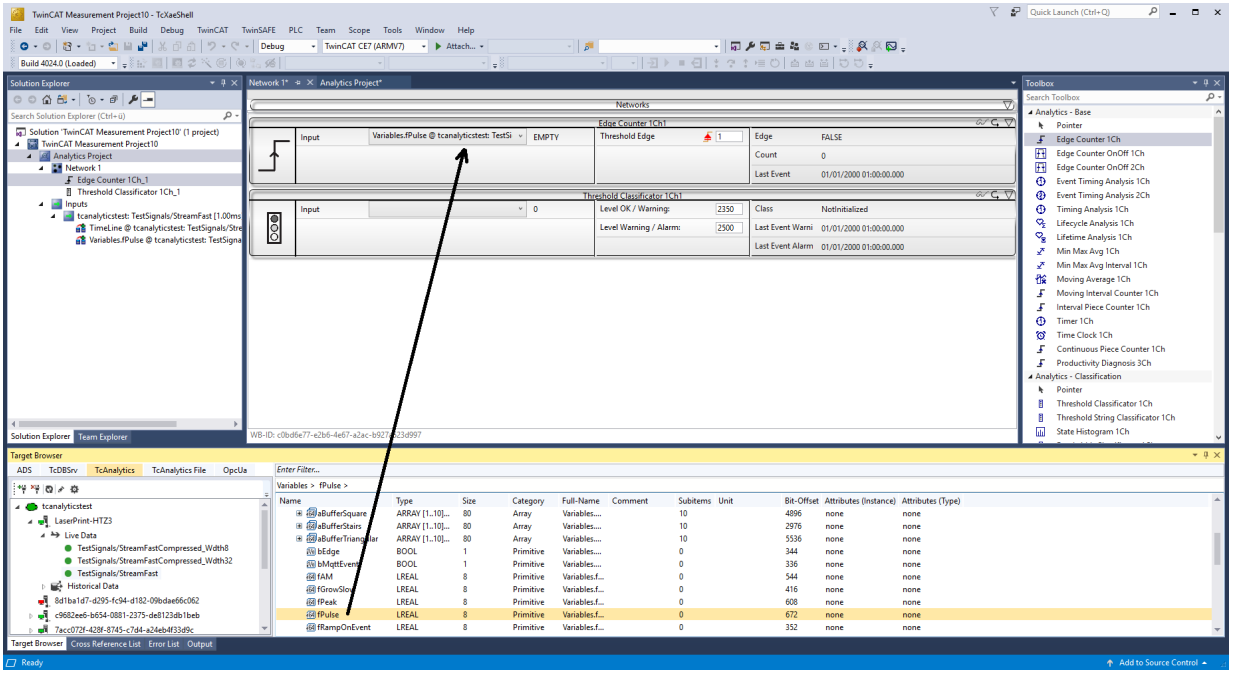
7. Click the icon highlighted in green in the toolbar of this Analytics extension. A window opens in which you can specify the connectivity data of your message broker.



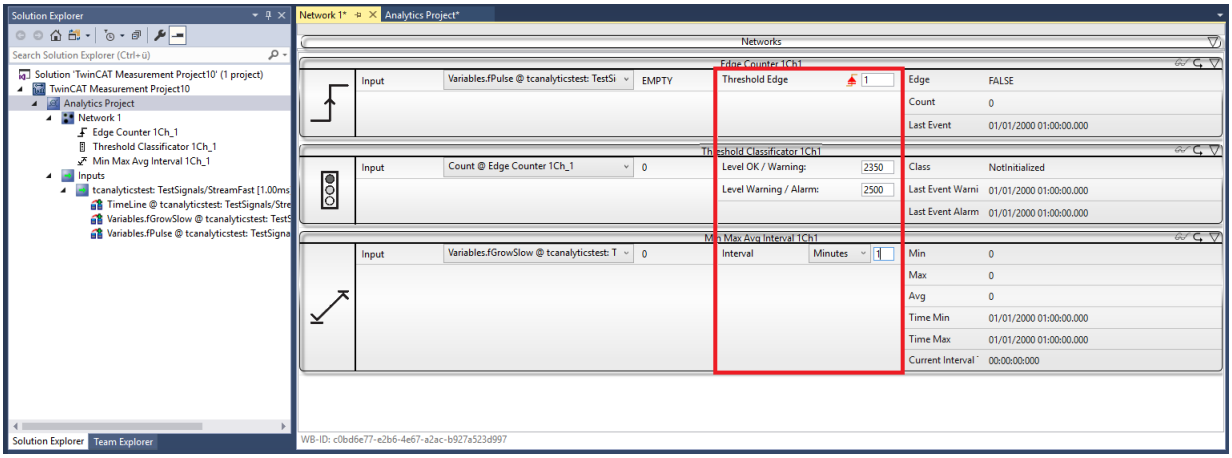
8. Select your MQTT Analytics client (TwinCAT Analytics Logger, TwinCAT IoT Data Agent or Beckhoff EK9160). There is a unique ID for each control. This ID is displayed in the Target Browser.
9. Clicking on the **gear icon**, you will get to the Machine Administration page. Here you can assign a system alias name that will be displayed in the Target Browser instead of the ID.



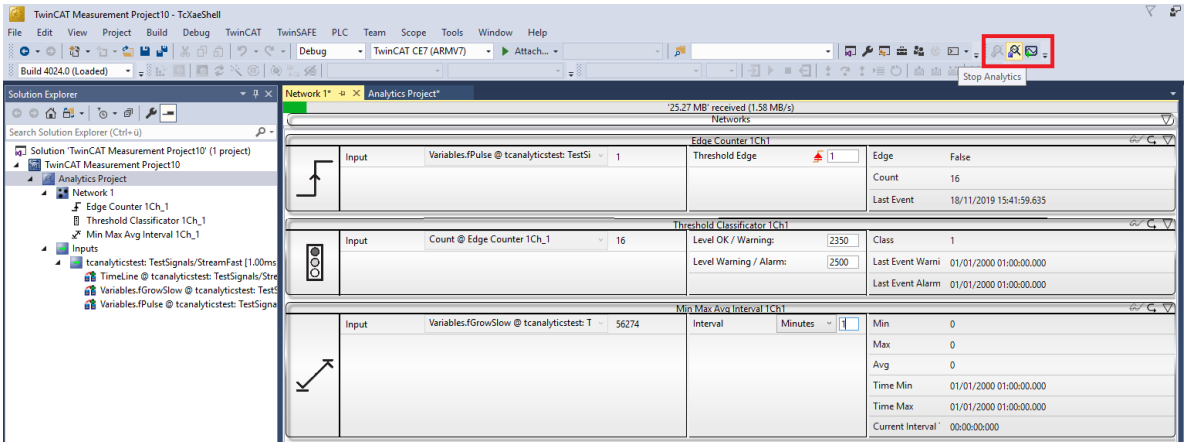
10. In the next step, you can choose between live data and historical data for each MQTT Analytics client. In this case, the historical data is provided by the TwinCAT Analytics Storage Provider.



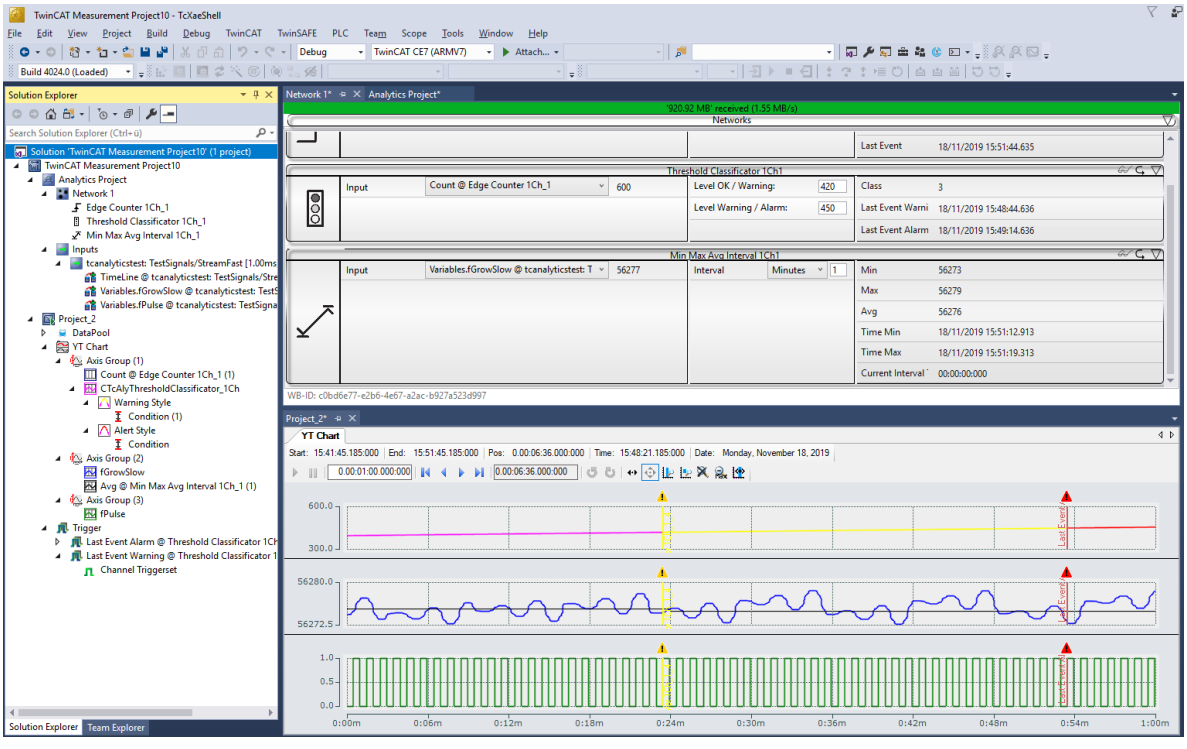
11. You can drag and drop the variables into the inputs of the specific algorithm. In most algorithms, conditions such as thresholds, time intervals, logical operators etc. can be specified. These settings are made in the middle of each module.



⇨ Finally, your first Analytics Project is complete. To start the analysis, click **Start Analytics**. To stop the analysis, click **Stop Analytics**.



⇨ Before starting the analysis or during runtime, you can click the **Add Reference Scope** button. This will automatically create a Scope configuration that matches your Analytics project.

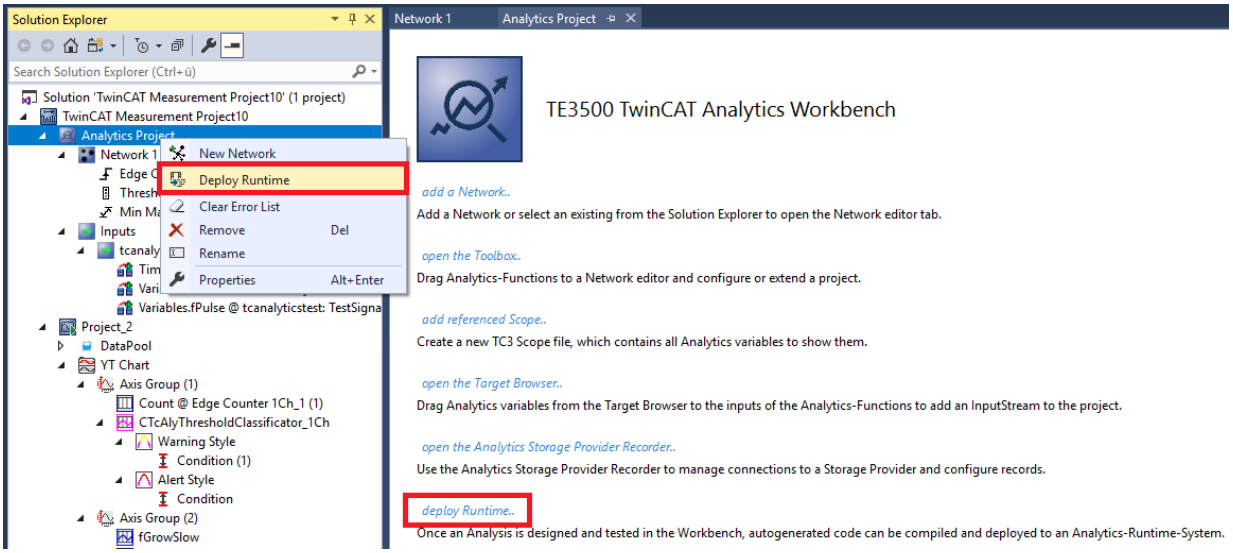


⇨ The analysis results can be displayed in the Scope View graphs using drag-and-drop. For example, a mean value can be displayed as a new channel in the view. Timestamps as markers on the X-axes show significant values.

4.6 24h Analytics application

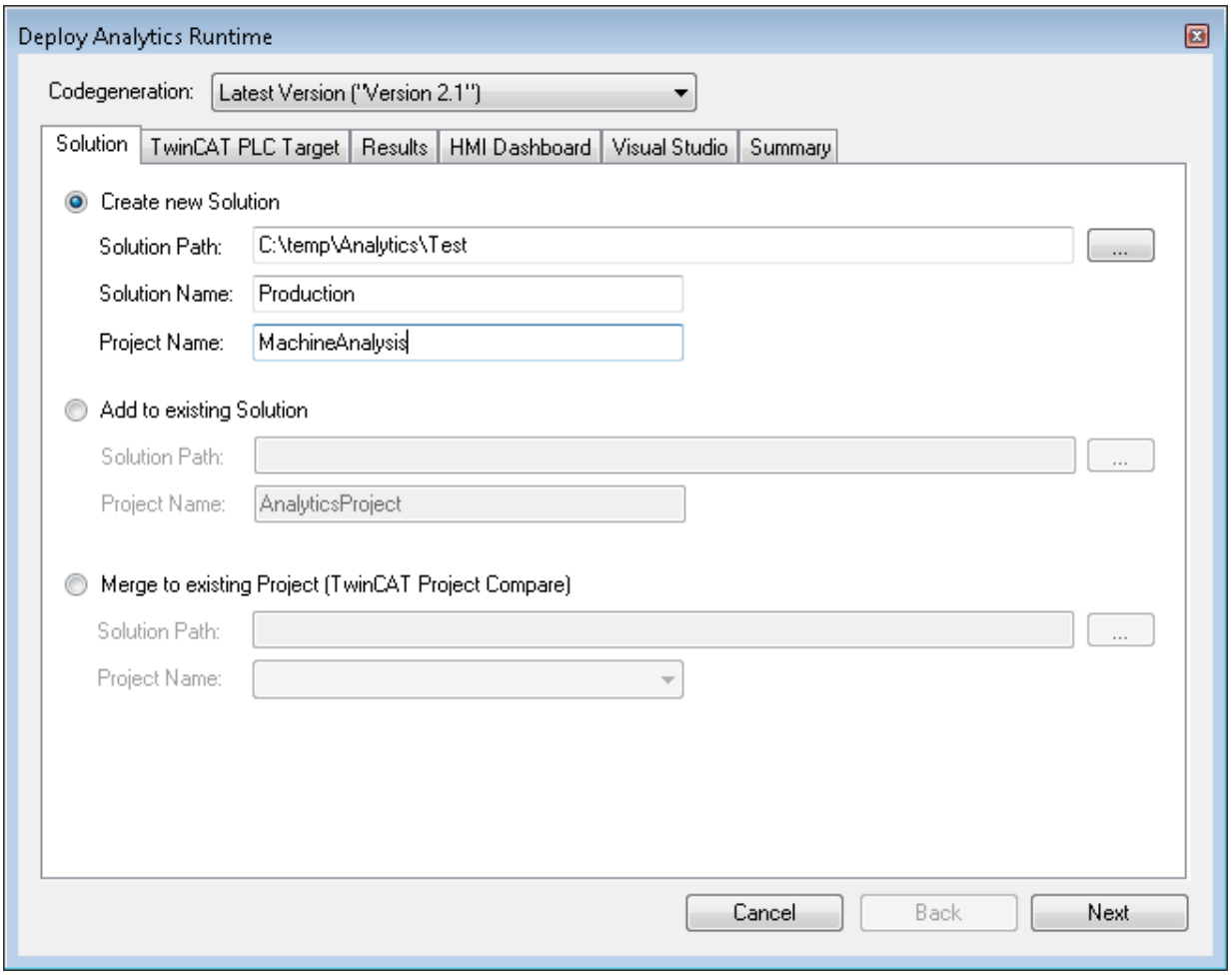
The last major step in the TwinCAT Analytics workflow is the continuous 24-hour machine analysis. It runs in parallel with the machine applications in the field. To make this very easy, the TwinCAT Analytics Workbench can automatically generate PLC code and an HTML5-based dashboard of your Analytics configuration. Both can be downloaded into a TwinCAT Analytics Runtime (TC3 PLC and HMI Server) and provide the same analysis results as the configurator tool in the engineering environment.

✓ First, save your configuration and open the Analytics Deploy Runtime Wizard. This can be done from the context menu in the Analytics Project tree item or from the start page.

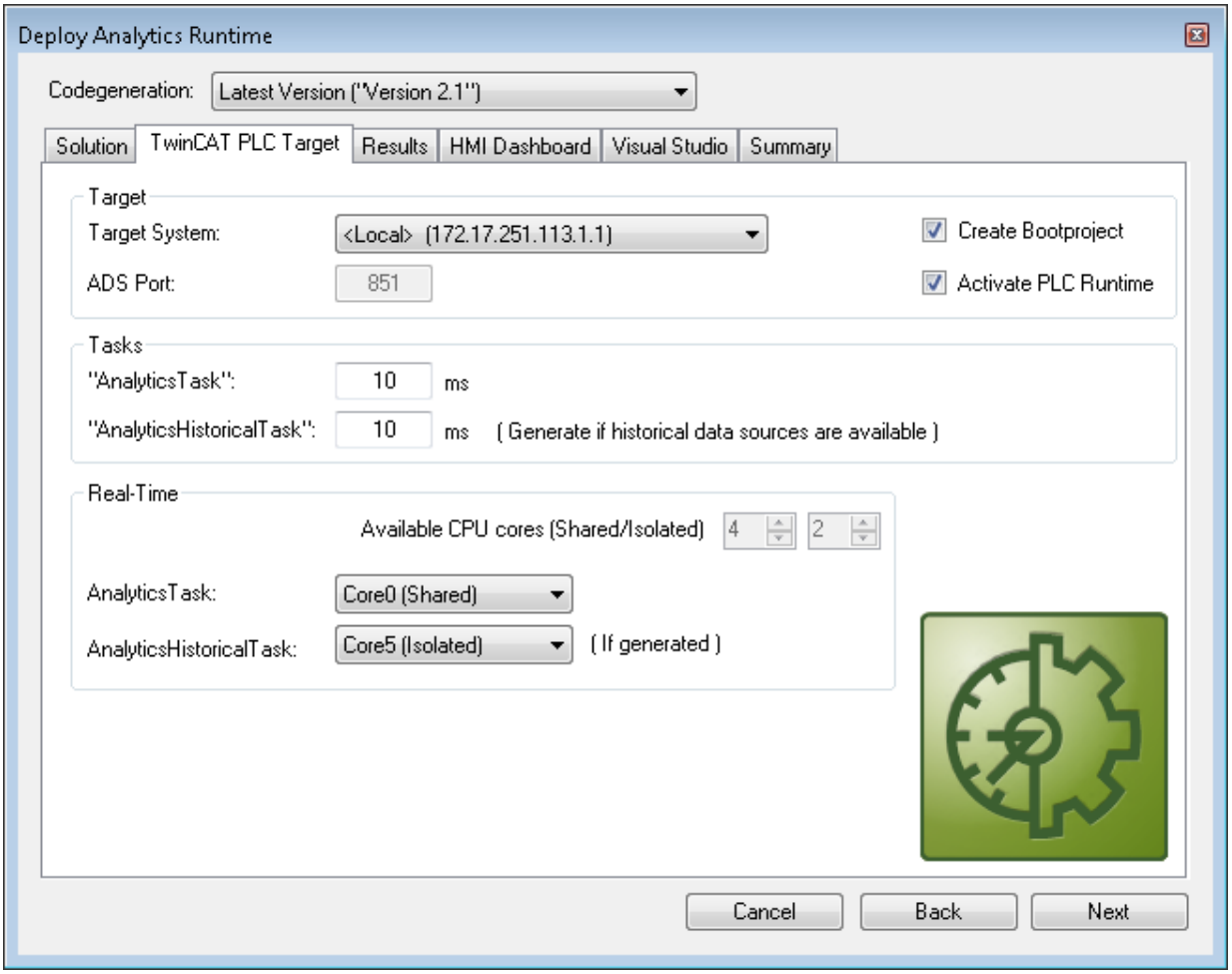


1. When the wizard is open, you can click through some tabs. The first one is called Solution. Here you can decide how your Analytics project should be used in the PLC code: As...
completely new solution.

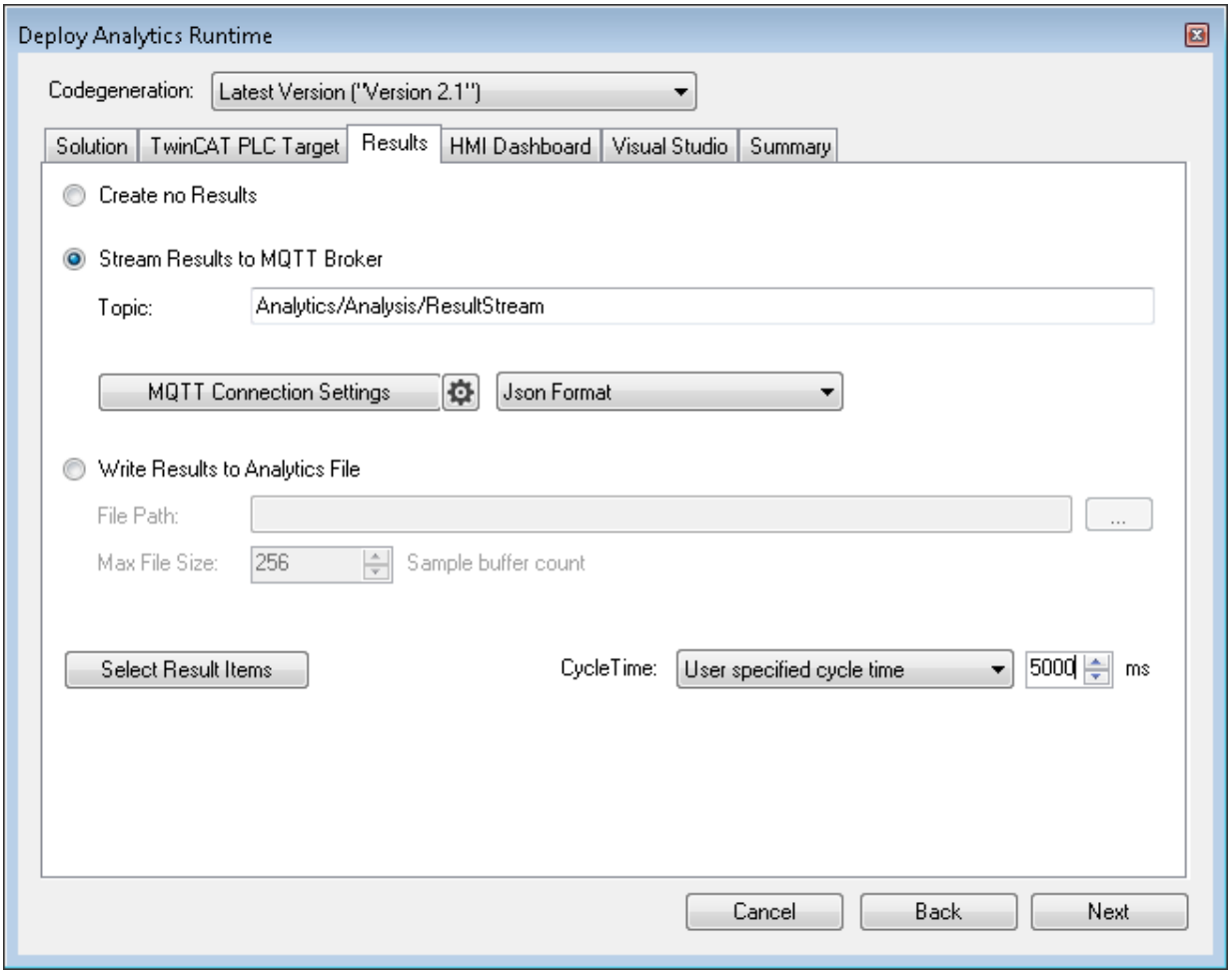
part of an existing solution.
update of an existing Analytics solution.



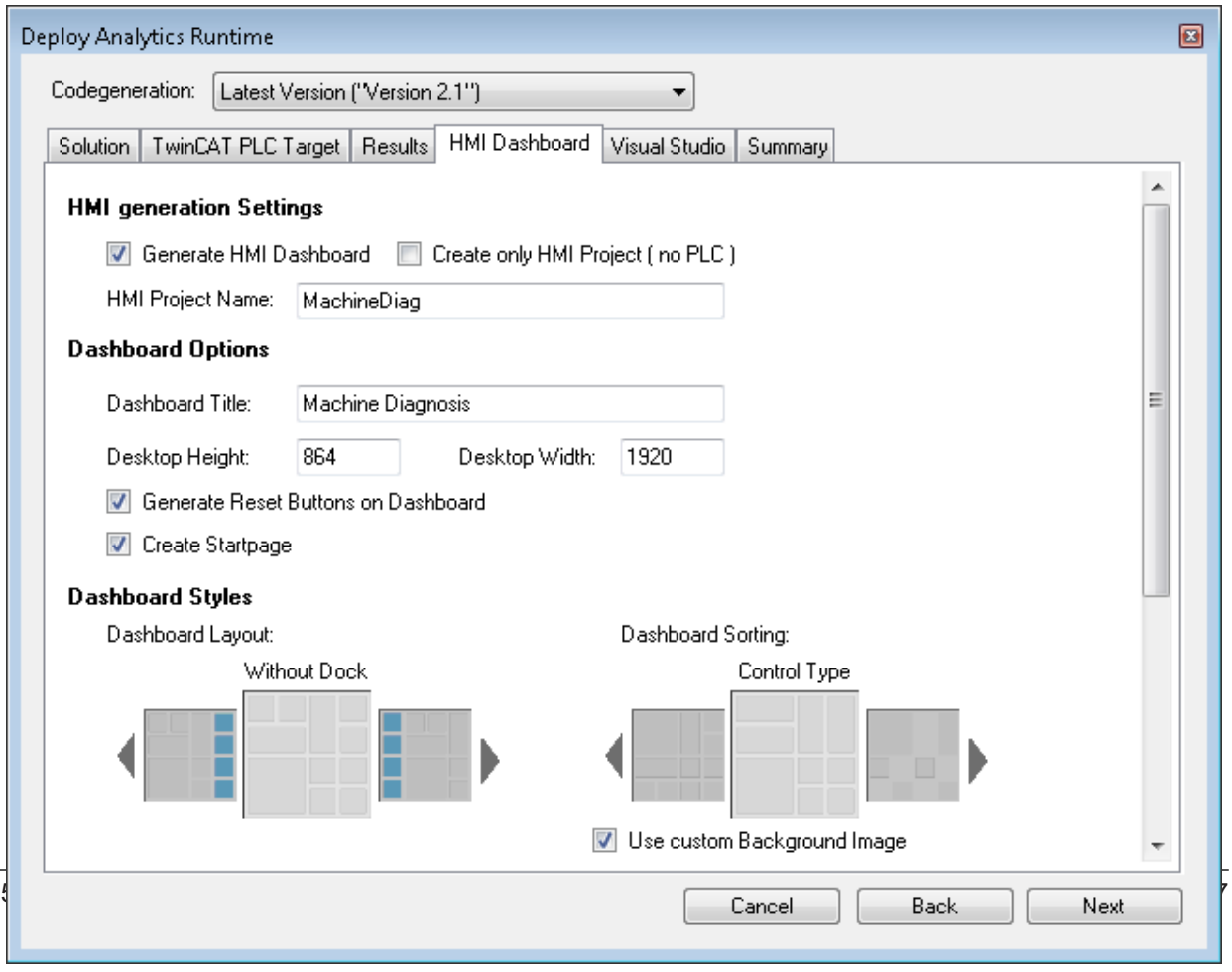
2. In the **TwinCAT PLC Target** tab you can select the ADS target system that runs the TwinCAT Analytics Runtime (TF3550). The created project is immediately executable. For this purpose you can set the Activate PLC Runtime option. In addition, it can be selected that a boot project is created directly.



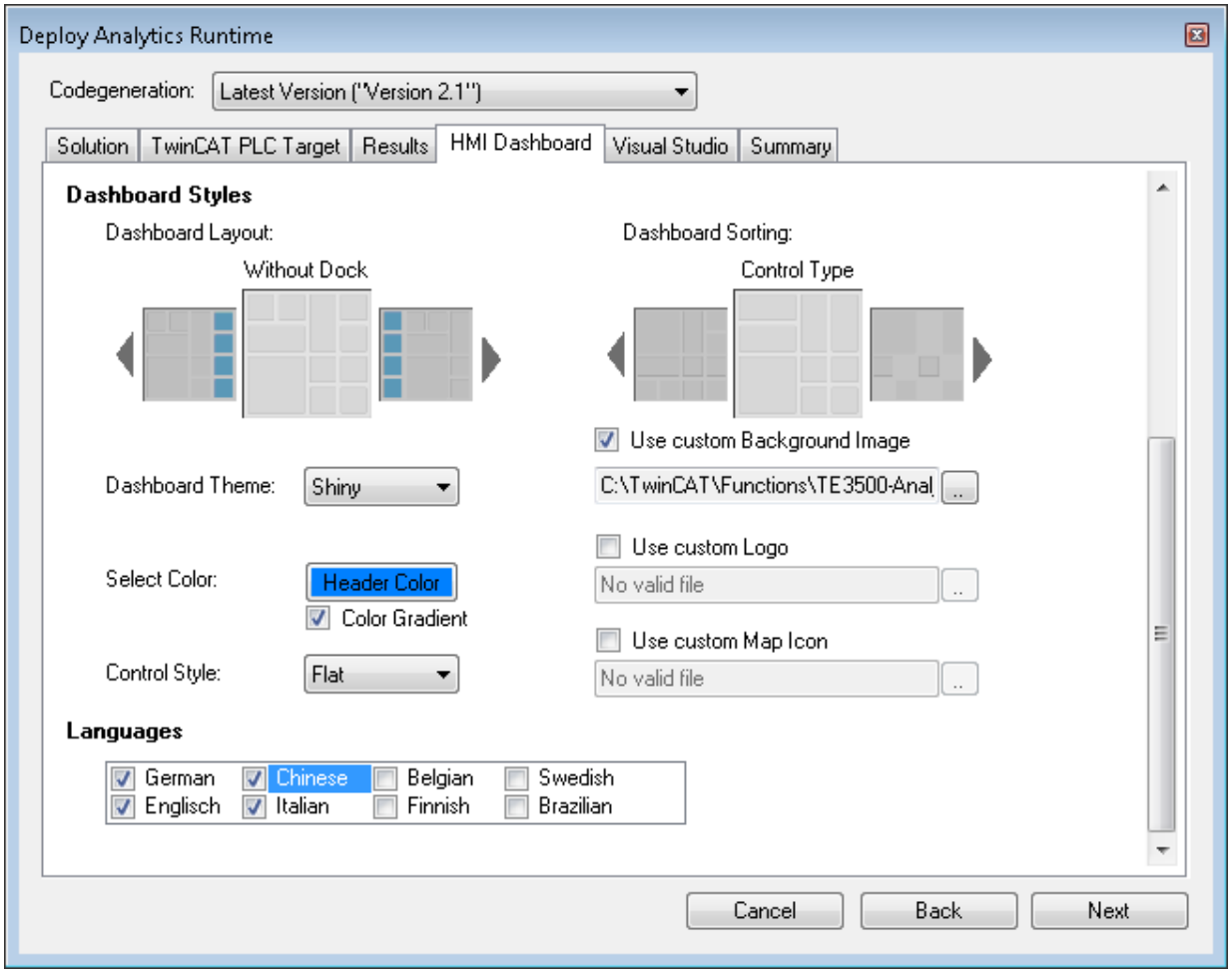
3. Especially for virtual machines, it is important to run the project on isolated cores, which is also an option in this tab. The next tab **Results** is needed only if you have selected the **Stream Results** option in the algorithm properties. If you want to send results, you can decide here in which way (locally in a file/ through MQTT) and which format (binary/JSON) this should be done. This is also generated automatically and executed immediately after activation.



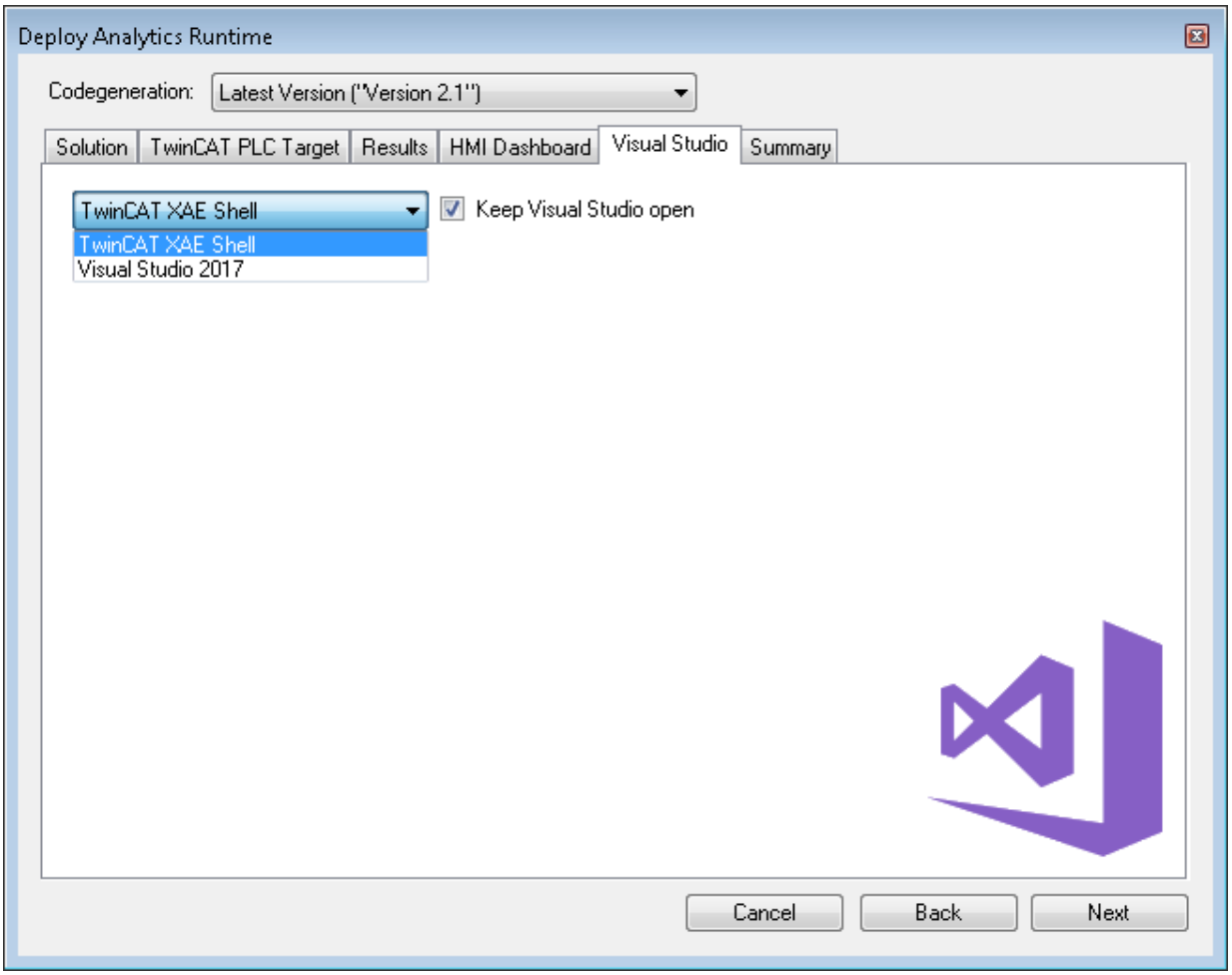
Downsampling of the results is possible by specifying a cycle time. The next tab is for the **HMI Dashboard**. A prerequisite for the automatic generation of the dashboard is the selection of HMI Controls for the corresponding algorithms whose results are to be displayed in the dashboard.



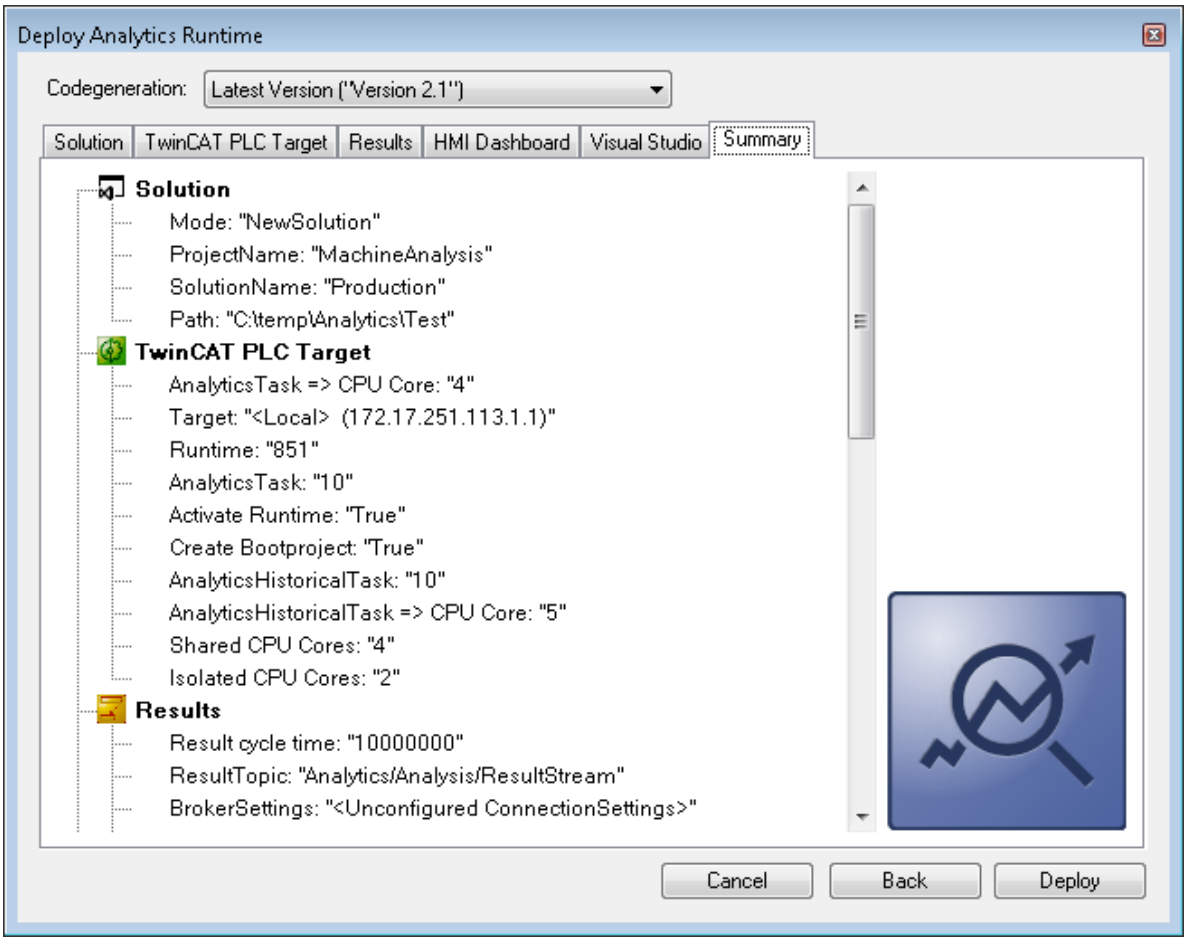
4. You can choose different options for your Analytics Dashboard, such as a start page with a map, layouts, sorting algorithms, custom colors and logos. If you select multiple languages for the Analytics Controls, a language switching menu will also be generated.



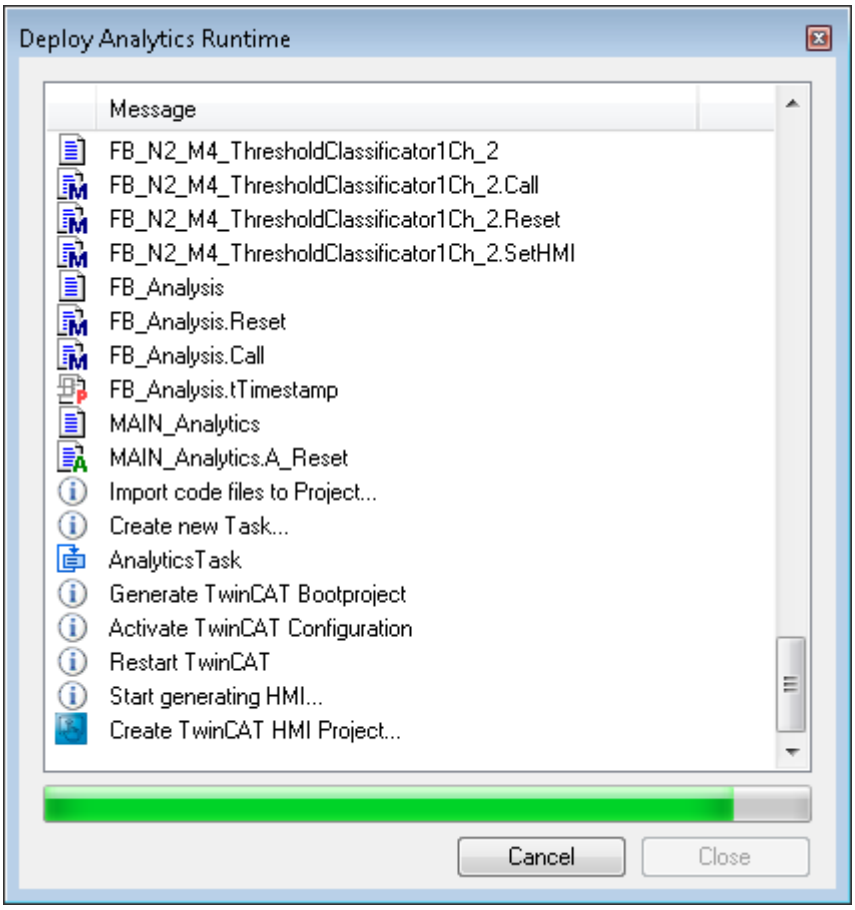
5. Select one of the installed versions of Visual Studio® and, whether the instance should start visibly or just be set up and activated in the background.

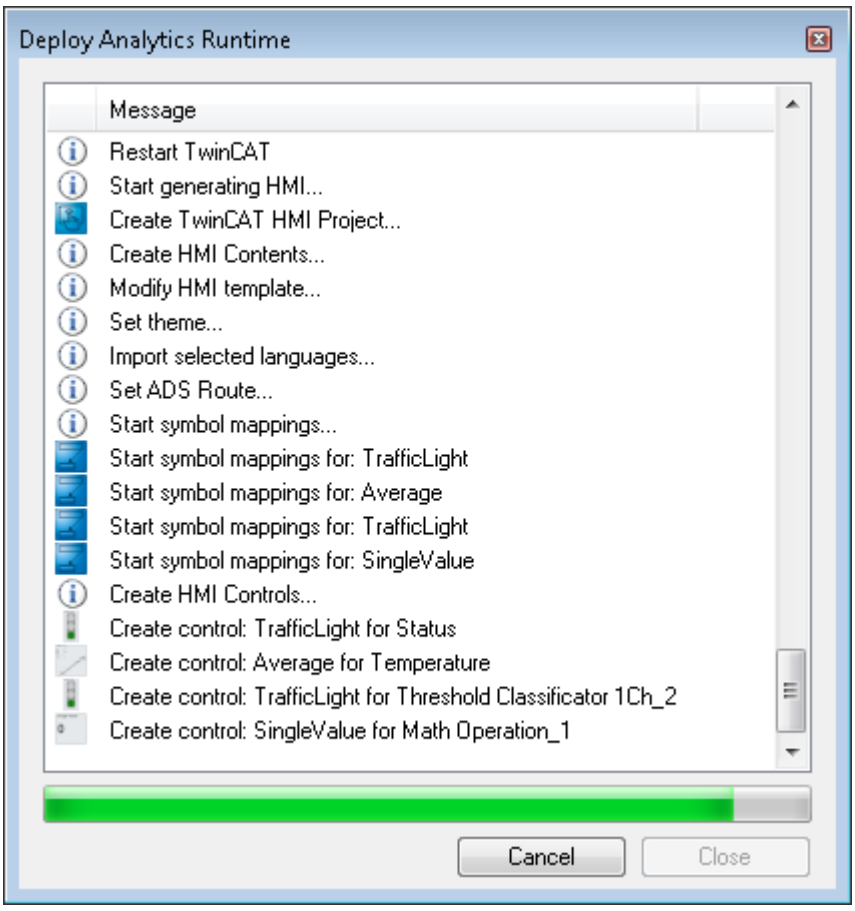


⇨ At last you can find an overview.

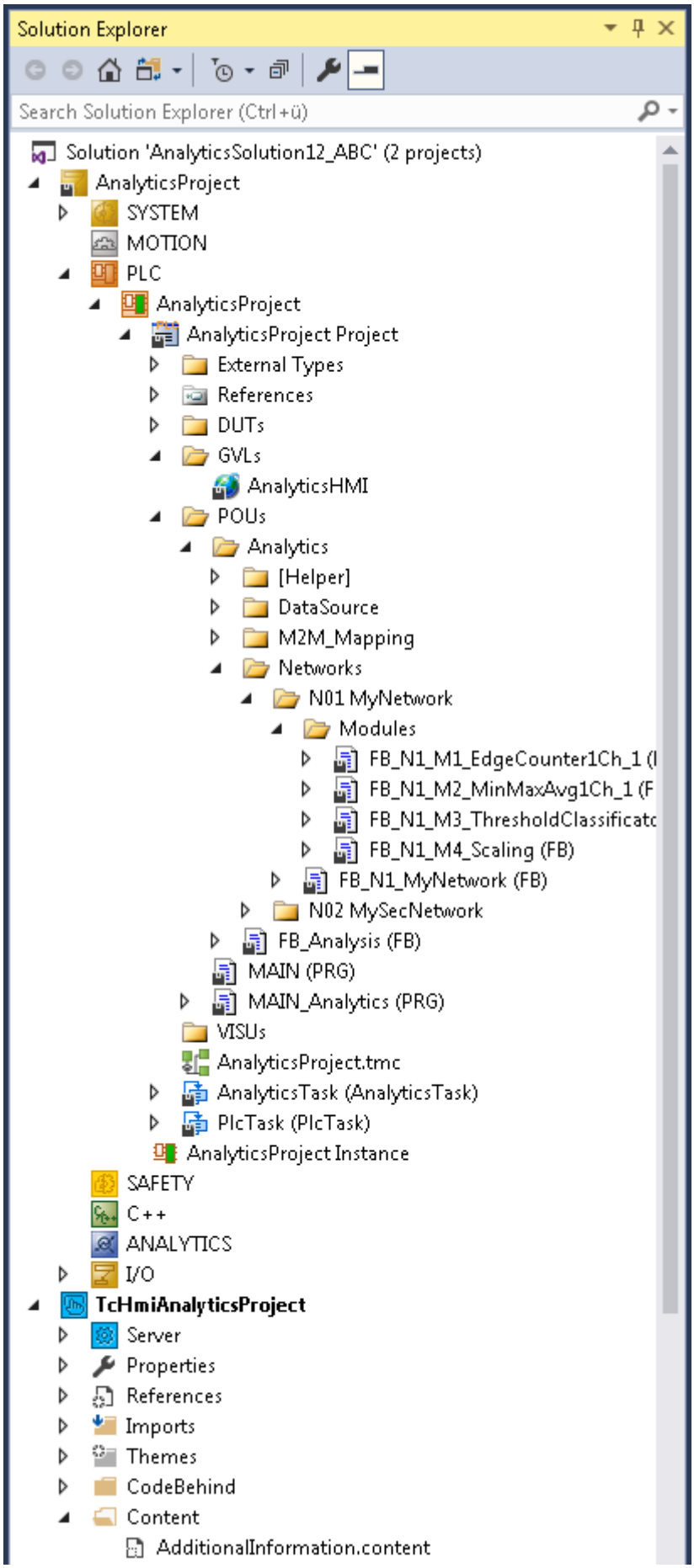


6. Now you can click the **Deploy** button to start the generation process. The PLC project and the HMI dashboard are now generated.





⇨ After the "Deploy Runtime succeeded" message, you will find a new Visual Studio®/XAE shell instance on your desktop. The new Solution and both projects are created.



5 Technical introduction

The basic idea of the TwinCAT Analytics Storage Provider (ASP) is to have a gateway that largely frees the user from configuring a data sink, i.e. a storage or a database. The user does not need to set up his own table structure in a database. He only has to configure which of the supported data sinks he wants to use for storing his data.

Service Management

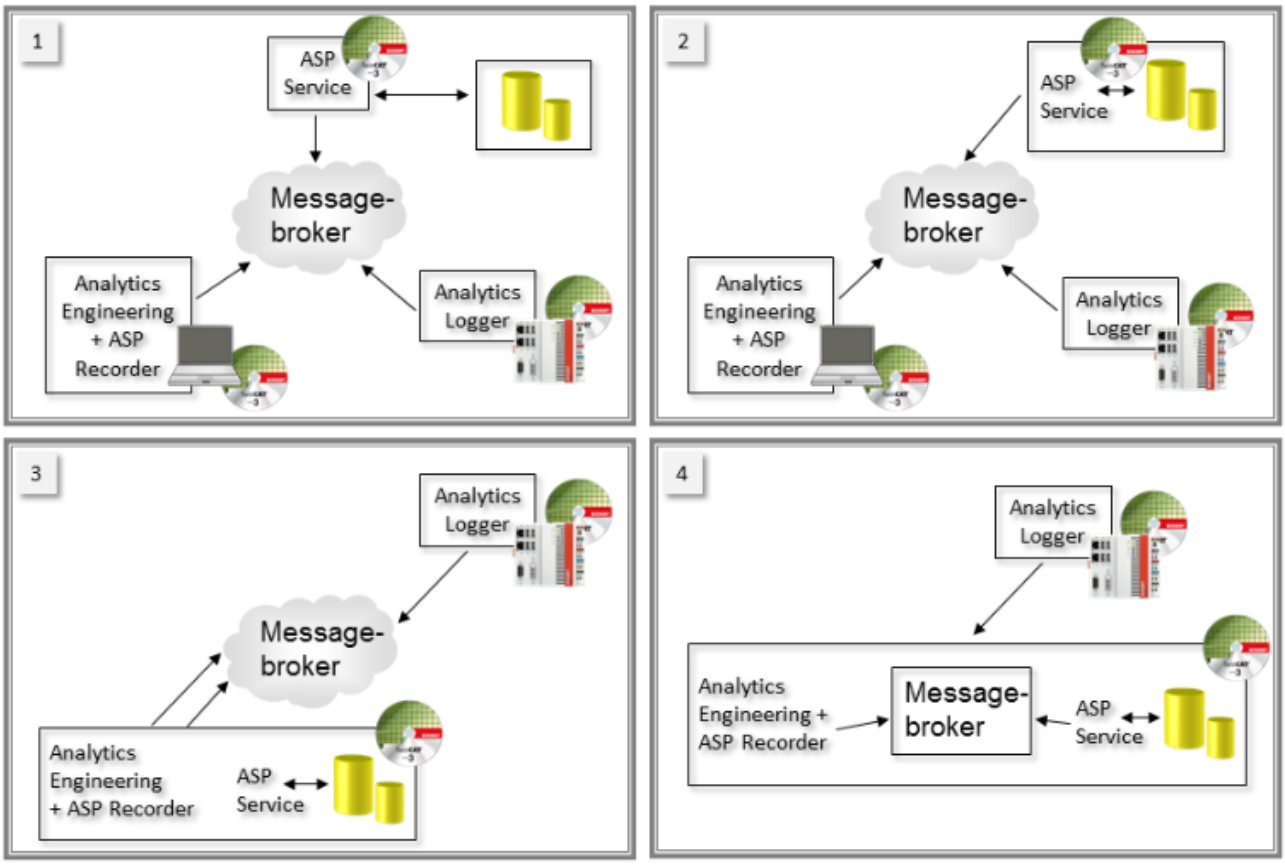
The Analytics Storage Provider service can run anywhere on the network. It is implemented as a Windows service. The service can run on hardware devices, such as industrial PCs or embedded PCs in the local network, and also on virtual machines in the same network, or in a cloud system, for example.

Data Management

The Storage Provider works with the binary format of TwinCAT Analytics. This allows it to receive and store streams from an MQTT message broker and to create and send new streams itself. The user only needs the recorder, which is integrated with the TwinCAT Analytics Workbench or the service tool in his own engineering system. The variables themselves are displayed in the TwinCAT Target Browser. For the Analytics binary format, they are divided into live and historical data. Live data can be used as input to the Analytics Storage Provider. Historical data are the values from the database/storage provided by the Storage Provider.

Topologies

The many degrees of freedom offered by IoT technologies enable different topologies. The following picture shows the most important constellations.



1. Each SW package runs on its own HW device or virtual machine.
2. The Analytics Storage Provider Windows service runs on the same device as the database/storage.
3. Analytics Engineering, Analytics Storage Provider, and database or storage are on the same device. Only the Message Broker and Analytics Logger (data source) run on other devices.

4. In this topology view, only the Analytics Logger runs on its own PC. This may be the case in a machine application. All other tools in the Analytics tool chain reside on one device, including the MQTT message broker.

Topologies with additional ASP clients

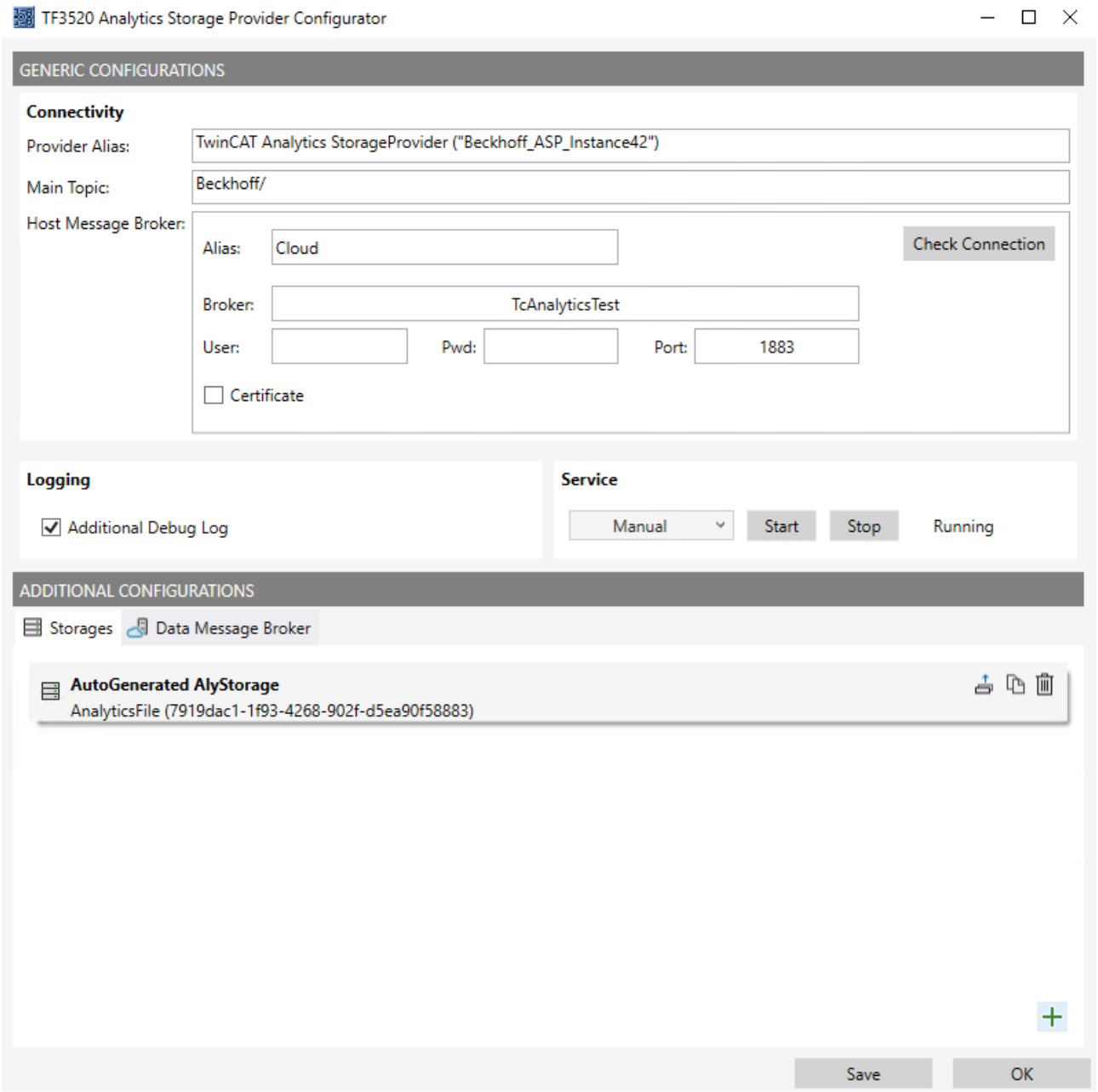
Currently, two additional clients are available from the Analytics Storage Provider perspective. A command line based client that allows execution from almost any application. And a PLC library that can also be used to influence the actions of the storage provider.

6 Configuration

The configuration of the Analytics Storage Provider is divided into two main parts. First, you need to configure the service with its stores. This is done in the TwinCAT Analytics Storage Provider Configurator. You must also configure the recordings and pipelines yourself. In other words, which variables should be stored in which stores under which conditions. To do this, go to the TwinCAT Analytics Storage Provider Manager. In this chapter you will also find the supported databases and storages.

6.1 Configurator

You can configure the service with its message brokers and stores in the TwinCAT Analytics Storage Provider Configurator. A distinction is made between a Host Message Broker and various Data Message Brokers. A Host Message Broker can also be a Data Message Broker at the same time. The Host Message Broker is special because this is where the Storage Provider's service registers.



6.1.1 Generic Configurations

General settings for the Storage Provider service can be made in the Generic Configurations.

GENERIC CONFIGURATIONS

Provider Alias:
Each Analytics Storage Provider Service has its own GUID for identification. You can enter a provider alias in the general configuration area so that this can be replaced by a meaningful name.

Main Topic:
Basically, you do not have to worry about the topic. However, you can enter the so-called Main Topic here. It describes the first part of the overall topic. Beckhoff-specific additions are then added. This results in a very simple plug-and-play system with TwinCAT Analytics.

Host Message Broker:
The so-called Host Message Broker can be configured here. The Storage Provider Service logs on to this message broker itself and makes its data available. This broker can also be a so-called Data Message Broker, which is used to receive data to the Storage Provider Service. There can only be one Host Broker in the system, but several Data Brokers.

Logging:
Check various logging options.

Service:
Displays the status and default settings for starting the Windows service.

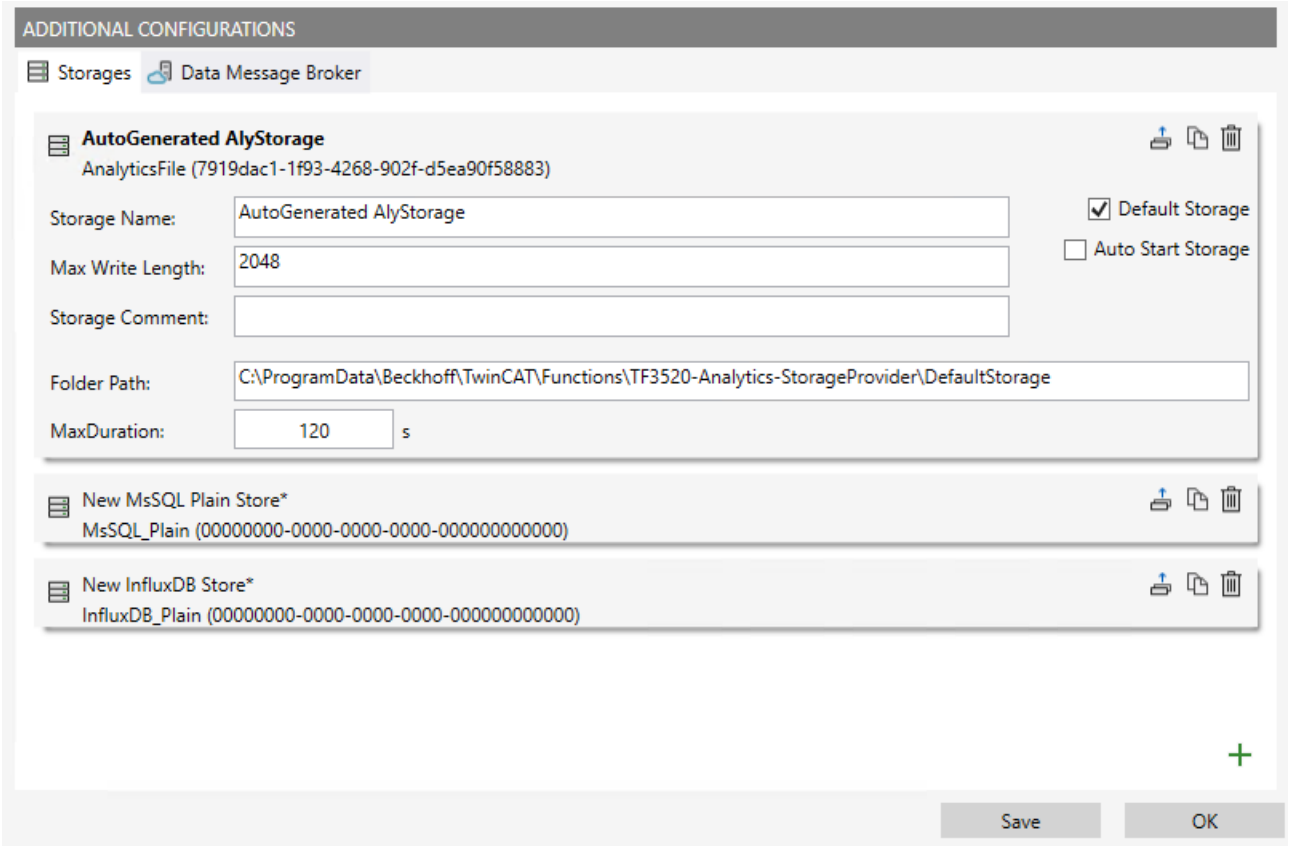
6.1.2 Additional Configurations

The Stores and Data Message Broker of the Storage Provider Services are set under Additional Configurations.

ADDITIONAL CONFIGURATIONS

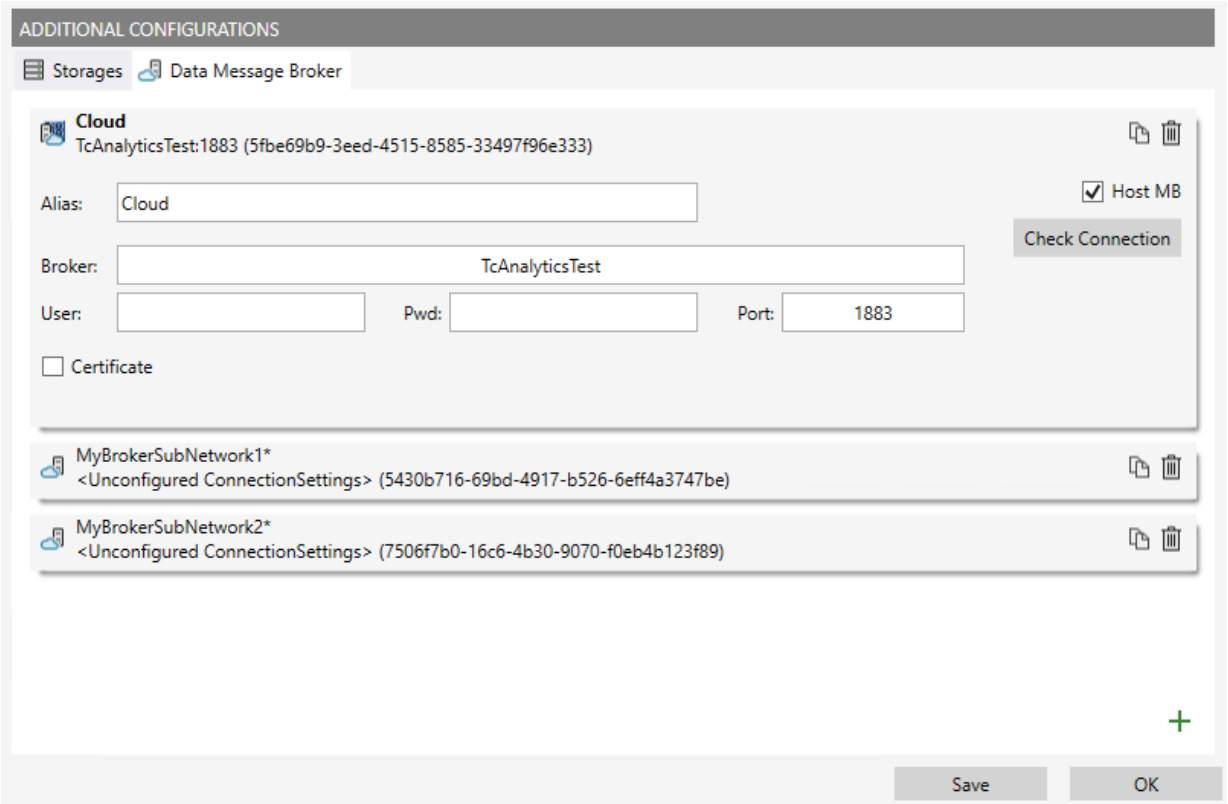
Storages

The stores that are available to the Analytics Storage Provider Service can be created in the Storages tab. These can be of various types, such as Microsoft SQL® or csv files. New stores can be added in binary or plain text format using the plus sign in the bottom right-hand corner. The configuration of the individual stores is described in this sub-chapter [▶ 47].

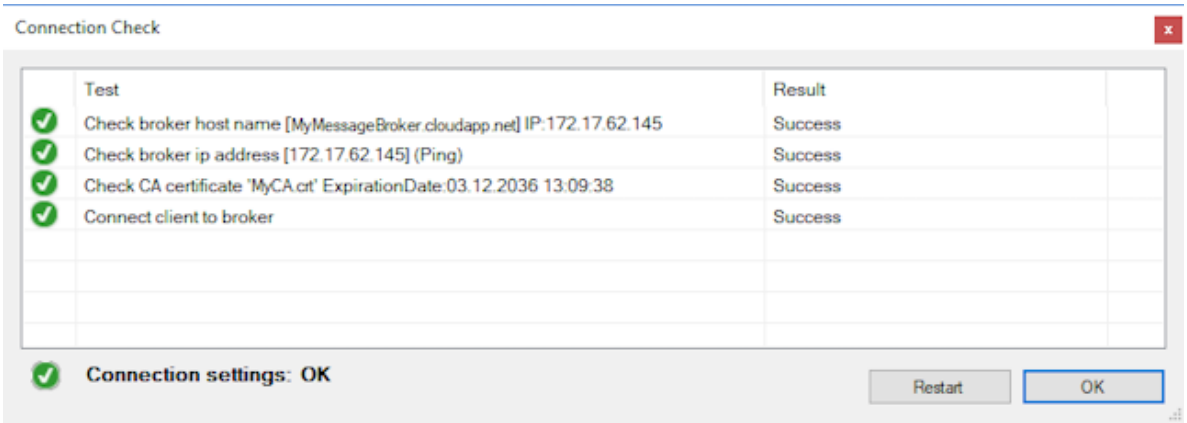


Data Message Broker

1. In the Data Message Broker tab, you can create the message brokers that can provide input data for the Storage Provider in addition to the Host Message Broker.



2. The settings can be checked using the Check Connection button.
   ⇨ The result is displayed in the following window:



If you click on **Save**, the settings are saved in the directory *C:\TwinCAT\Functions\TF3520-Analytics-StorageProvider\Configurations*.

6.1.2.1 Databases/Stores

The following overview shows which database connections are supported by which platform.

Database	Windows		TwinCAT/BSD	
	Local	Remote	Local	Remote
Analytics File	X	X	X	X
CSV	X	X	X	X
MS SQL	X	X	-	X
PostgreSQL	X	X	-	X
InfluxDB 2	X	X	-	X
AzureBlob	-	X	-	X

6.1.2.1.1 TwinCAT Analytics Binary File

TwinCAT Analytics Binary File is a TwinCAT-specific Storage. Therefore no external software is necessary. You can use this type of storage directly after installing the Analytics Storage Provider. This is the same file that the TwinCAT Analytics Logger provides in its “offline” configuration without MQTT Message Broker.

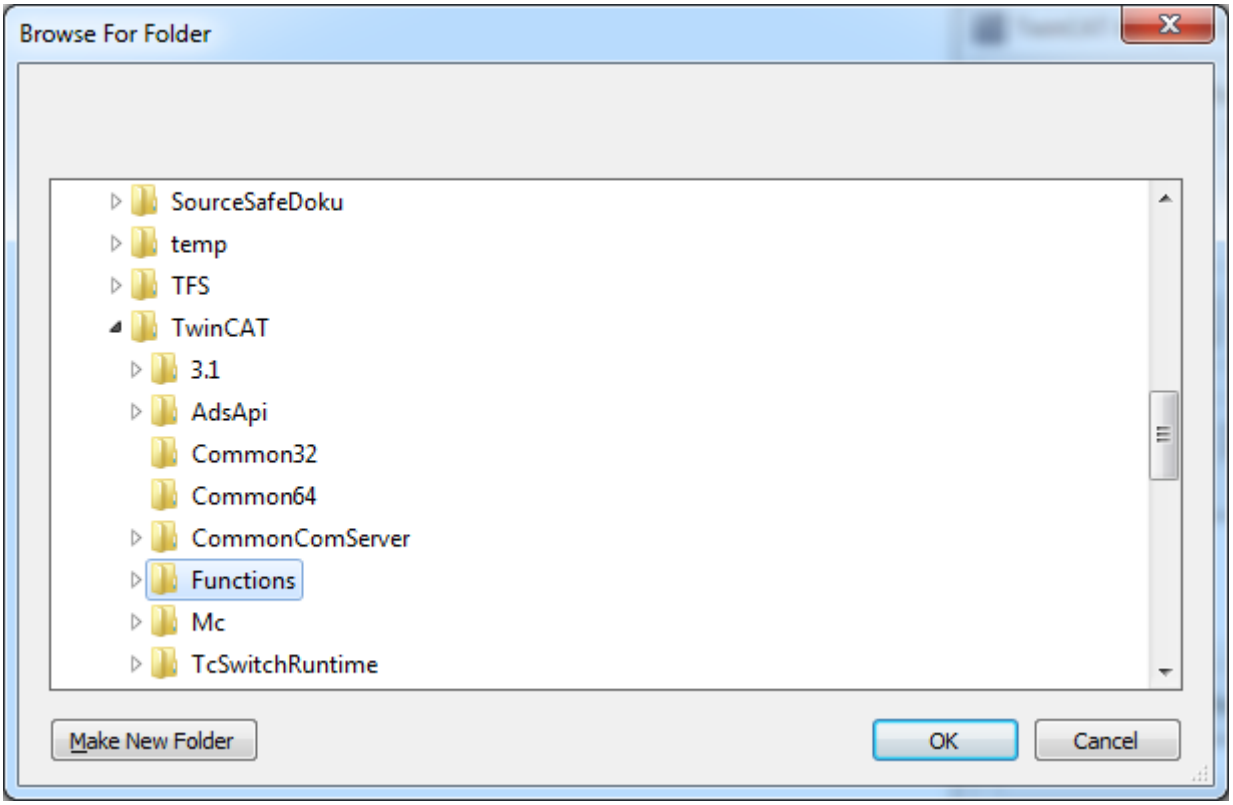
Fig. 1:

Storage Name:
Assign a descriptive name that describes the purpose of the storage well. It will reappear in various places during configuration in the Manager.

Max Write Length:
The amount of data that is written to the .tay file in one call is specified here.

Storage Comment:
You can enter additional information about the storage here.

File Path:
For the configuration, you must select your preferred folder on the local device on which the Storage Provider is running.



The folder used is displayed in the Connection String window for confirmation.

The folder architecture that is created after the pipeline is started is currently divided into five hierarchical levels:

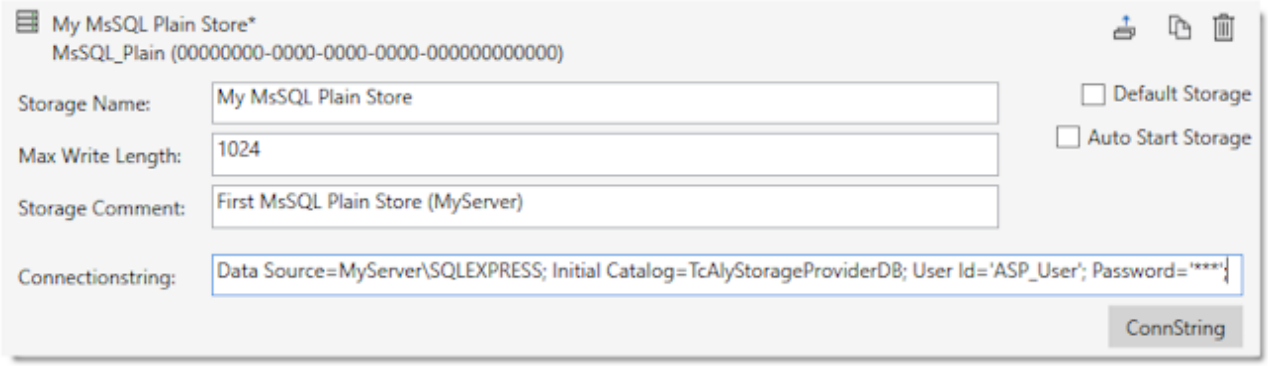
1. System ID (this is a GUID of the system sending the data) - can be replaced by the alias name in the Analytics Logger settings or in local engineering by the Machine Administration Page.
2. Recording name - can be set using the configurator.
3. Record name - can be set by the configurator and automatically influenced with auto IDs, date, cycle time and other placeholders at runtime.
4. Internal ID - cannot be changed.
5. Layout GUID - cannot be changed and corresponds to the data storage of the Analytics Logger.

Max Duration:
This value specifies in seconds how much data is written to a .tay file. After time X, the data is written to a new .tay file. Ring buffers that are configured in the Manager only affect complete .tay files of a current recording.

6.1.2.1.2 Microsoft SQL (binary / plain text)

With Microsoft SQL Server, you have another On-Premises solution for storing the Analytics binary data.

Enter the Connection String for your MS SQL server here.

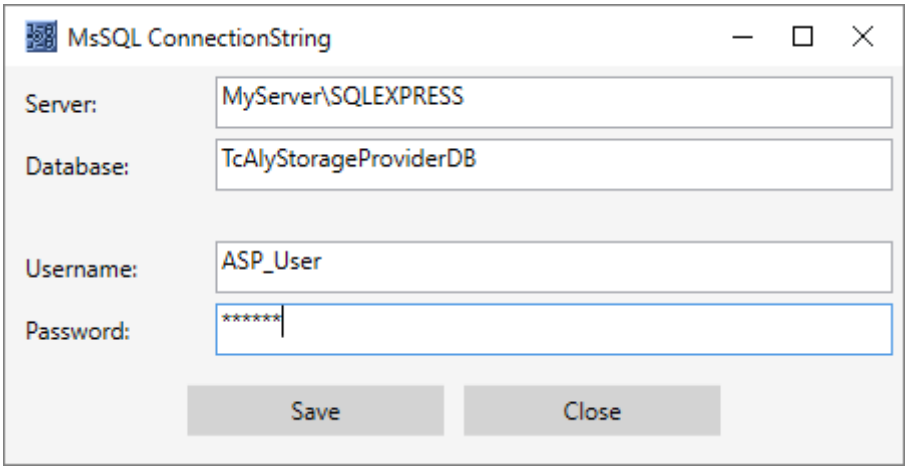


Storage Name:
Assign a descriptive name that describes the purpose of the storage well. It will reappear in various places during configuration in the Manager.

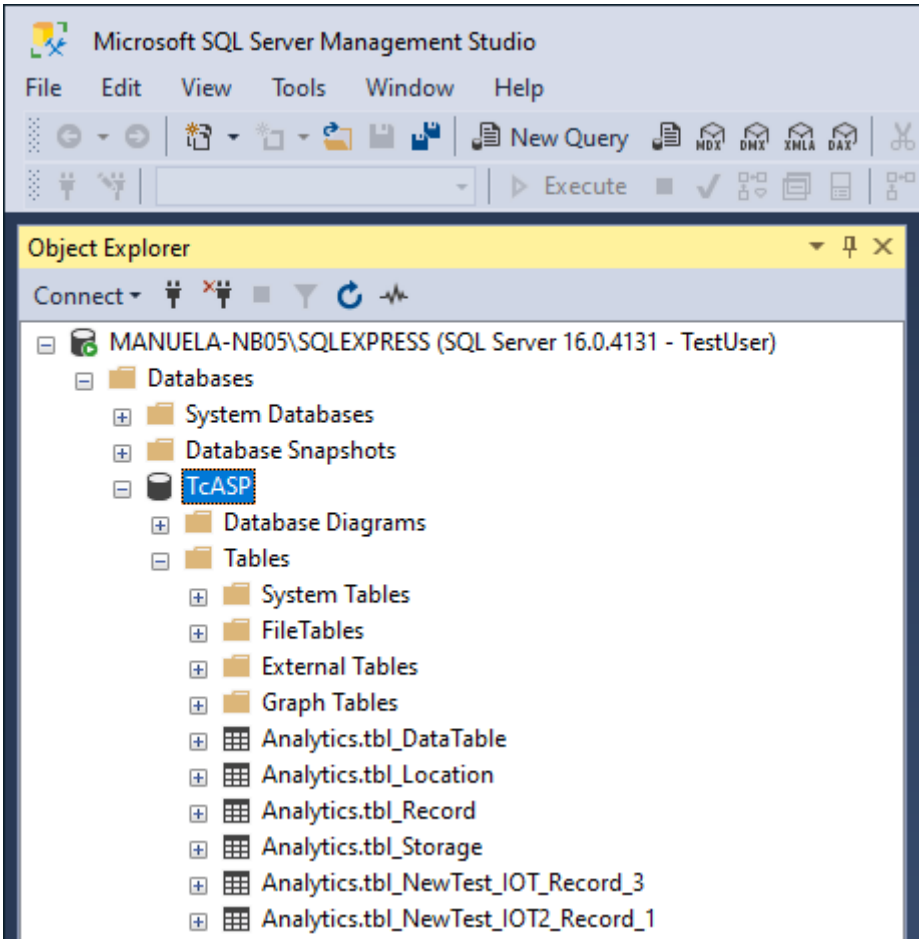
Max Write Length:
The amount of data that is saved in a tbl_Data record is specified here.

Storage Comment:
You can enter additional information about the storage here.

Connection String:
Click on the **ConnString** button to open the input mask. Make the configuration settings there, including for remote databases that are accessible via network connections.



After starting the storage, communication with the database begins. At this point, the Storage Provider itself creates the four required tables. Each recording configuration is saved in a separate table. As an example, you can see the following screenshot from Microsoft SQL Server Management Studio.



The ring buffer functionality for the Microsoft SQL® database can also be set in the Manager.

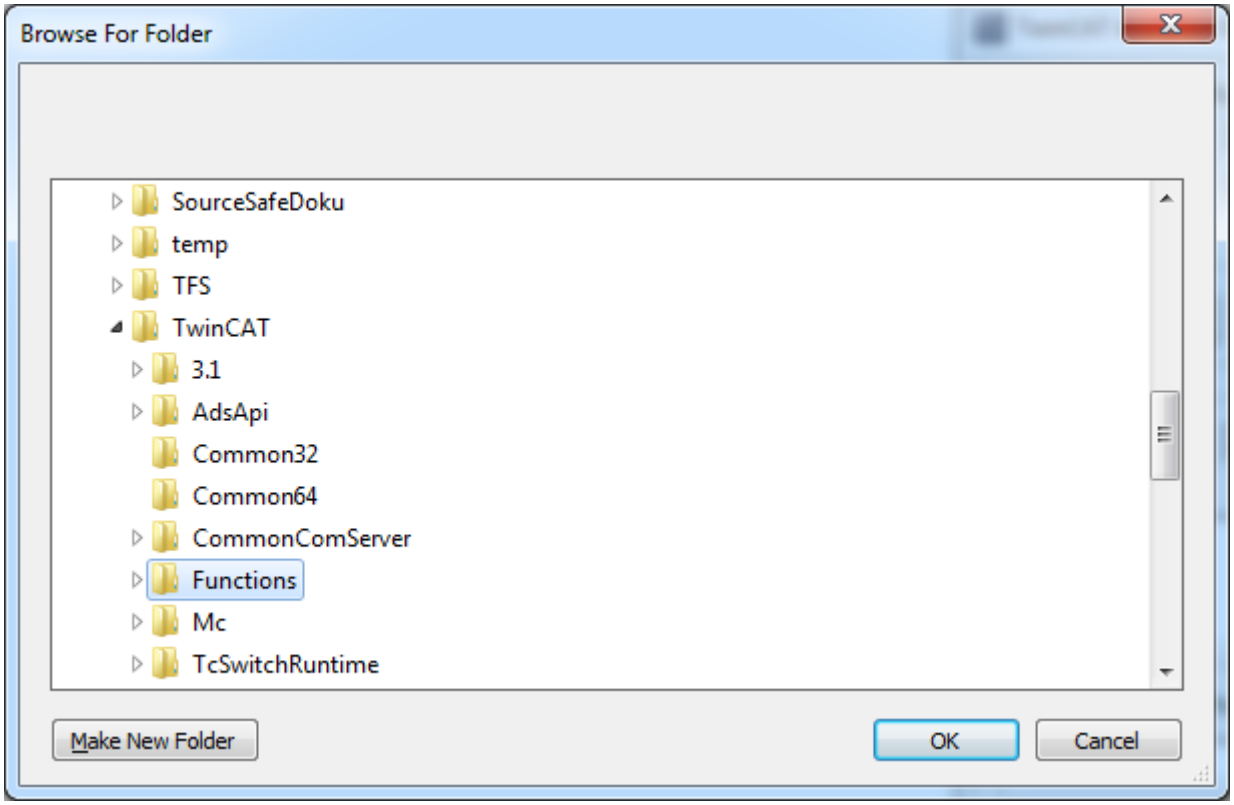
6.1.2.1.3 CSV file

Storage Name:
Assign a descriptive name that describes the purpose of the storage well. It will reappear in various places during configuration in the Manager.

Max Write Length:
The amount of data that is written to the .csv file in one call is specified here.

Storage Comment:
You can enter additional information about the storage here.

File Path:
For the configuration, you must select your preferred folder on the local device on which the Storage Provider is running.



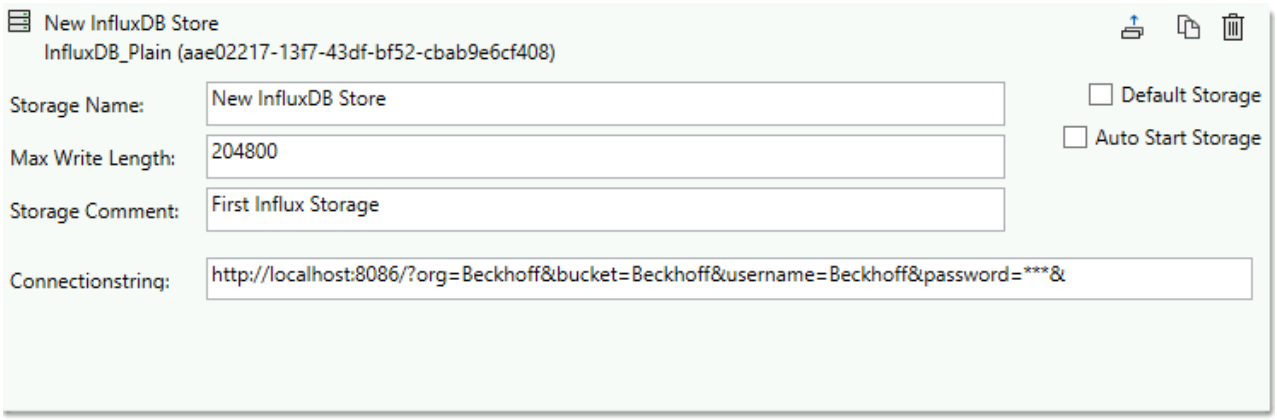
The folder used is displayed in the Connection String window for confirmation.

Max Duration:
It is also possible to define the timespan to be saved in a CSV file. The decimal places of floating point numbers can also be limited.

Decimalplaces:
The number of decimal places can be set here. The value "-1" stands for unspecified, the value "2" stands for two decimal places, for example.

6.1.2.1.4 InfluxDB

With the support of InfluxDB, you have another on-premises solution for storing Analytics binary data.

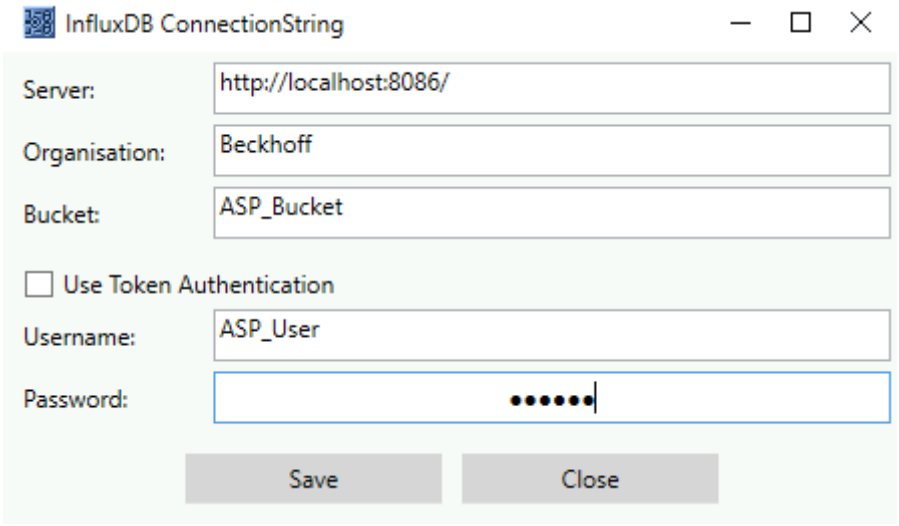


Storage Name:
Assign a descriptive name that describes the purpose of the storage well. It will reappear in various places during configuration in the Manager.

Max Write Length:
The amount of data that is written to the Influx in one call is specified here.

Storage Comment:
You can enter additional information about the storage here.

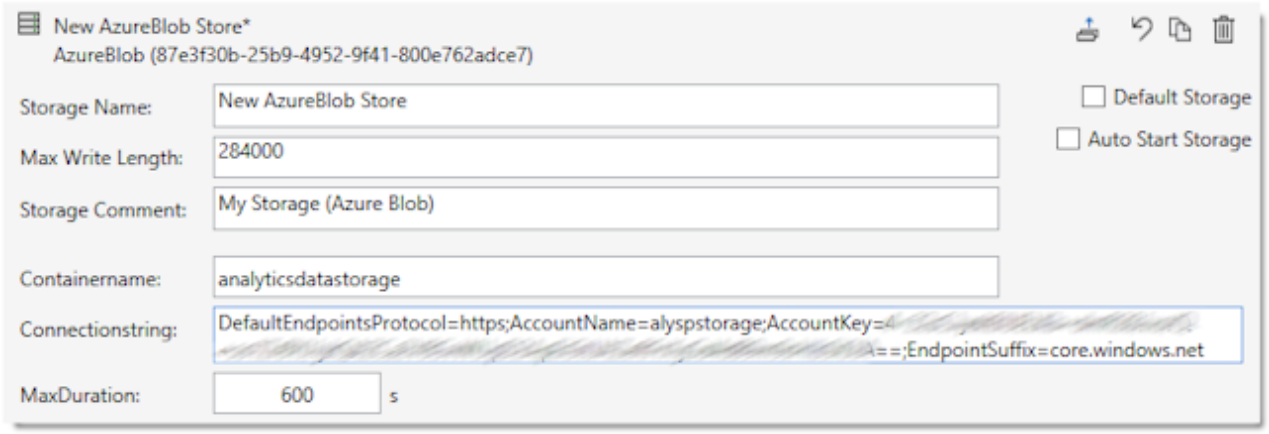
Connection String:
Click on the **ConnString** button to open the input mask. You can make the configuration settings there, including for remote databases that are accessible via network connections. The server should always be specified with the port number.



After starting the storage, communication with the database begins.

6.1.2.1.5 Microsoft Azure Blob

To use Microsoft Azure Blob Storage, you need a Microsoft Azure Cloud account. There you get also your individual Connection String for the configuration of the TwinCAT Analytics Storage Provider.



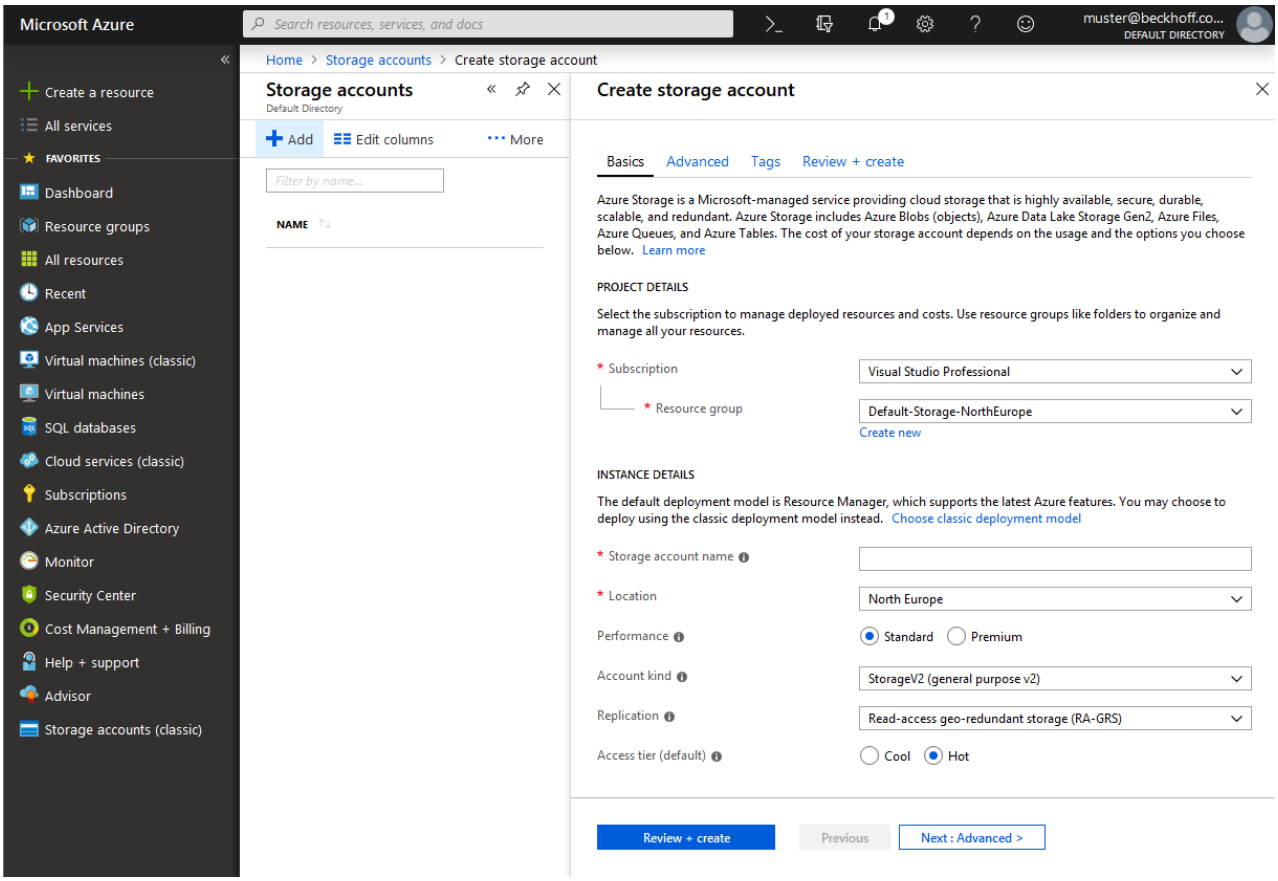
Storage Name:
Assign a descriptive name that describes the purpose of the storage well. It will reappear in various places during configuration in the Manager.

Max Write Length:
The amount of data that is written to the blob store in one call is specified here.

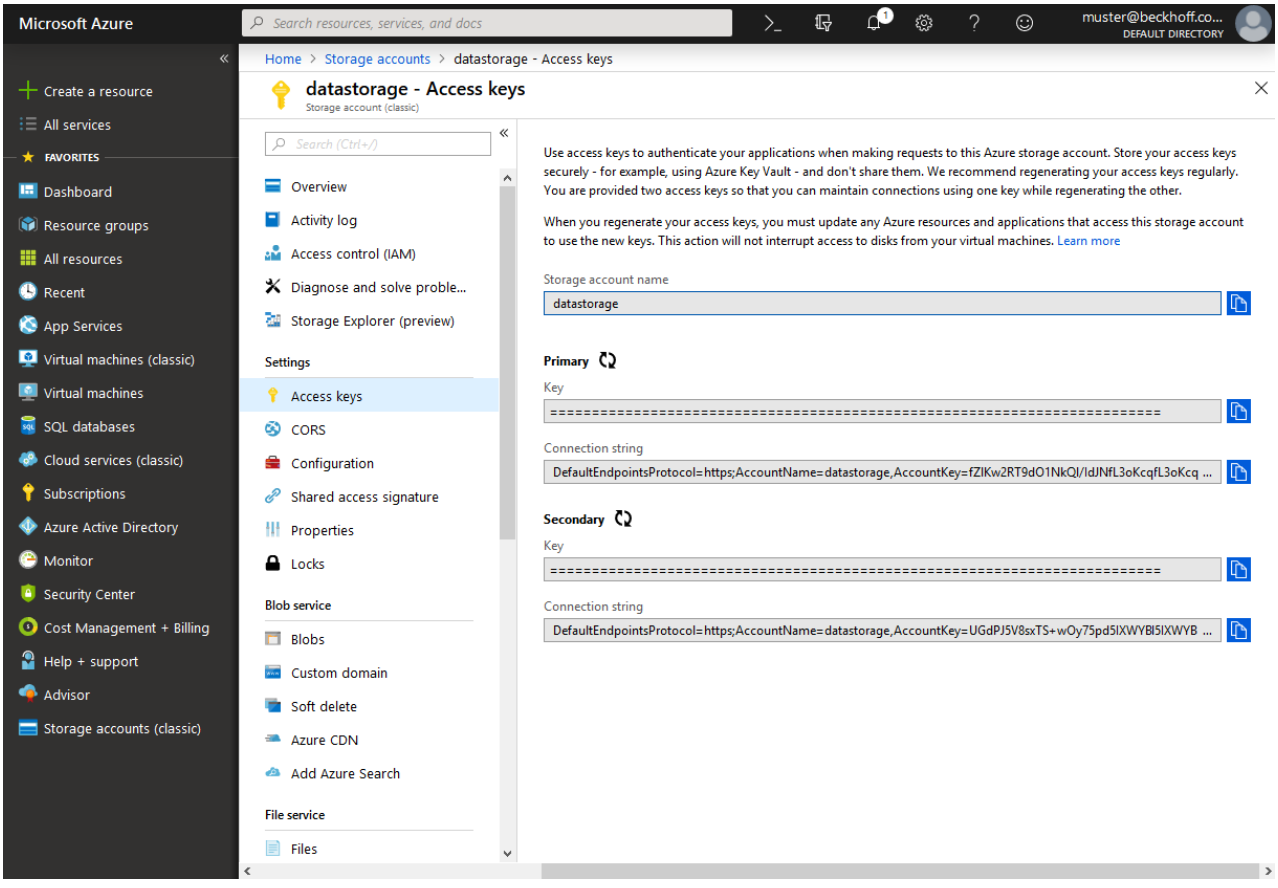
Storage Comment:
You can enter additional information about the storage here.

Copy the Connection String into the description field. The storage must be created in Azure itself.

Select **Storage accounts (classic)**.



After creating the Storage, you will find the secondary Connection String under **Access keys**. This string must be used in the configuration of the Analytics Storage Provider.



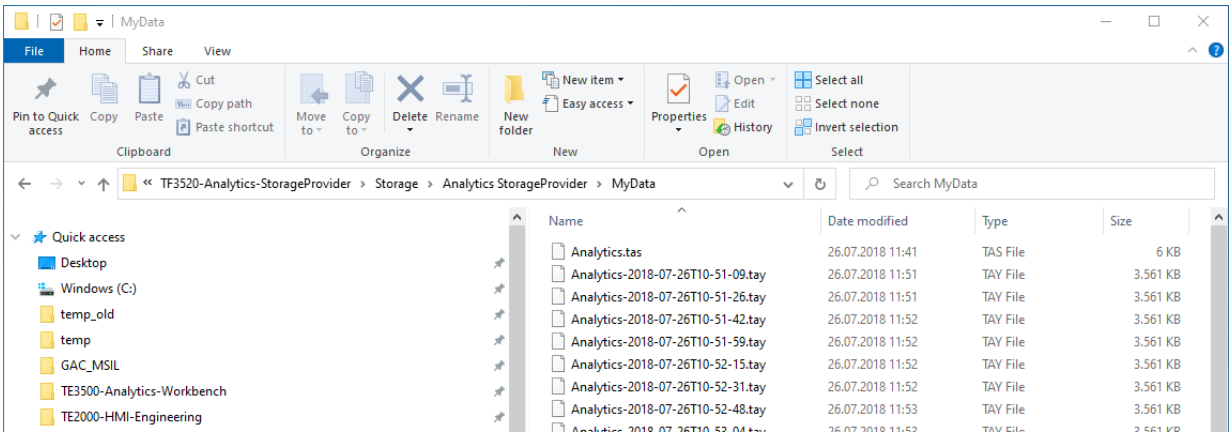
6.1.3 Importing/converting Analytics Files

In the following it is assumed that you have installed TwinCAT under "C:/TwinCAT". Otherwise, you must adjust the specified paths accordingly.

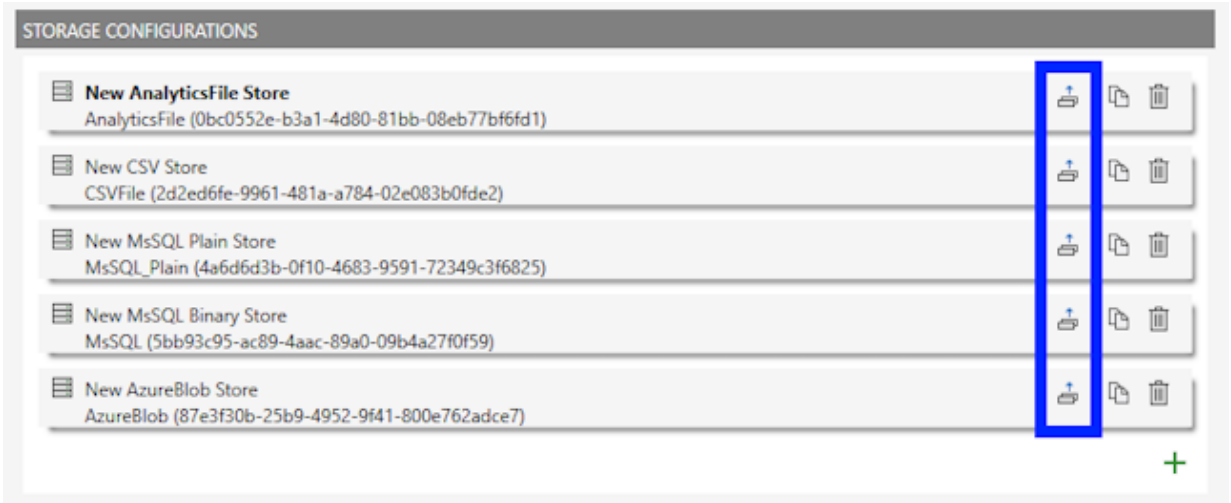
You can import recordings from the Analytics Logger stored in Analytics File Format (*Analytics.tas, Analytics-<Date>.tay*) into the Storage Provider. In general, you can convert data saved by the Storage Provider as an Analytics File into other formats. Analytics File is always the source format.

To do this, perform the following steps:

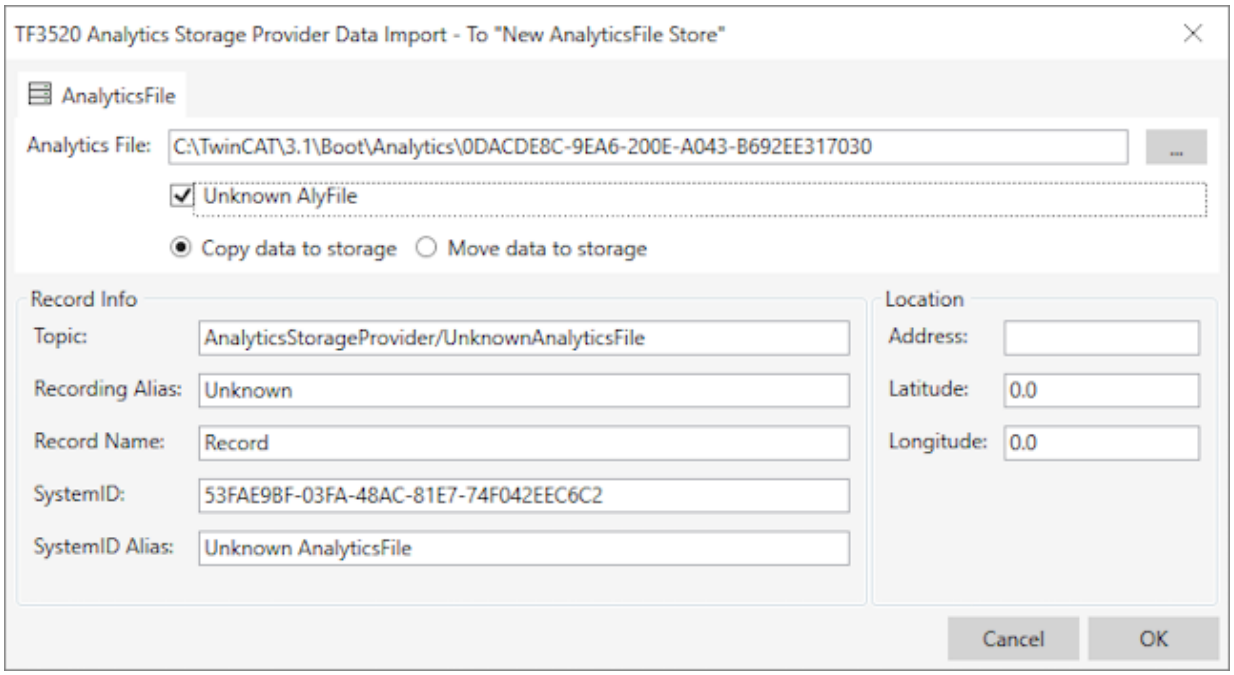
1. Save the folder with the Analytics Files in your Storage Provider location
   By default, here: *C:\TwinCAT\Functions\TF3520-Analytics-StorageProvider\Storage\Analytics StorageProvider* (create the folders manually if they do not exist)



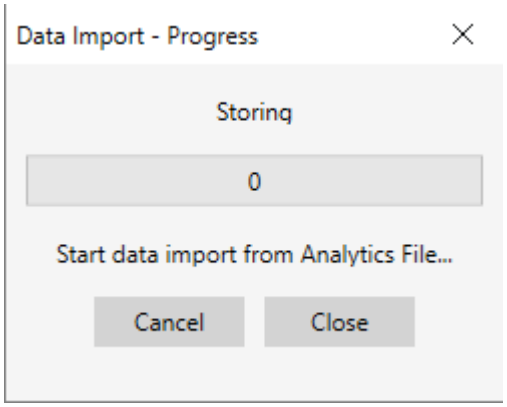
2. Open *TwinCAT.Analytics.StorageProvider.Configurator.exe*. The program can be found under the path *C:\TwinCAT\Functions\TF3520-Analytics-StorageProvider\Tools*
3. Select the Storage into which the Analytics File is to be imported and press the **DataImport** button.



4. Then select the path to the Analytics File in the DataImport dialog and enter all other known parameters for this recording.



5. Press **OK** and the data import begins.

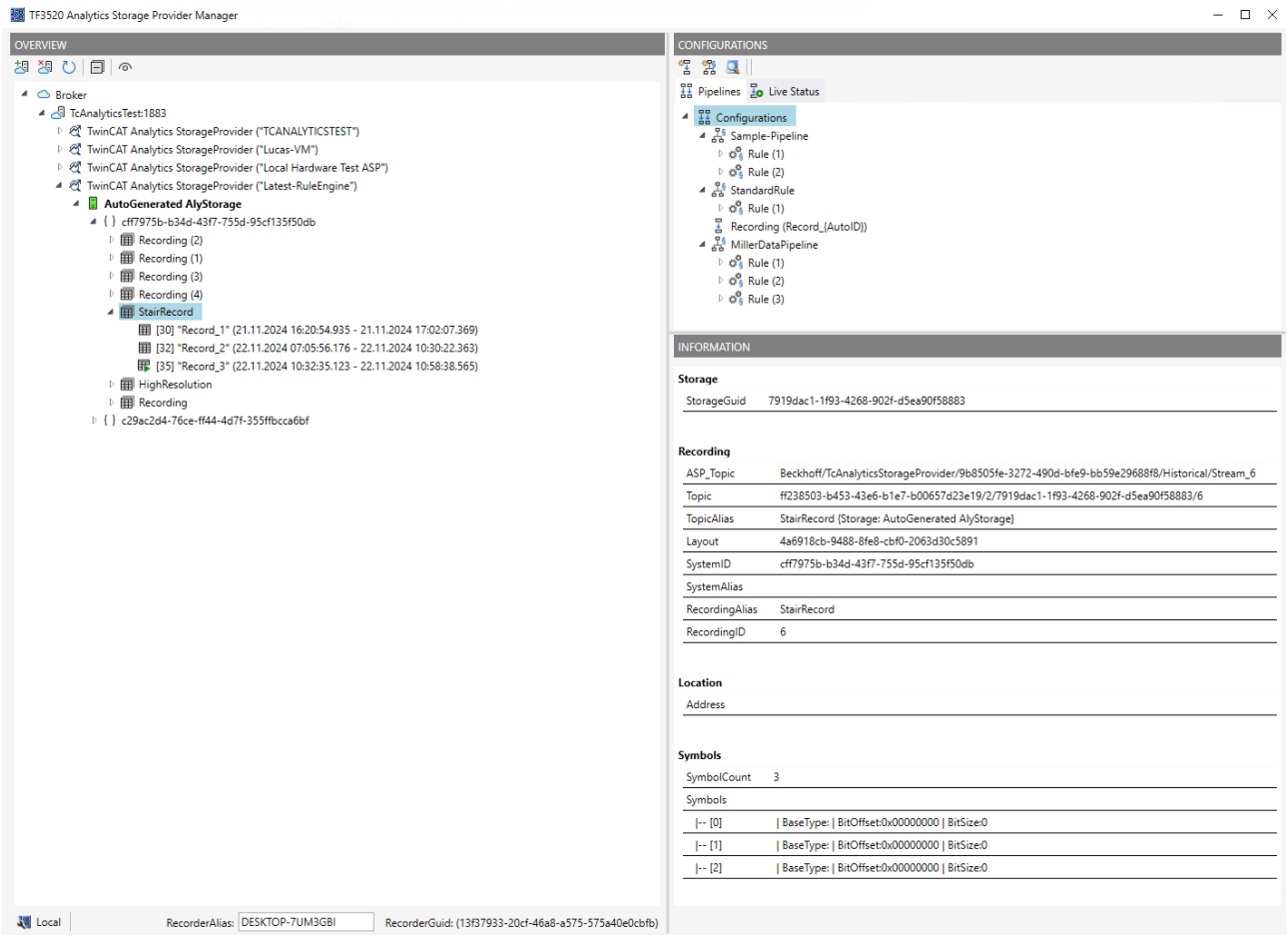


⇨ Now you can see your imported data in the TwinCAT Target Browser [▶ 95].

⇨ You may have to wait a short time or restart your Storage Provider.

6.2 Manager

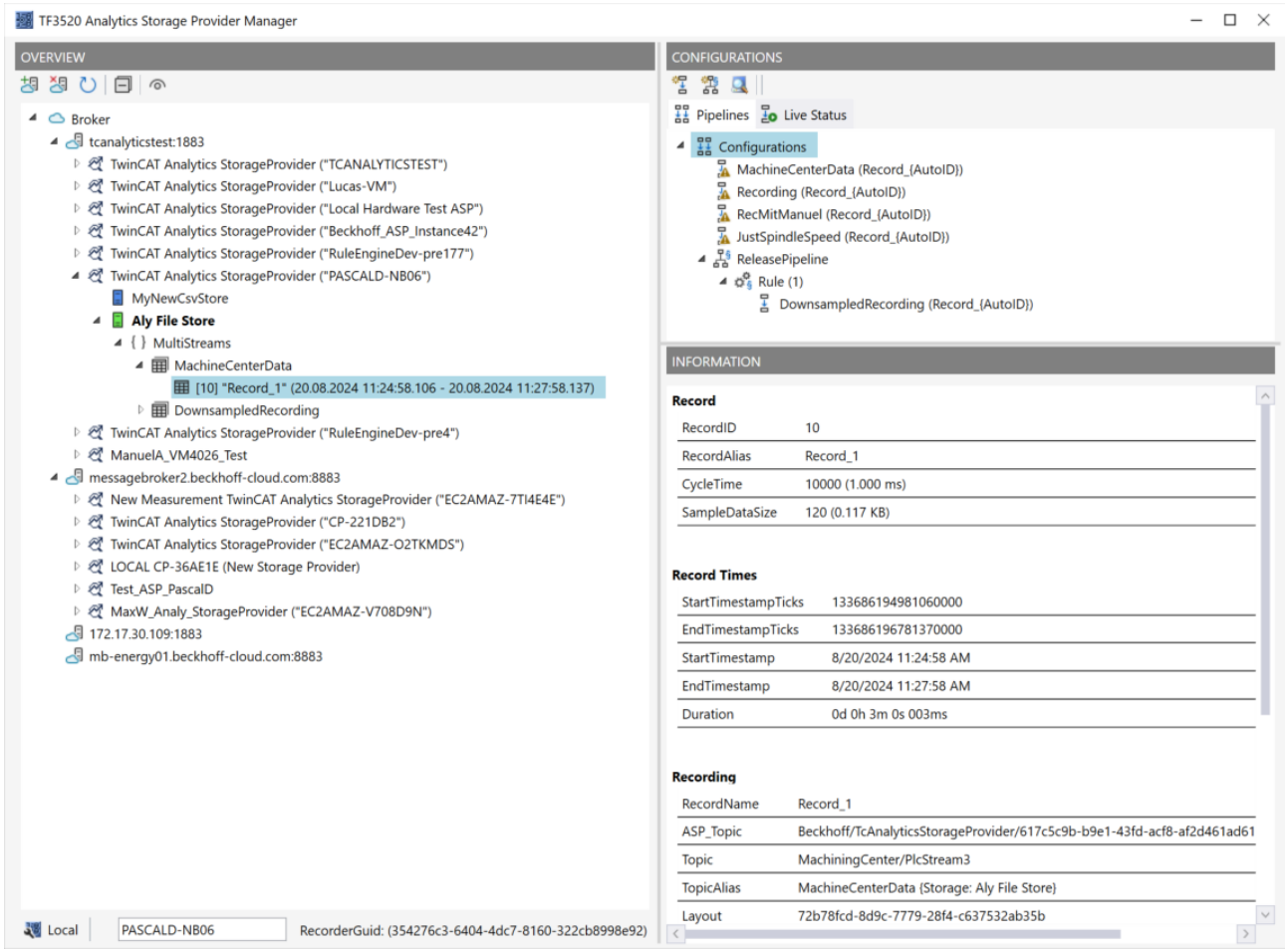
You can define your data pipelines in the TwinCAT Analytics Storage Provider Manager. A pipeline is defined in such a way that you can select the data sources (Data Message Broker) and the symbols available there. Optionally pre-process the data using rules and ultimately store the data in one or more stores. The stores must have been created in the Configurator [▶ 45] for this.



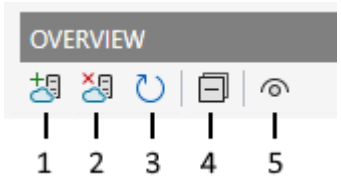
6.2.1 Manager ("Recorder")

The Analytics Storage Provider Recorder is part of the Analytics Engineering setups. Therefore, you can find the recorder in the installation of the TwinCAT Analytics Workbench and the TwinCAT Analytics Service Tool.

With the Storage Provider Recorder recording definitions can be created, started and managed. In addition, it is possible to manage the data memories of individual Analytics Storage Providers. All important properties of the found Analytics Storage Providers and historized data are clearly displayed.



Toolbar Manager window ("OVERVIEW")

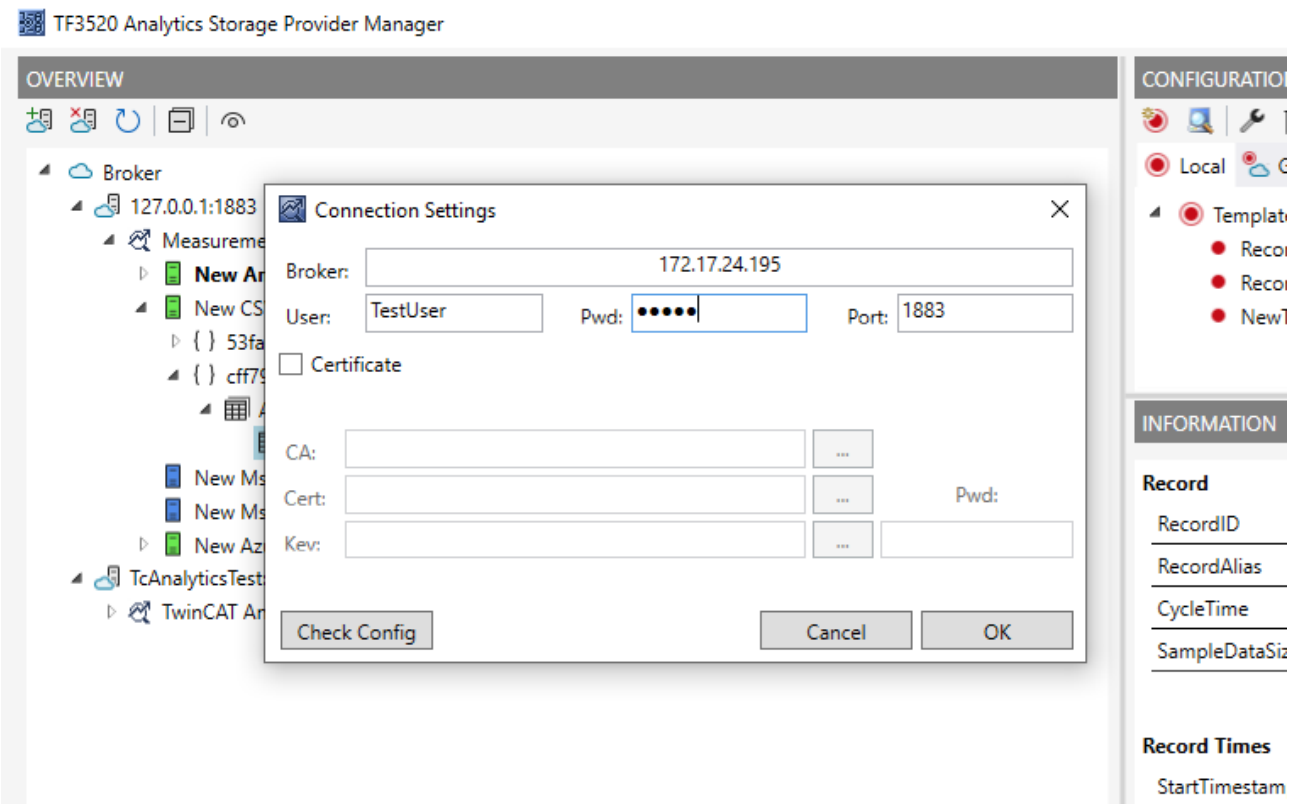


1	Add new broker
2	Remove selected broker
3	Refresh display
4	Collapse all nodes
5	View switch between dark/light mode

Function Manager window ("OVERVIEW")

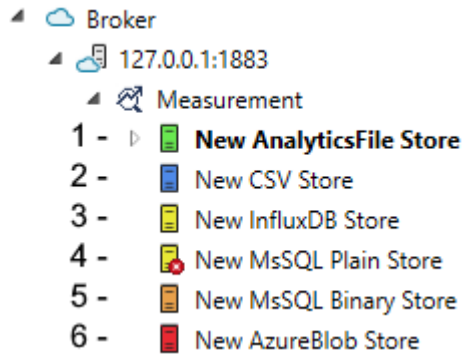
First assign a **RecorderAlias**. This helps to group the started recordings and to find its self started ones again.

After that, one or more brokers can be set up. This is done via the already known input mask for MQTT connection properties.



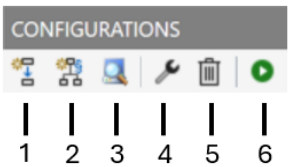
Once a connection to the broker could be established, all Analytics Storage Providers connected to it will be listed.

"Storage" status



1	Storage Online
2	Storage Offline
3	Storage starts
4	Storage starts with error. Still trying to start it
5	Storage is shut down
6	Storage is in the error state

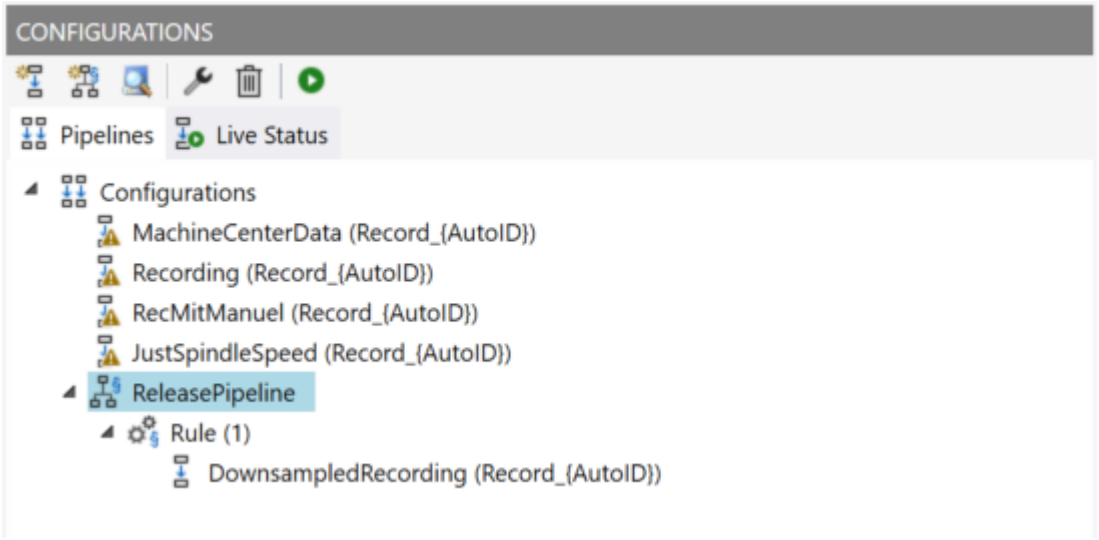
Toolbar Manager window ("CONFIGURATIONS")



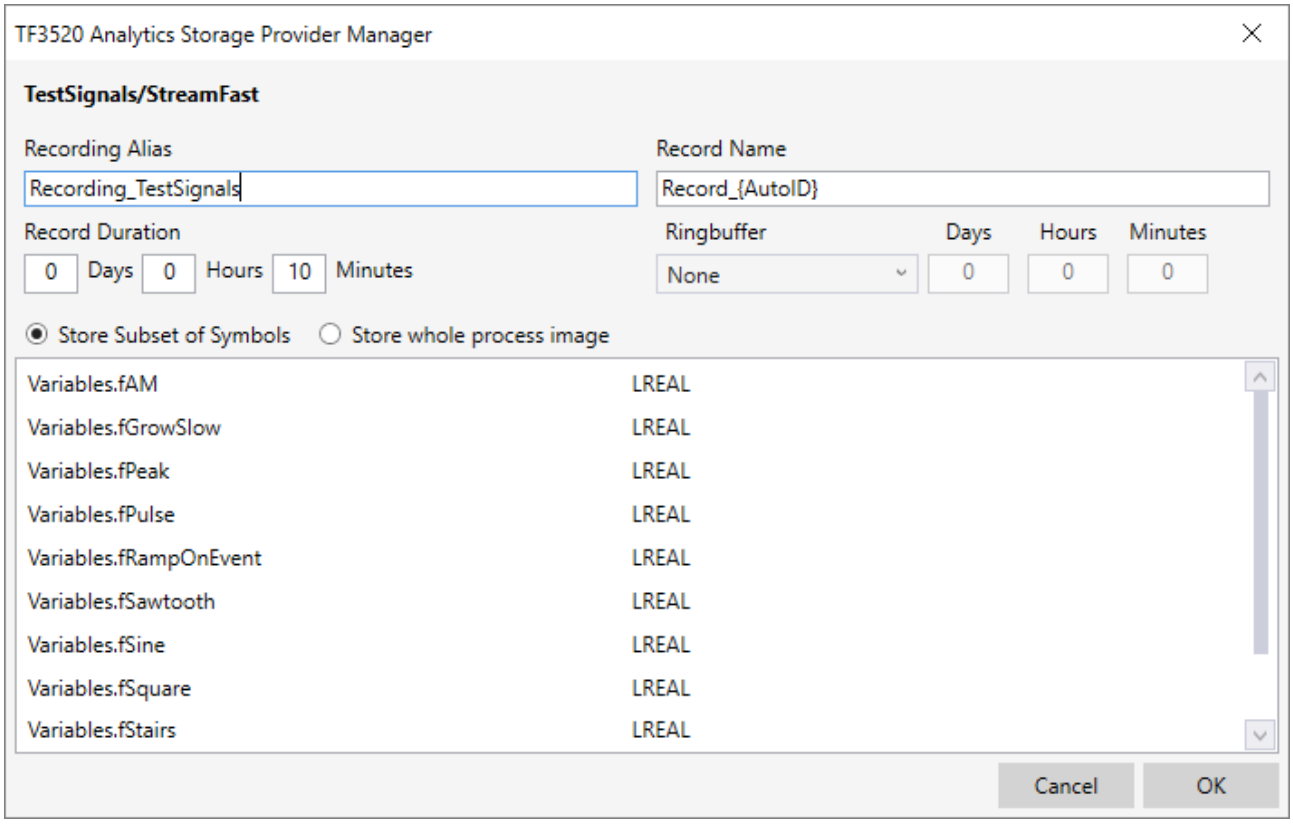
1	Create a new pipeline
2	Create a new pipeline with Rule Engine
3	Open Target Browser for connecting simple pipelines
4	Edit a selected pipeline
5	Delete a selected pipeline
6	Start a selected pipeline

Function Manager window ("CONFIGURATIONS")

The window is divided into two tabs. Pipelines and Live Status. Under Pipelines you will find the configurations of your pipelines. You can define new pipelines from here. Edit existing. Delete or start.



To create a new simple pipeline, click the "Create new pipeline" button. The following dialog opens.



You can now drag and drop the symbols you want to record from the Target Browser into the dialog. You also assign a Recording Alias and a Record Name.

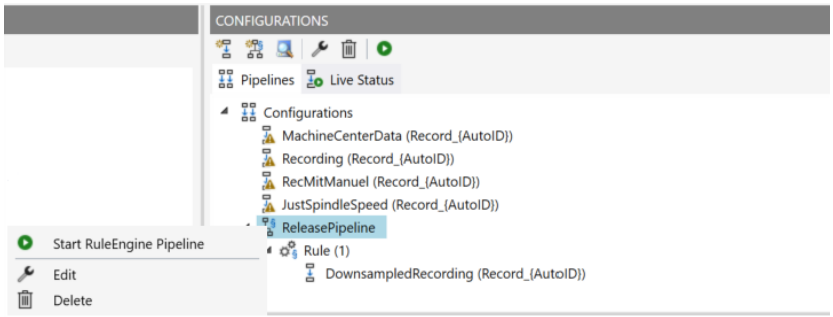
Various placeholders are available for the Record Name:

"{AutoID}"	
"{Topic}"	
"{SystemID}"	
"{Layout}"	
"{CycleTime}"	
"{SampleSize}"	
"{RecordStart}"	

You can also configure recording names and a duration (otherwise the recording will run endlessly until it is stopped manually). A ring buffer can be set according to storage space or time.

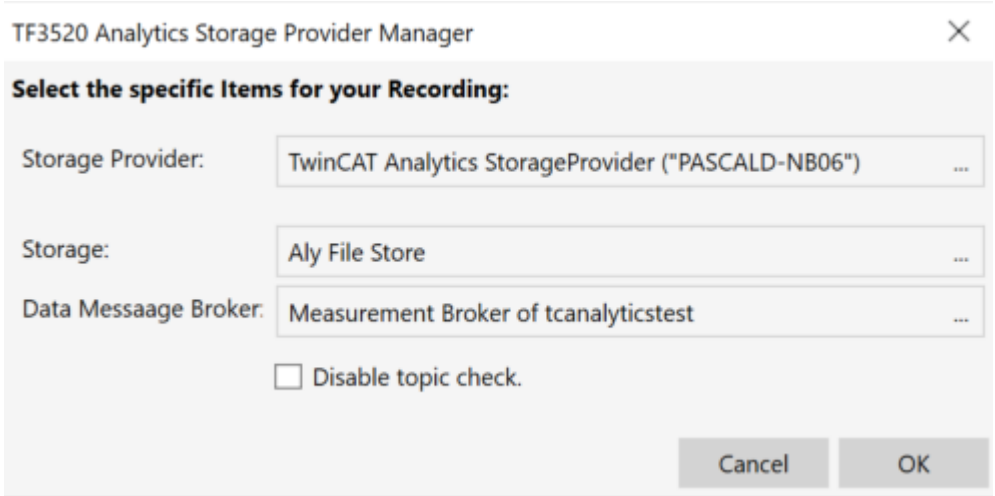
The entries are confirmed with **OK** and a new local recording definition is created.

It is now possible to start this definition directly via the toolbar or the context menu.



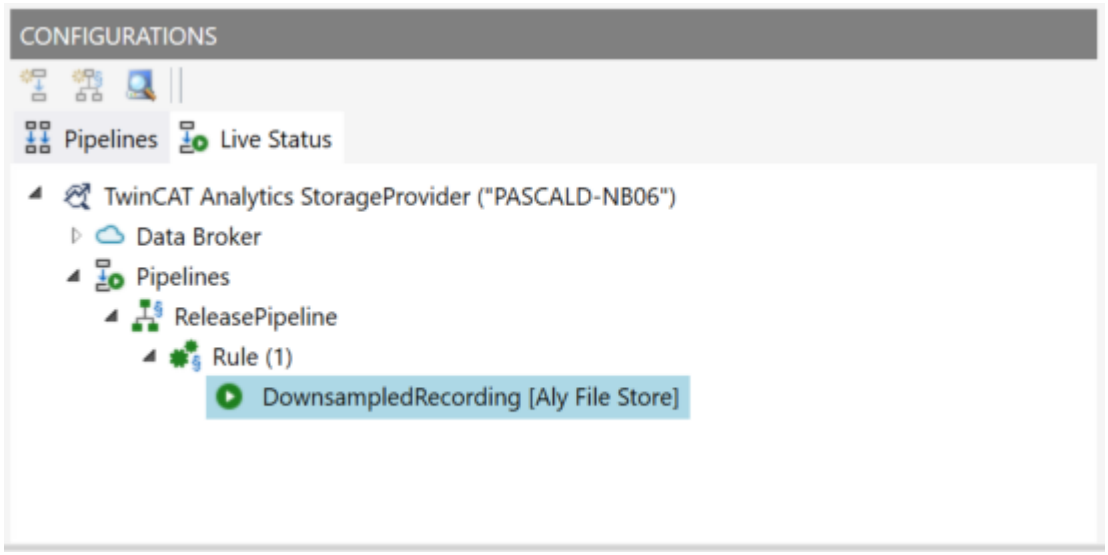
However, it is also possible to make the definition globally accessible. This can be done via the context menu with the entry **Publish Recording**.

The following dialog then opens:



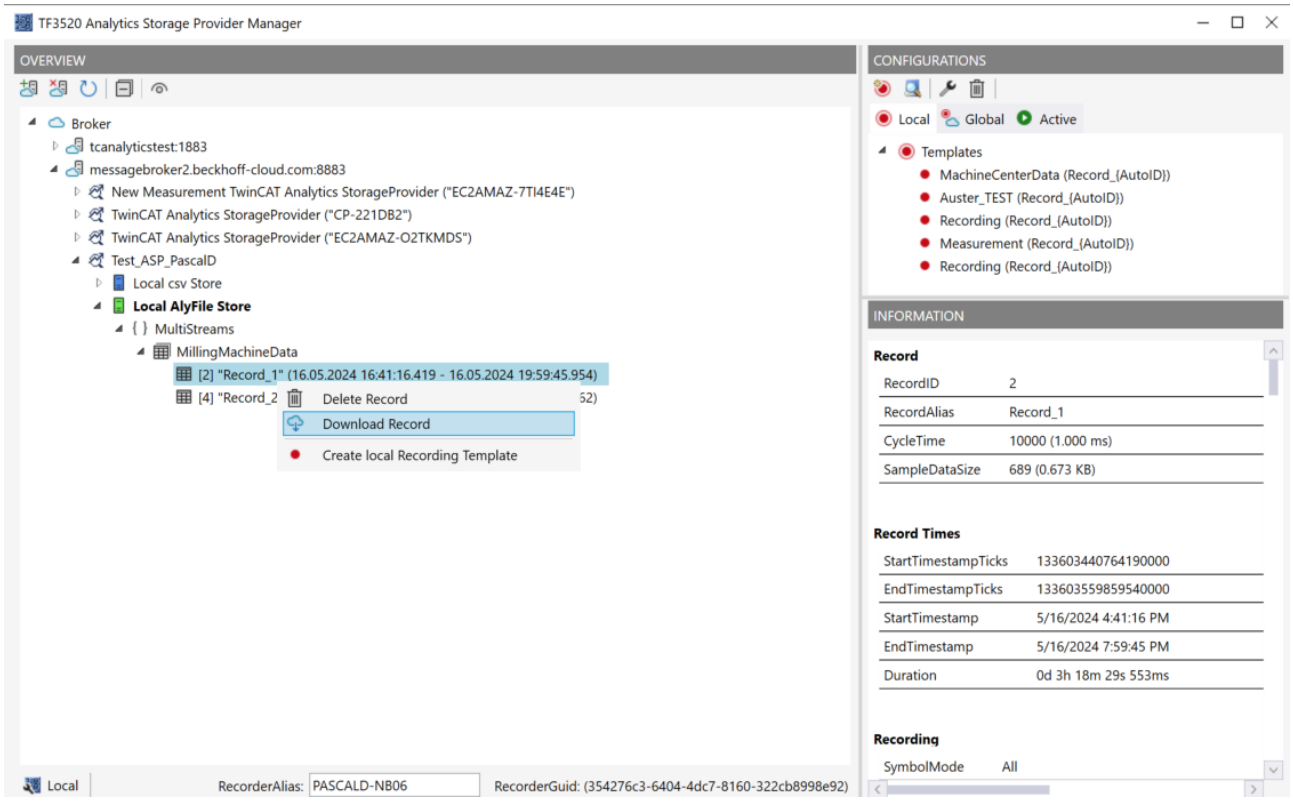
Here you can now select the desired Analytics Storage Provider via which the definition is to be published. In addition, the definition is assigned a Storage and a Data Broker of the selected Analytics Storage Provider. After the selection, the recording definition is confirmed with **OK** and published to the selected Analytics Storage Provider. This means that it can be found by any Storage Provider Manager that is connected to the MQTT Broker.

After starting a pipeline, the view automatically jumps to the second tab, the Live Status.



All active recordings from all users are listed here. The recordings can be ended in this tab and it is also possible to jump to the resulting record.

Furthermore, a download function is available in the Manager to download data to your local engineering. Select the corresponding record and start the download via the context menu. Regardless of the store in which the data is stored, it is always saved as an Analytics File on the local engineering system.



In combination with the data import function [▶ 55] in the Storage Configurator, the Analytics File also serves as a practical exchange format. Both functions can be used to move data from a CSV file to an InfluxDB2, for example, or from a Microsoft SQL® to a CSV file, etc.

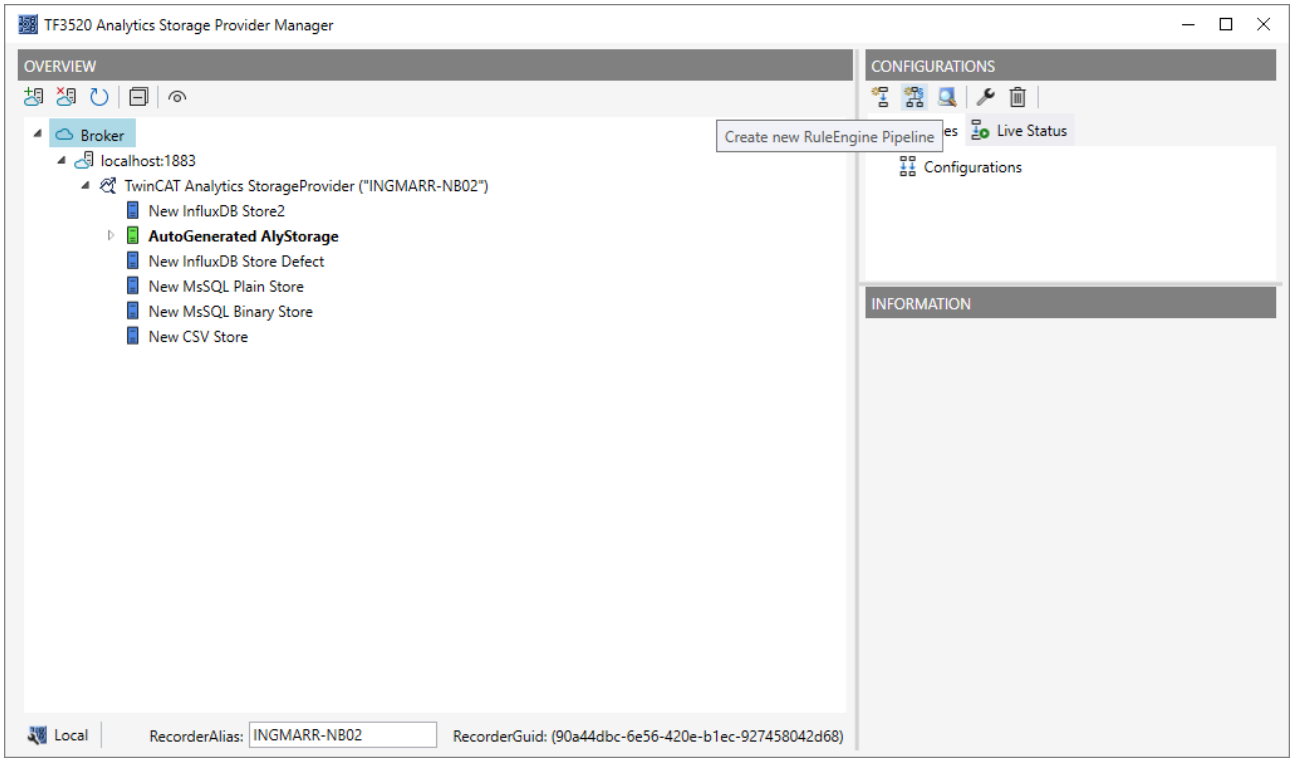
6.2.2 Data handling with Rule Engine

Rule Engine pipelines can be used to automatically map a wide variety of storage scenarios. Data can be collected from multiple MQTT data sources, processed and stored in Recordings. In addition, data from recordings can be read and processed on an event or timer-controlled basis, as well as manually triggered. Possible processing operations include aggregating, filtering and sampling the data sources.

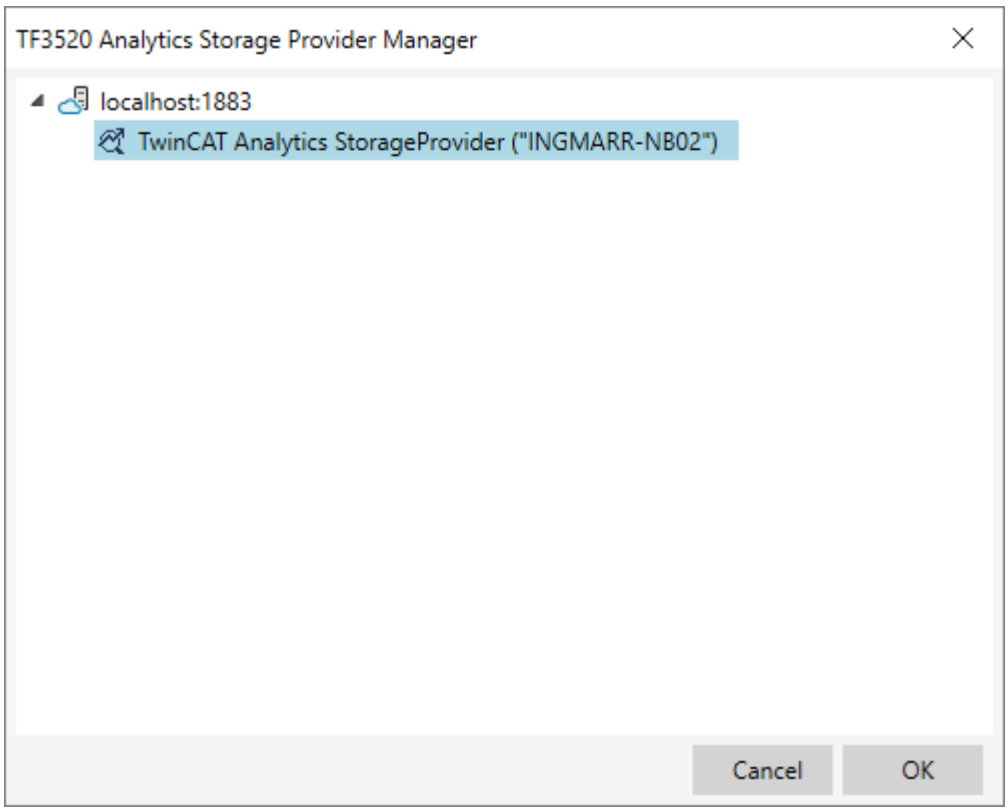
The RuleEngine pipeline function is available from Storage Provider version 3.15. onwards.

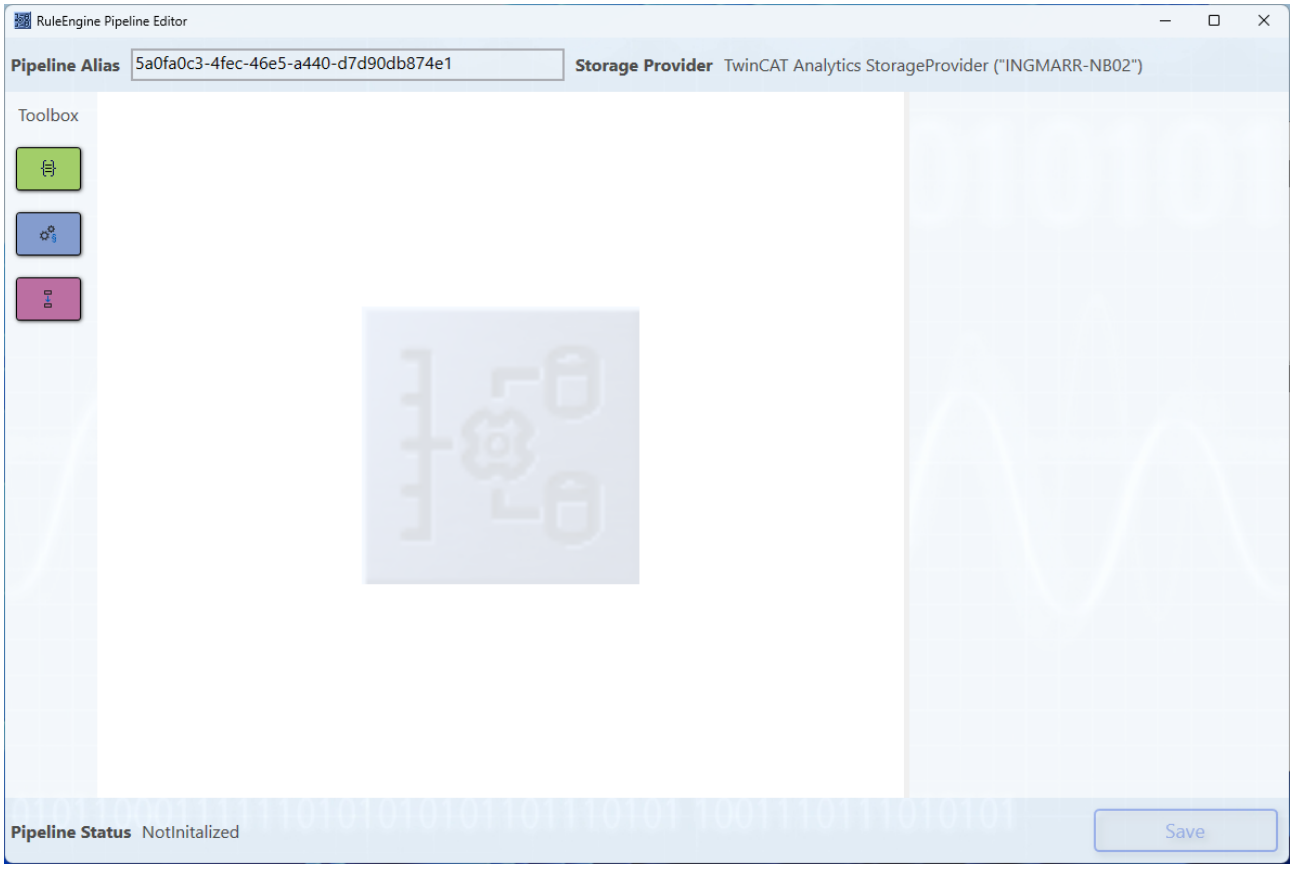
6.2.2.1 Configuration

A RuleEngine pipeline is configured graphically in the RuleEngine Pipeline Editor. This can be opened via the Manager.

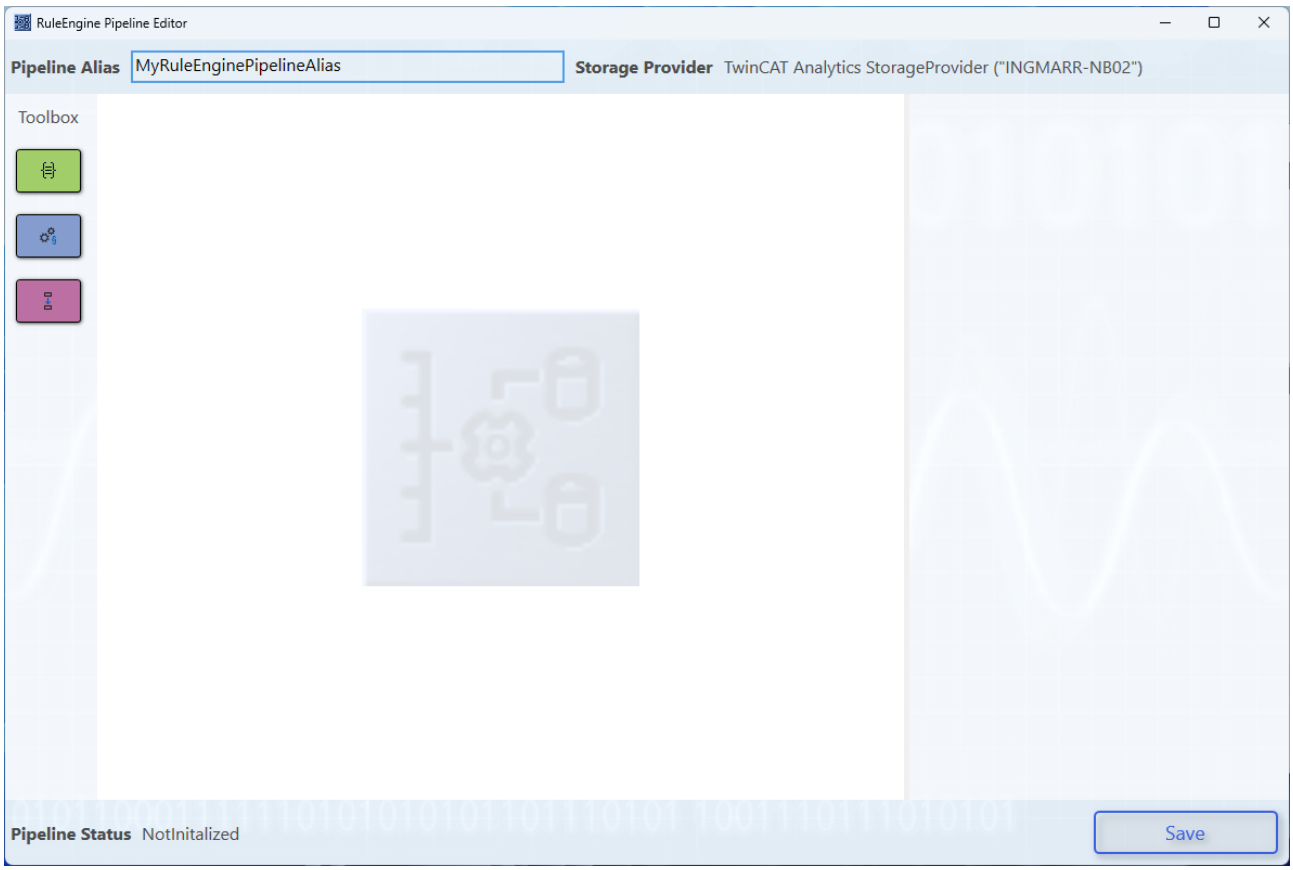


First, the desired Storage Provider must be selected. A RuleEngine pipeline is developed specifically for a Storage Provider. This cannot be changed subsequently. In the RuleEngine Pipeline Editor, the name of the selected Storage Provider is displayed at the top right.





An alias can be assigned for a RuleEngine pipeline. The alias is freely selectable and can be edited via the text field at the top left of the RuleEngine Pipeline Editor.



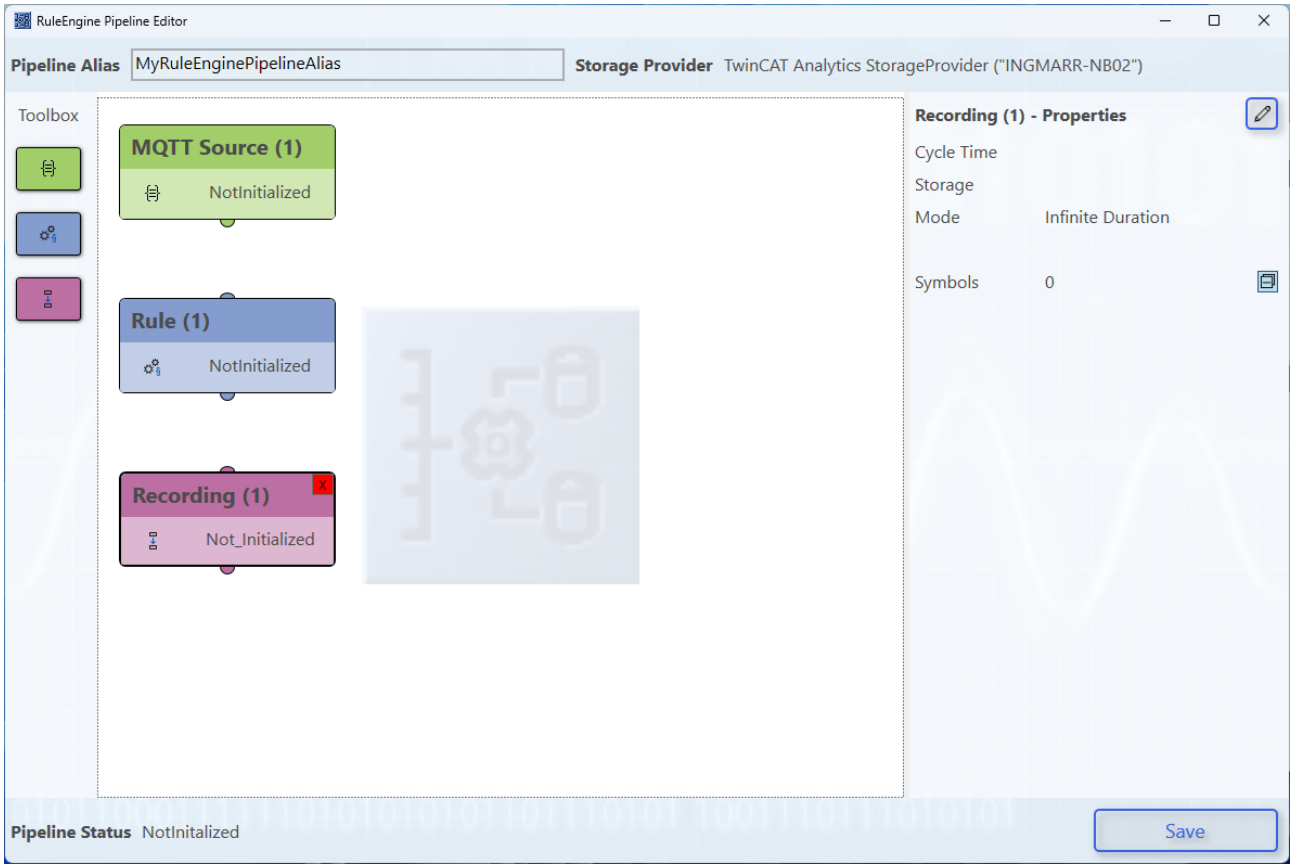
6.2.2.1.1 Adding elements

Various elements can be dragged from the toolbox (left) in the RuleEngine Pipeline Editor into the configuration area (center). The following elements can be used:

MQTT-Source: Corresponds to a data source (e.g. TwinCAT Analytics Logger or TwinCAT IoT Data Agent).

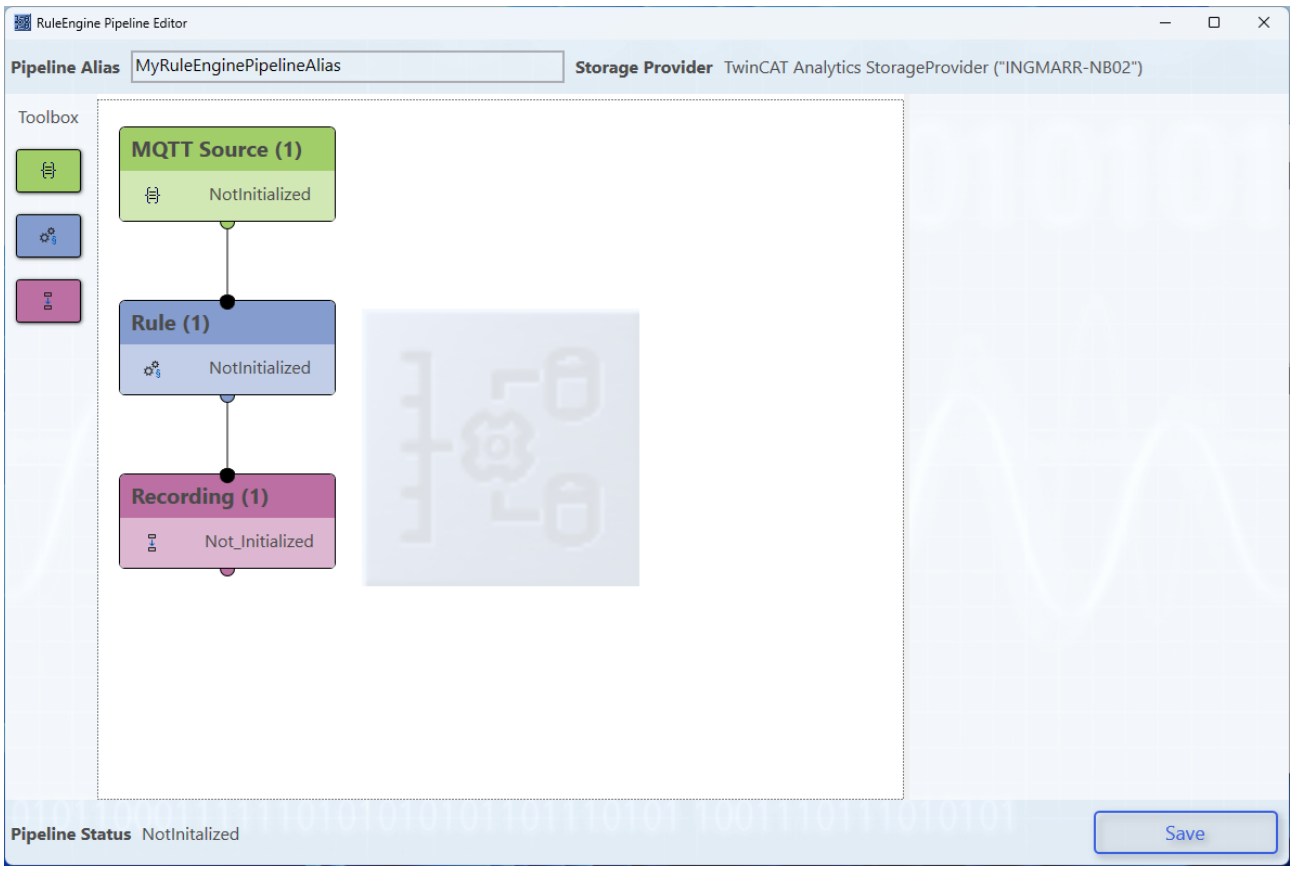
Rule: Pre-processing steps can be defined with a rule.

Recording: Describes a data recording (previously known recording).



6.2.2.1.2 Linking elements

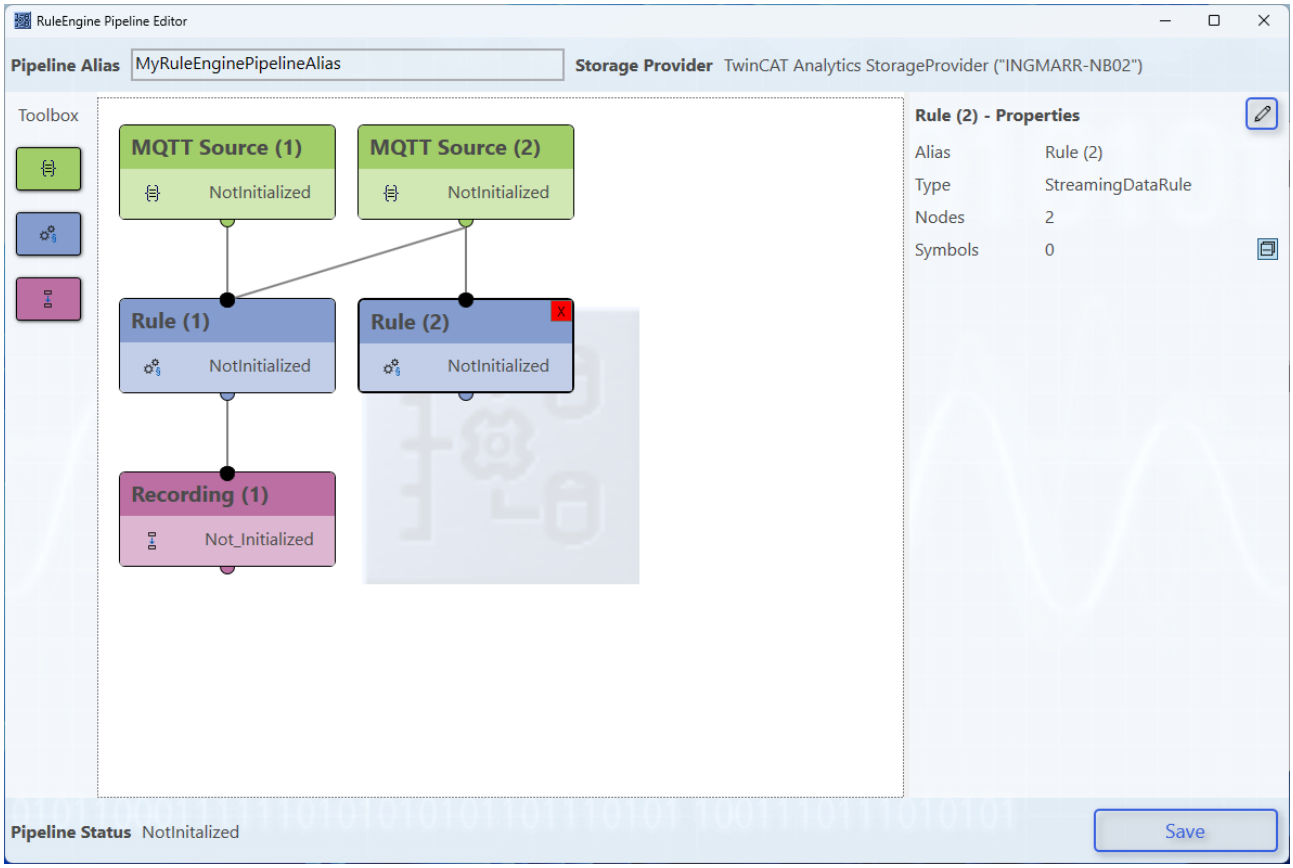
The added elements can be linked to each other via connections. A connection can be created via the output of an element. The mouse can be used to drag the connection to another element and thus create a link in the data flow. If you release the connector in the free area, a subsequent element is automatically created or a selection of available element types is offered.



The following connections are possible:

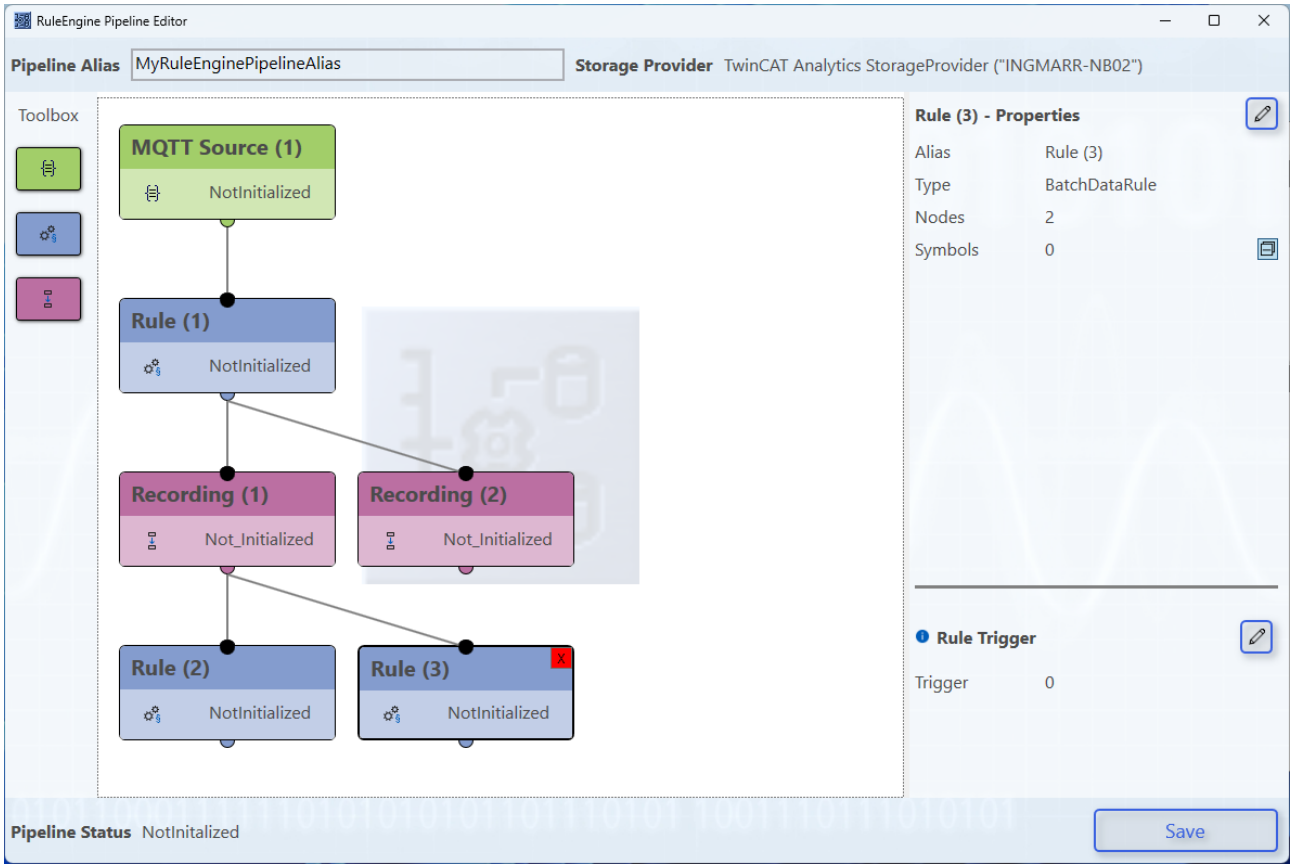
MQTT-Source - Rule:

An MQTT-Source can be used in several rules. In addition, several MQTT-Sources can be used in a rule, provided they run with similar system times. If this is not the case, the editor recognizes this and issues a corresponding warning.



Rule - Recording:

Any number of recordings for saving data can be defined for a rule. However, only data from one rule can flow into a recording. For a recording, any number of rules can be created for further processing of the saved data. Here too, a rule can only accept data from a recording. No subsequent rules can currently be created for recordings with a ring buffer. It is also not possible to combine data from an MQTT-Source and a recording as data sources for a rule.



MQTT-Source - Recording:

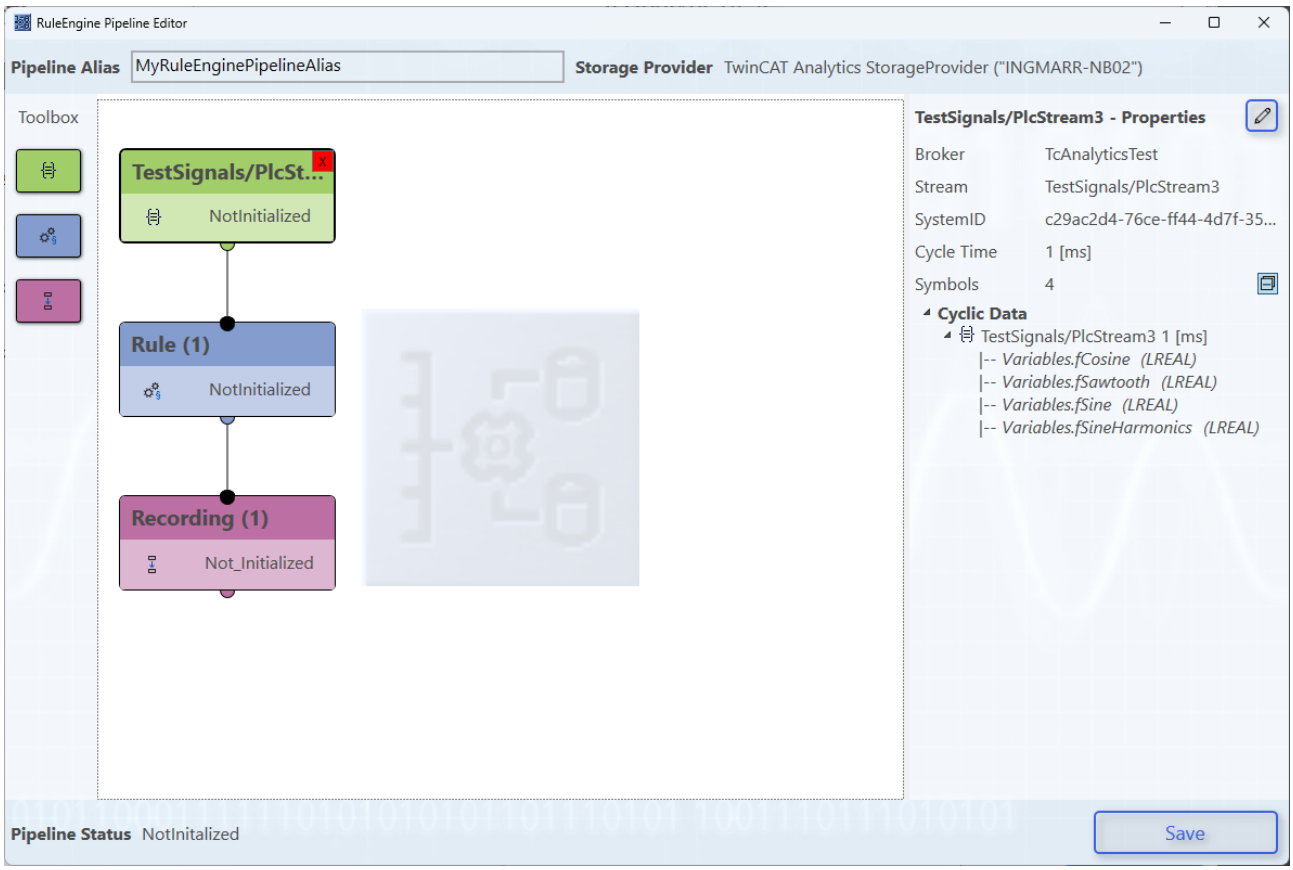
This connection is not possible. A rule must always be inserted as an intermediate connection.

6.2.2.1.3 Viewing properties of the elements

The properties of the selected element can be viewed via the Properties window on the right-hand side of the RuleEngine Pipeline Editor. The properties vary depending on the type.

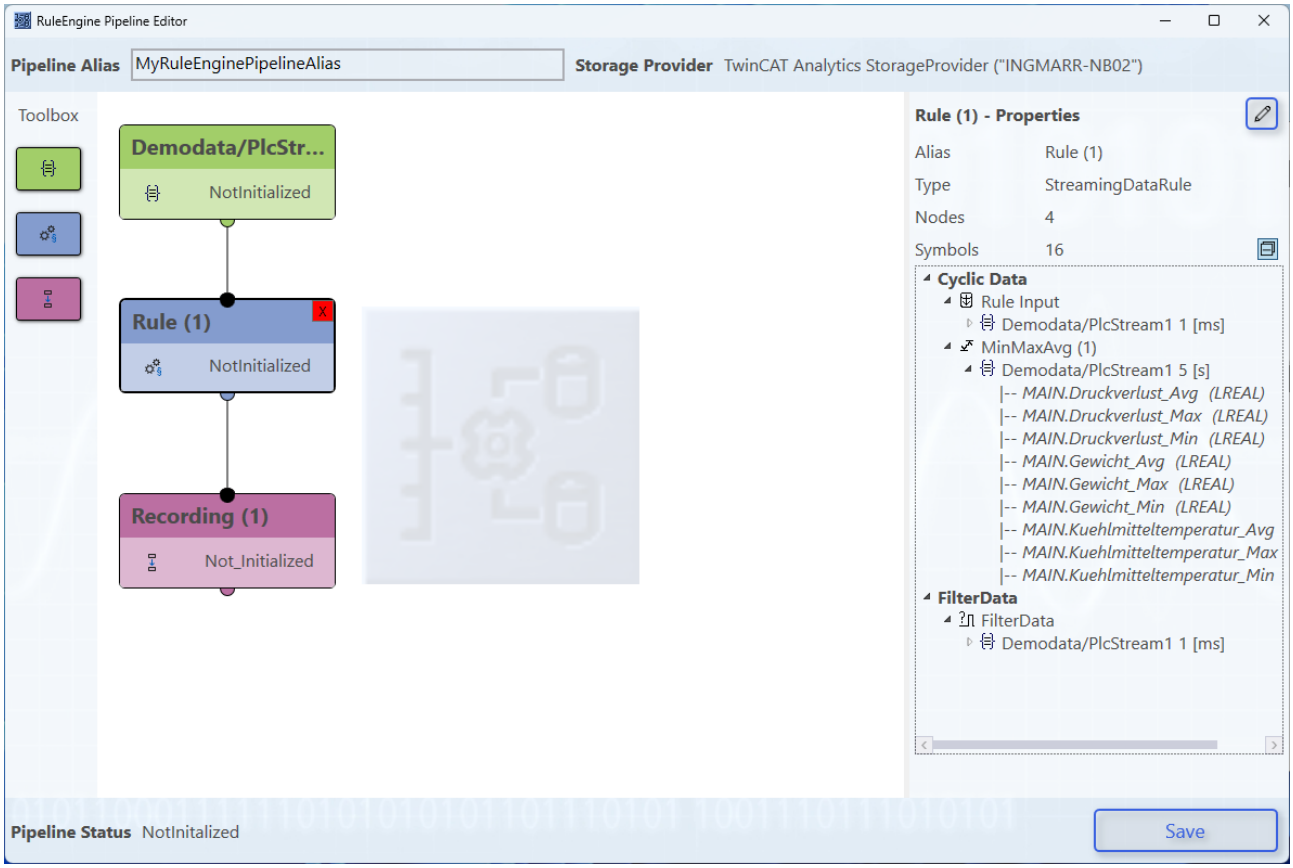
MQTT source

- **Broker:** Message broker of the data source.
- **Stream**: Topic of the data source
- **SystemID**: System ID of the data source
- **Cycle time**: Cycle time of the data source
- **Symbols**: Number and names of the symbols selected at the data source.



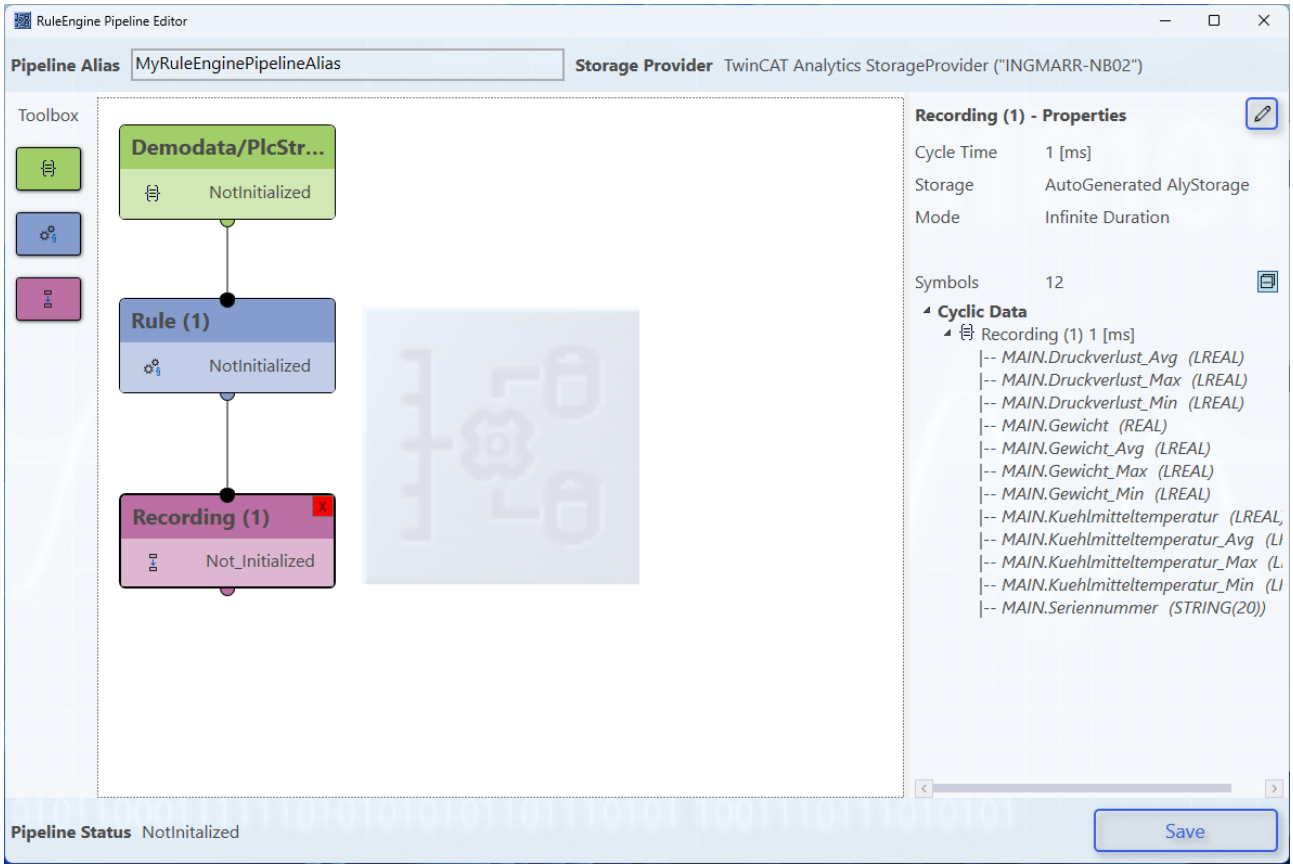
Rule

- **Alias**: Alias name of the rule
- **Type**: Type of rule. There are currently StreamingDataRules and BatchDataRules. StreamingDataRules refer to streamed data and run continuously to process the streamed data. BatchDataRules refer to data that has already been saved. These do not run continuously, but are started via triggers. They process a certain amount of stored data and then shut down again.
- **Nodes**: Number of nodes in a rule.
- **Symbols**: Number and names of the symbols that are output from the rule.
- **RuleTrigger** (only for BatchDataRules): Number of configured triggers.



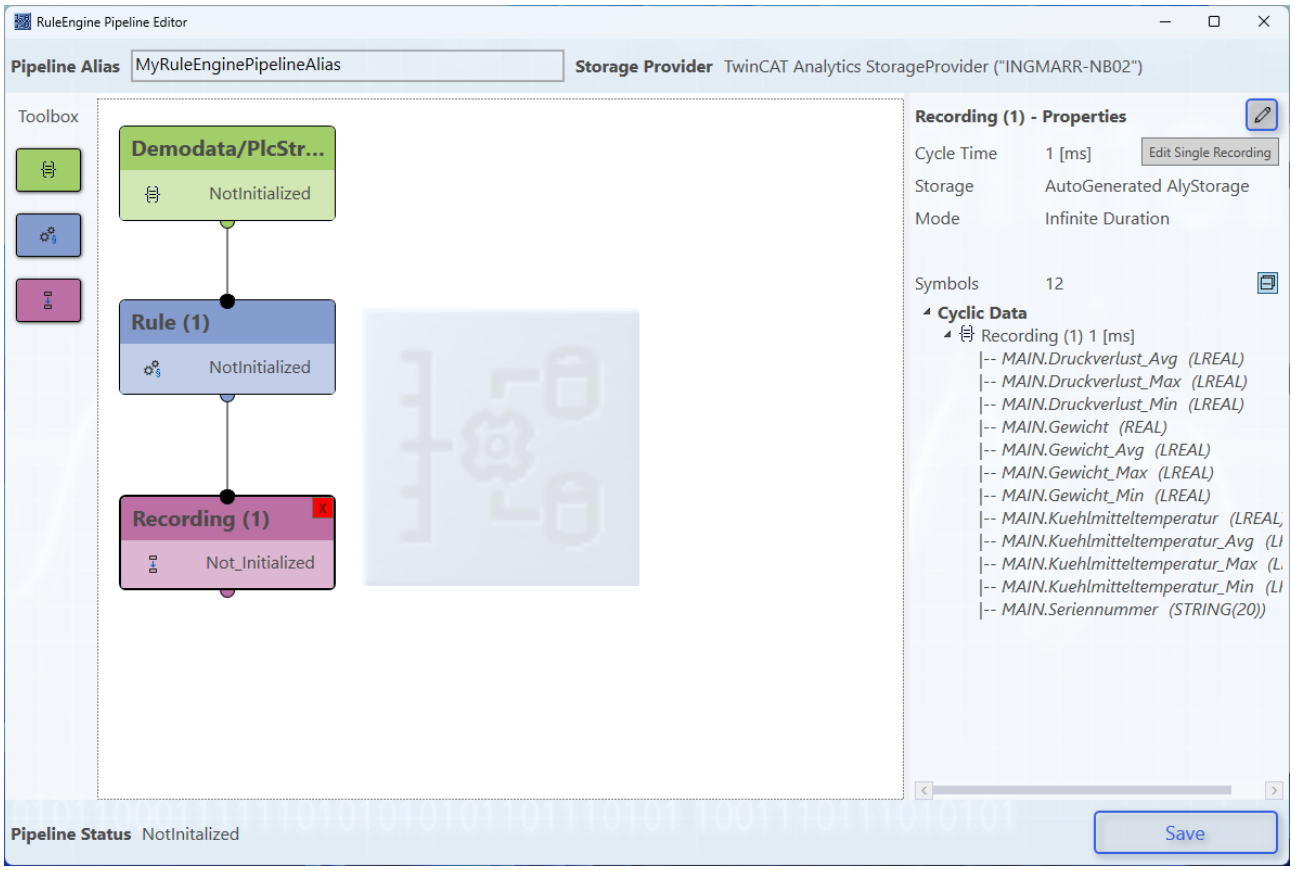
Recording

- **CycleTime**: Cycle time of the data recording
- **Storage**: Name of the data memory
- **Mode**: Recording mode (Infinite duration or ring buffer)
- **Symbols**: Number and names of the selected symbols



6.2.2.1.4 Editing elements

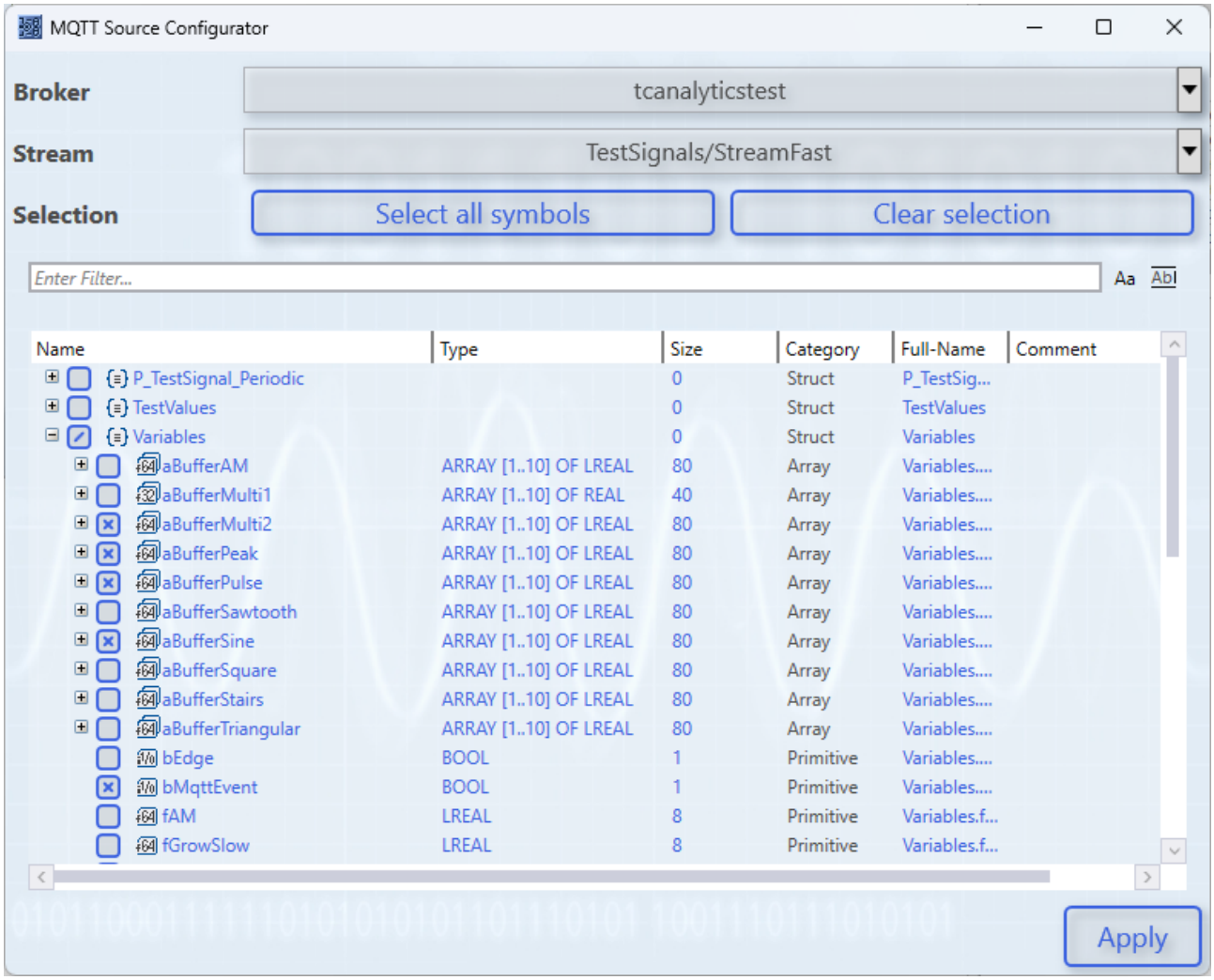
Editing takes place in a separate window. This can be opened by double-clicking on the element or by clicking the Edit icon in the properties.



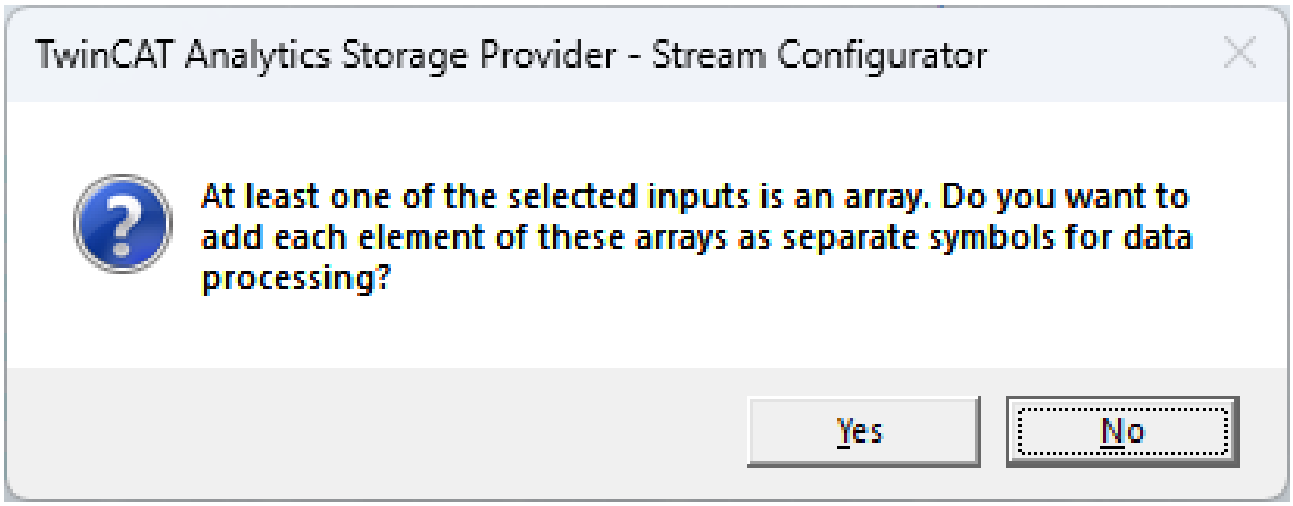
6.2.2.1.4.1 Editing MQTT-Sources

An MQTT-Source is edited in the MQTT Source Configurator. First, a message broker must be selected at Broker. All Data Brokers of the Storage Provider used for the RuleEngine pipeline that have also been added in the Manager can be selected. An MQTT-Stream (Stream) can then be selected based on its topic.

As soon as a stream has been added, the available symbols for the stream are displayed. Individual symbols can be selected using the checkboxes. All displayed symbols can be selected or deselected using the **Select all symbols** and **Clear selection** buttons.



If the selected symbols include arrays or structures, a user dialog will prompt you to confirm whether the subsymbols should be added as separate symbols when the configuration is saved. If these are added, the subsymbols can be used individually in rules or recordings for processing and saving.



6.2.2.1.4.2 Editing rules

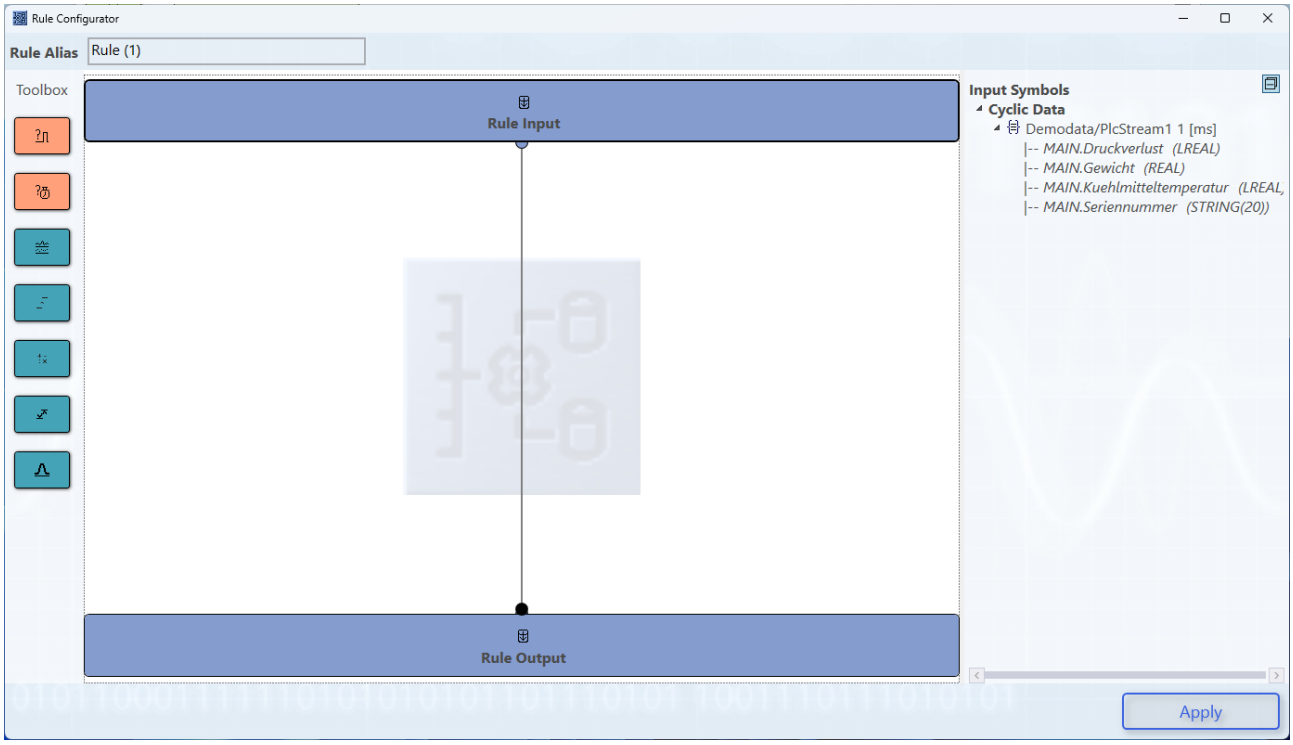
A rule is edited in the Rule Configurator. An alias name can be assigned to a rule at the top left. A rule consists of the following elements:

Rule Input:

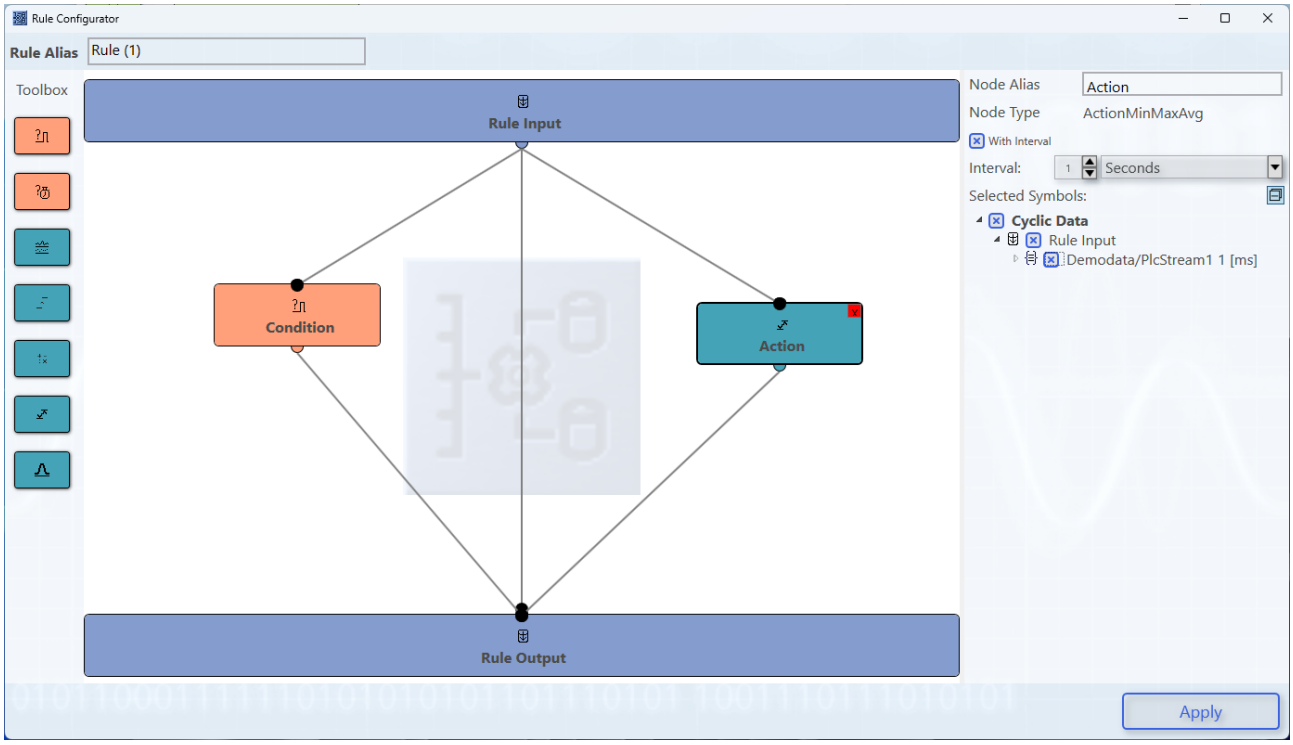
Corresponds to the start point within a rule. All symbols added to the rule are displayed here. For BatchDataRules, the time range for which the data is loaded when the rule is triggered can be selected here.

Rule Output:

Corresponds to the output of the rule. All symbols of the blocks that are connected to the rule output can be selected for subsequent recordings.

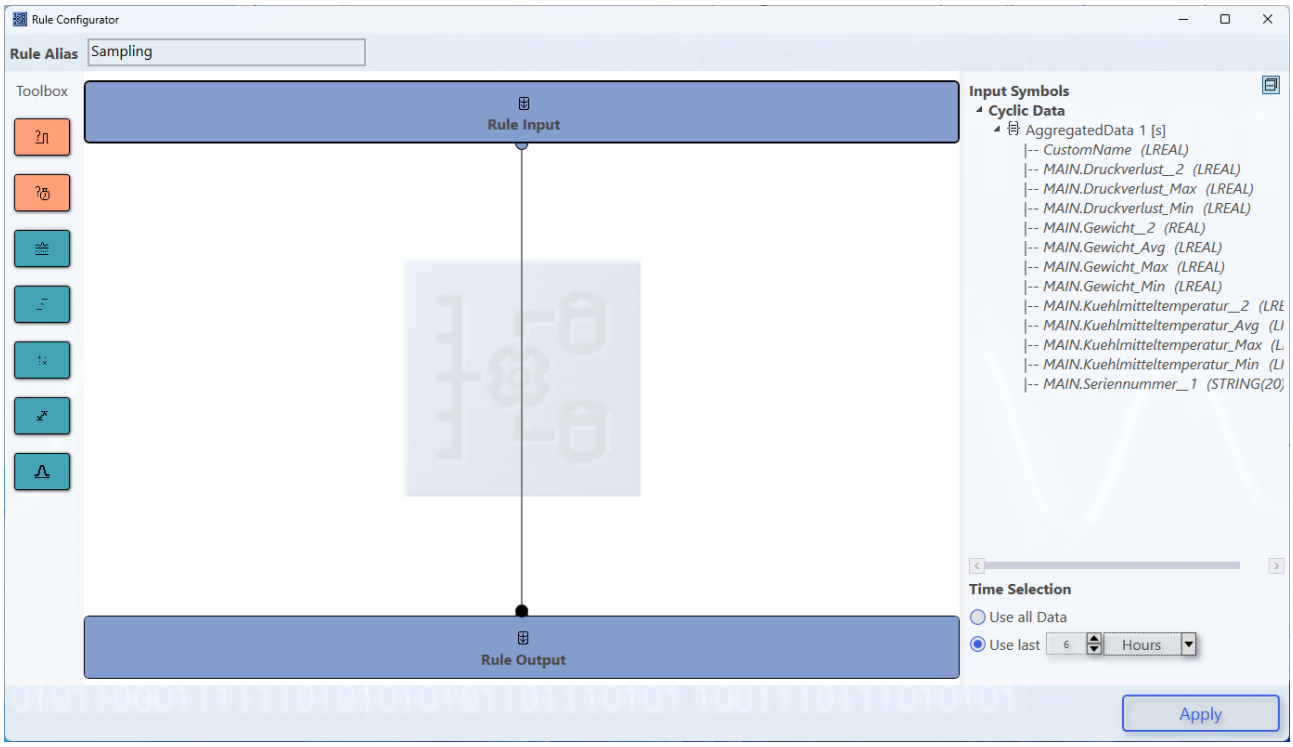


Additional blocks can be added for processing via the toolbox (left). A distinction is made between Condition blocks and Action blocks. Condition blocks serve as filters and enable incoming data to be filtered on a time basis or data basis. Action blocks offer the option of aggregating data (e.g. calculating min/max/avg or other operations). All blocks can be linked to each other with connections.



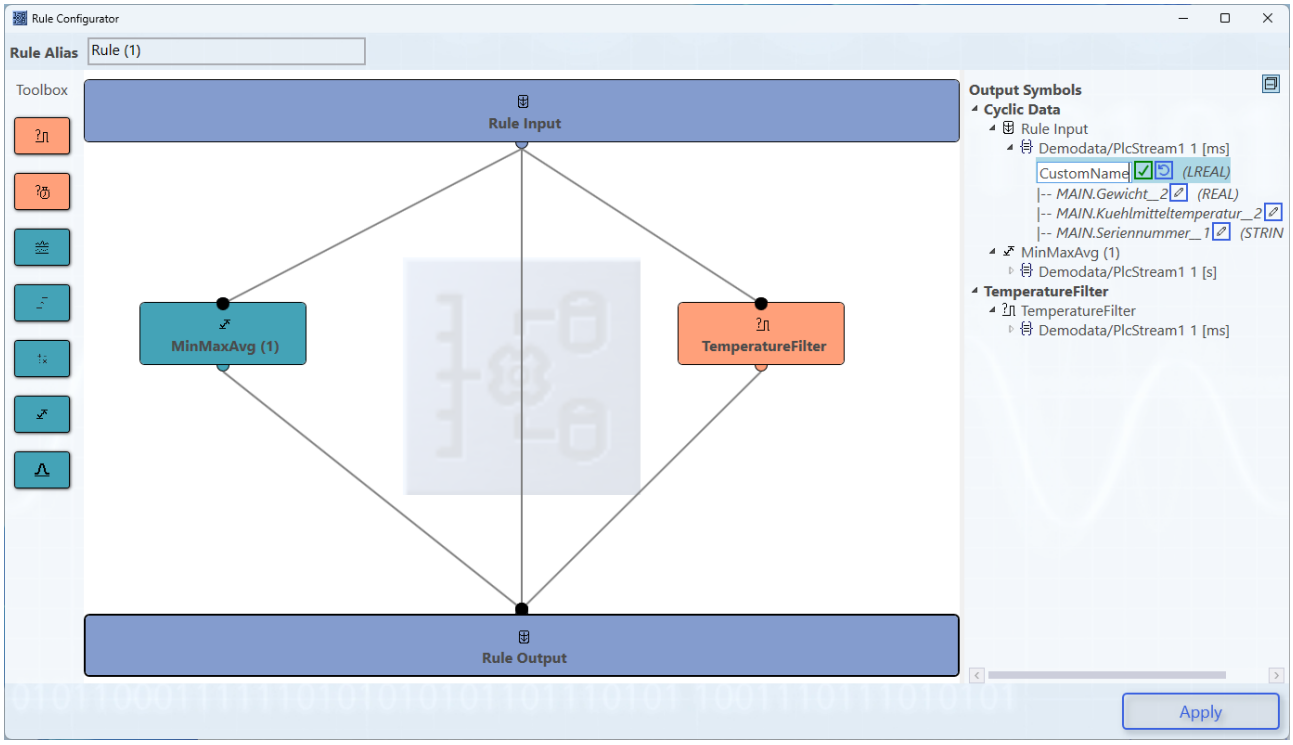
6.2.2.1.4.2.1 Input

Corresponds to the start point within a rule. All symbols added to the rule are displayed here. For BatchDataRules, the time range for which the data is loaded when the rule is triggered can be selected here.



6.2.2.1.4.2.2 Output

Corresponds to the output of the rule. All symbols of the blocks that are connected to the rule output can be selected for subsequent recordings. Variables can be renamed at the output node. To do this, you can click on the Edit icon behind the corresponding variable.



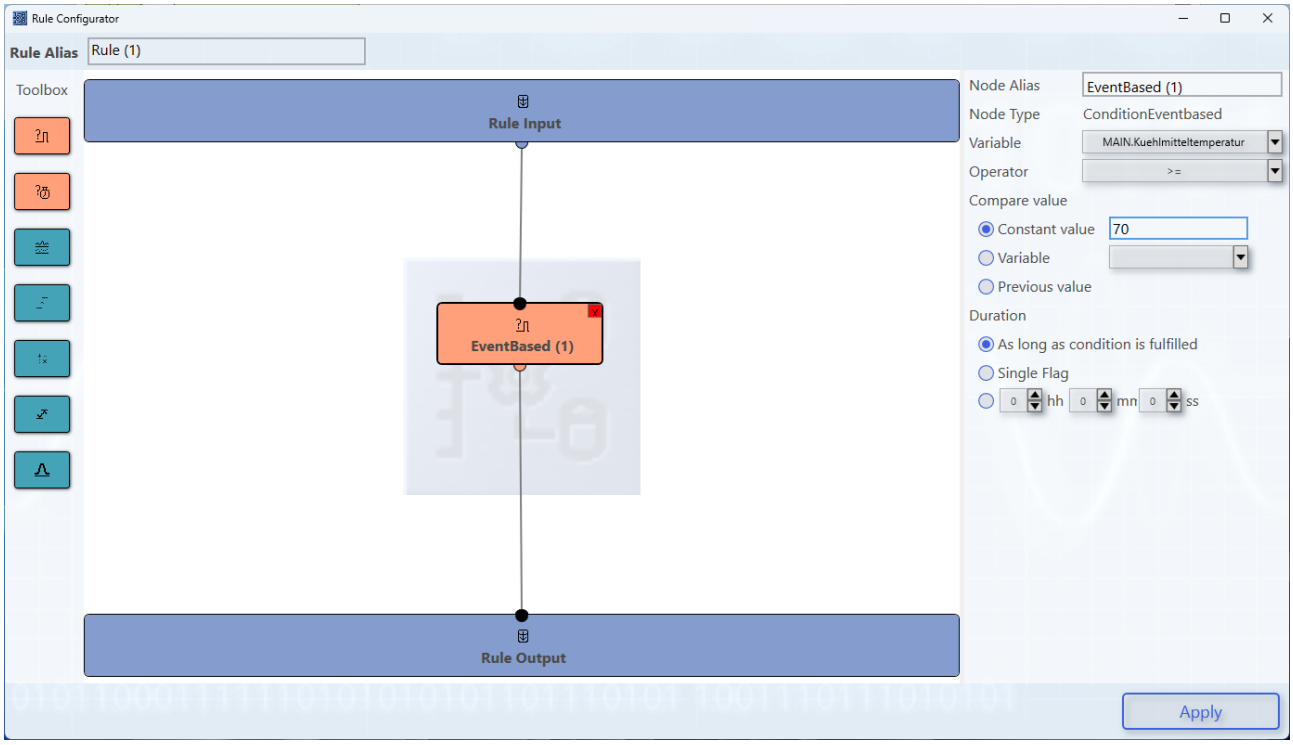
6.2.2.1.4.2.3 Conditions

Condition blocks serve as filters and enable incoming data to be filtered on a time basis or data basis.

6.2.2.1.4.2.3.1 Eventbased

The EventBased-Condition can be used to filter for data within the data where a specific event must have occurred. The following configurations can be made:

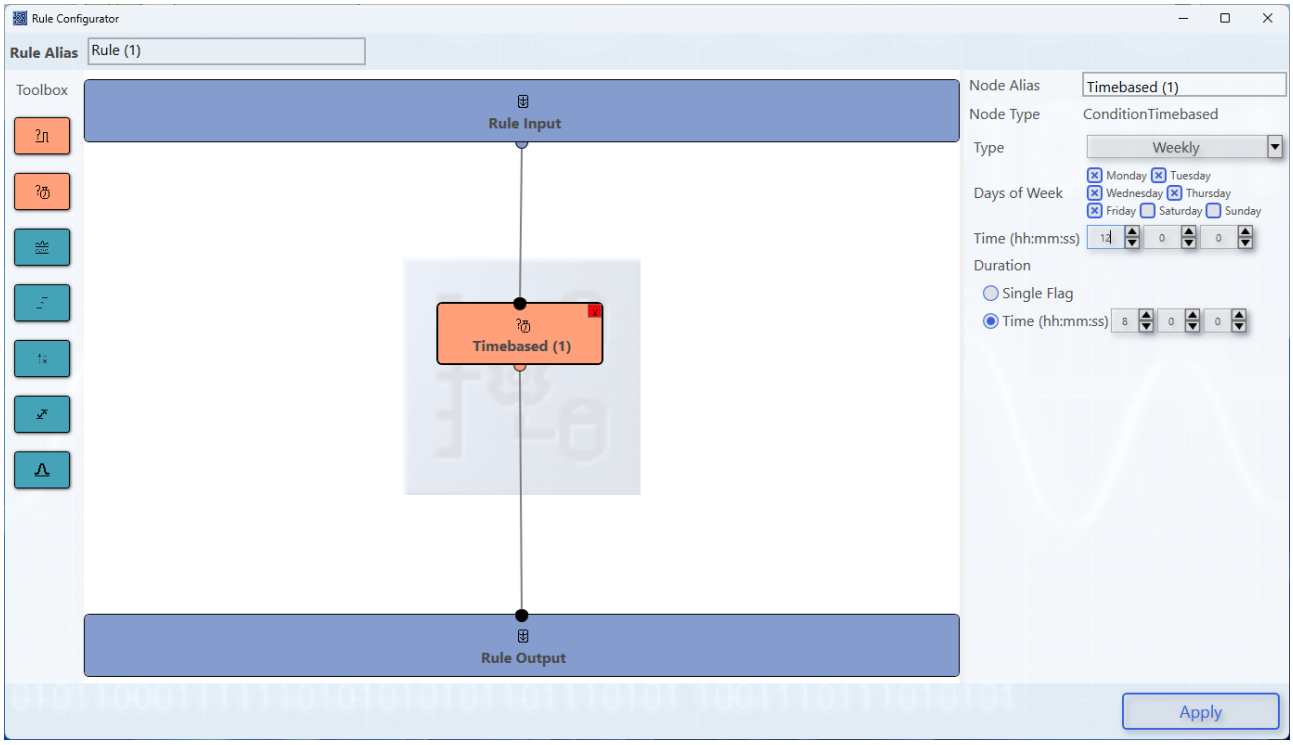
- **Variable**: Selection of the variables to which the condition must apply.
- **Operator**: Selection of the operator for the condition
- **Compare value**: Selection of the comparison value
  - **Constant value**: Comparison with a constant value
  - **Variable**: Comparison with another variable
  - **Previous value**: Comparison with the previous value of the selected variable.
- **Duration**: Selection of how long the data should be forwarded once the condition is fulfilled.
  - **As long as condition is fulfilled**: The data is forwarded as long as the condition is fulfilled.
  - **Single Flag**: The data is forwarded once when the condition is fulfilled, i.e. as with a rising edge.
  - Alternatively, the data can also be forwarded for a specific time, which can be configured manually.



6.2.2.1.4.2.3.2 Timebased

The Timebased-Condition can be used to filter within the data on a time basis. The time on which this is based is the respective timestamp of the data. The following configurations can be made:

- **Type**: Selection of the type of Timebased Condition. Three types are available
  - **Interval**: The condition is fulfilled as soon as the interval is reached.
  - **Weekly**: The condition is fulfilled on certain days of the week at a certain time.
  - **Monthly**: The condition is fulfilled in certain months on a certain day at a certain time.
- **Duration**: Selection of how long the data should be forwarded once the condition is fulfilled.
  - **Single Flag**: The data is forwarded once when the condition is fulfilled, i.e. as with a rising edge.
  - **Time**: Alternatively, the data can also be forwarded for a specific time, which can be configured manually.



6.2.2.1.4.2.4 *Actions*

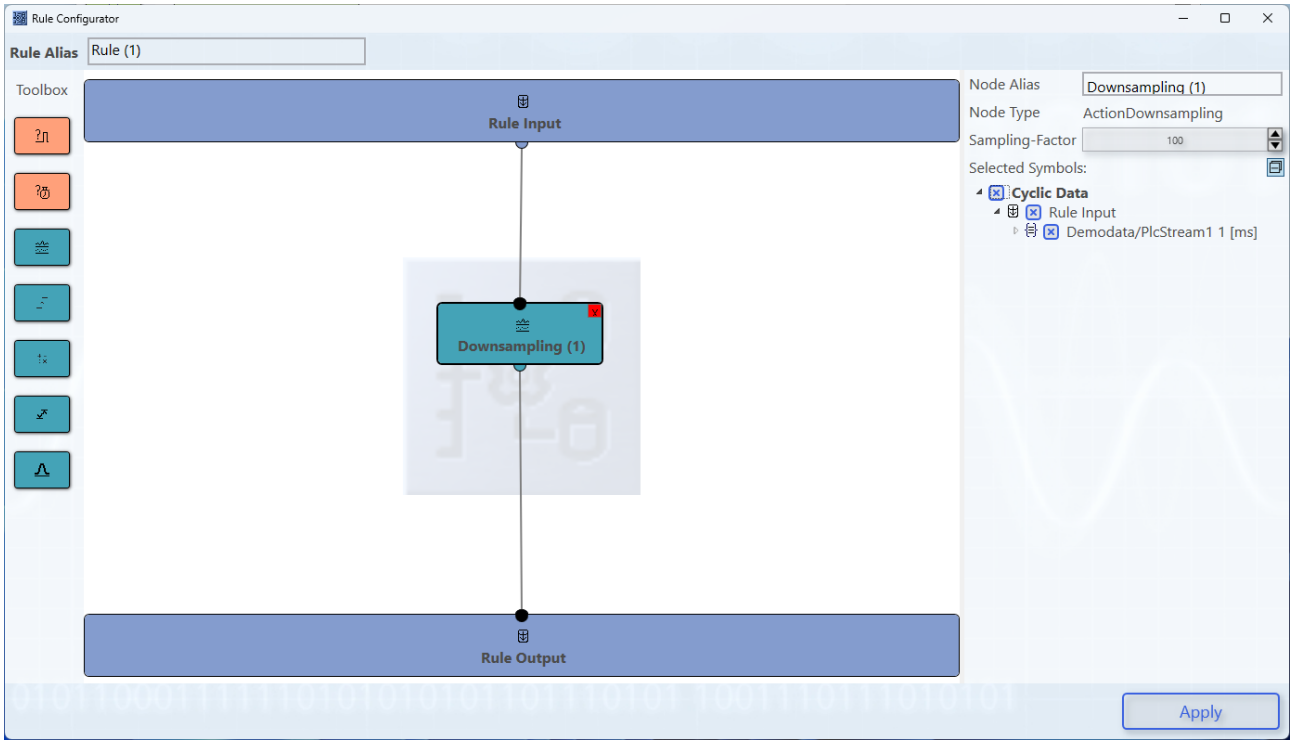
Action blocks offer the option of aggregating data (e.g. calculating min/max/avg or other operations).

6.2.2.1.4.2.4.1 Downsampling

Configuration of a downsampling of individual variables. The sampling factor indicates the sampling rate. In the Selected Symbols area, you can select the variables to which downsampling is to be applied. If you want to sample all the data in a rule, you can configure this in the corresponding recording.

The following symbols are provided for each selected symbol:

- _Sampled: Sampled symbol

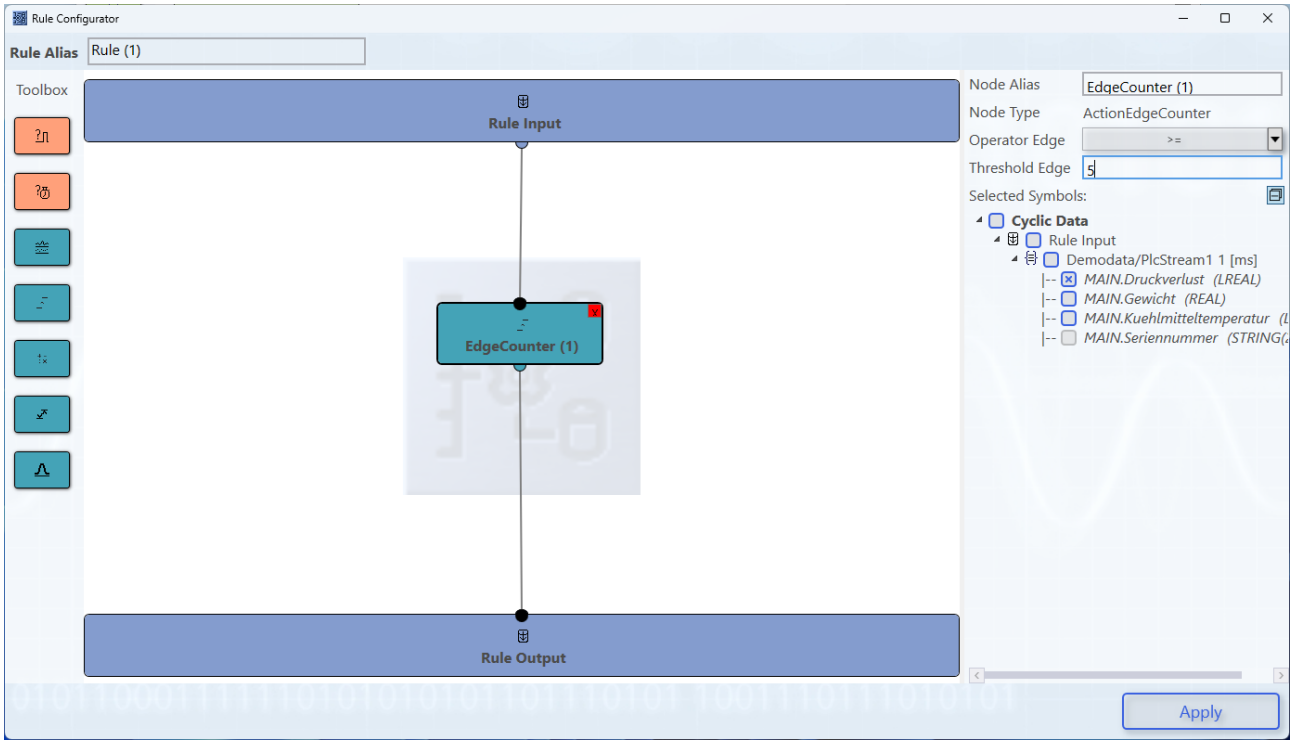


6.2.2.1.4.2.4.2 EdgeCounter

Determines the edges within the data. The Operator Edge and the threshold value to be compared must be configured for this. In the Selected Symbols area, you can select the variables to which the Edge Counter is to be applied.

The following symbols are provided for each selected symbol:

- _Count: Number of edges counted
- _Edge: Boolean whether an edge is detected
- _LastEvent: Timestamp of the last event

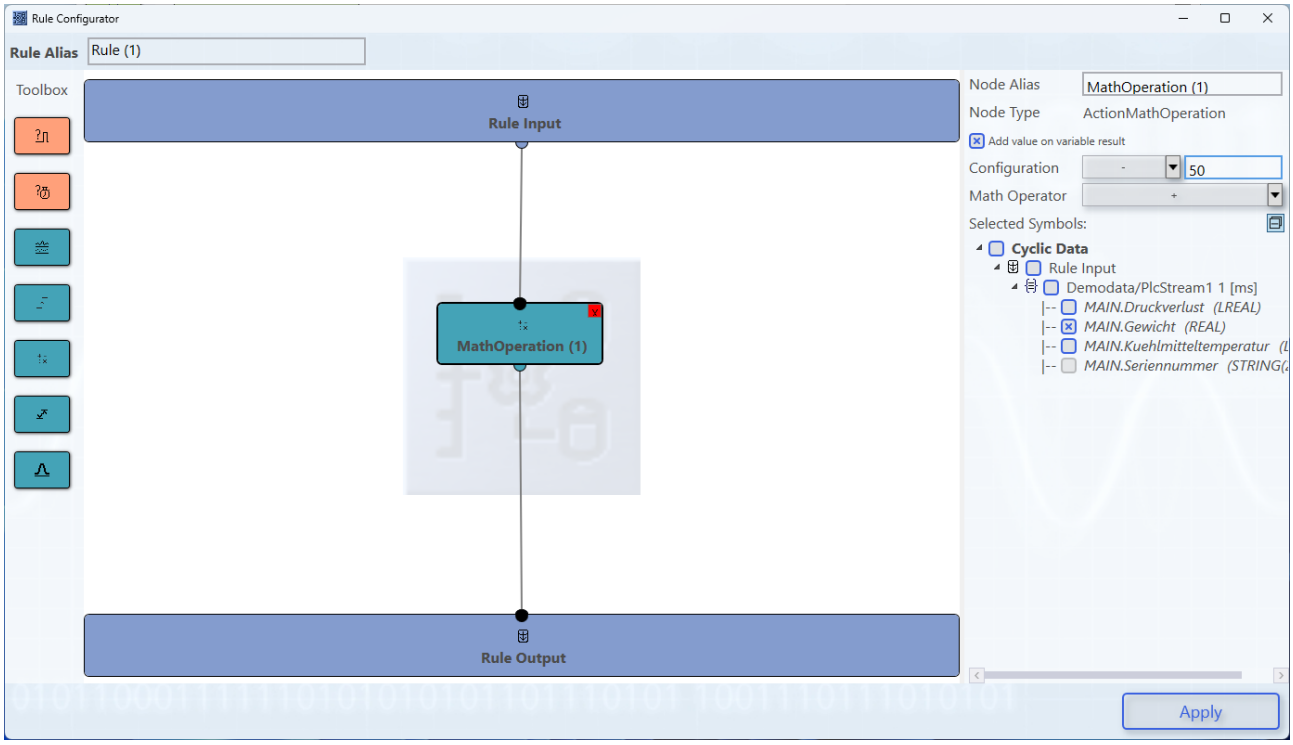


6.2.2.1.4.2.4.3 MathOperation

Symbols can be calculated with each other using the MathOperation block. The symbols to be calculated with each other can be selected under **Selected Symbols**. The operator can be selected under **MathOperator**. If the result is to be calculated with an additional constant value, activate the **Add value on variable result** checkbox. In the **Configuration** area, you can configure the operator and the constant value with which the result of the symbols is to be calculated.

The following symbol is provided:

- **Result**: Result of the calculation

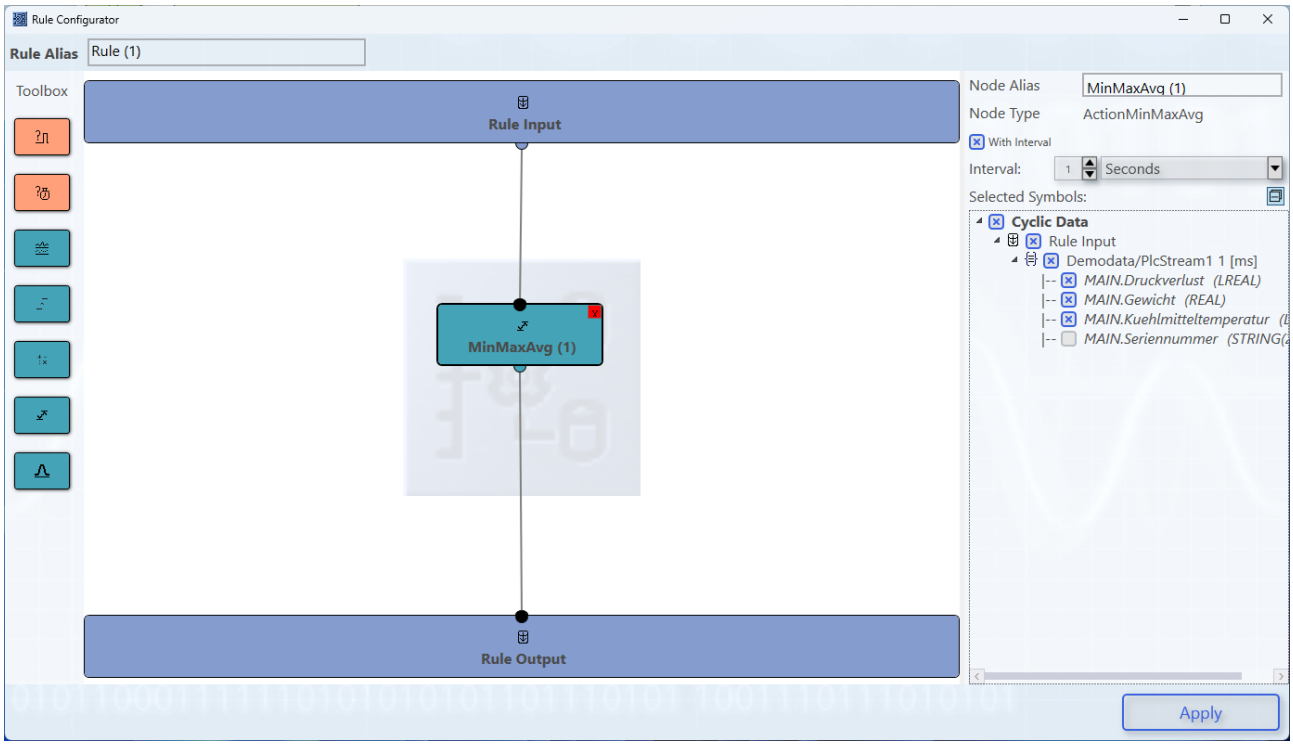


6.2.2.1.4.2.4.4 Min/Max/Avg

Calculation of the minimum/maximum and average value for selected symbols. In the **Selected Symbols** area, you can select the variables for which the values are to be calculated. If you activate the **With Interval** checkbox, you can configure an interval for which the values are to be calculated.

The following symbols are provided for each selected symbol:

- **_Min**: Minimum of the symbol
- **_Max**: Maximum of the symbol
- **_Avg**: Average of the symbol

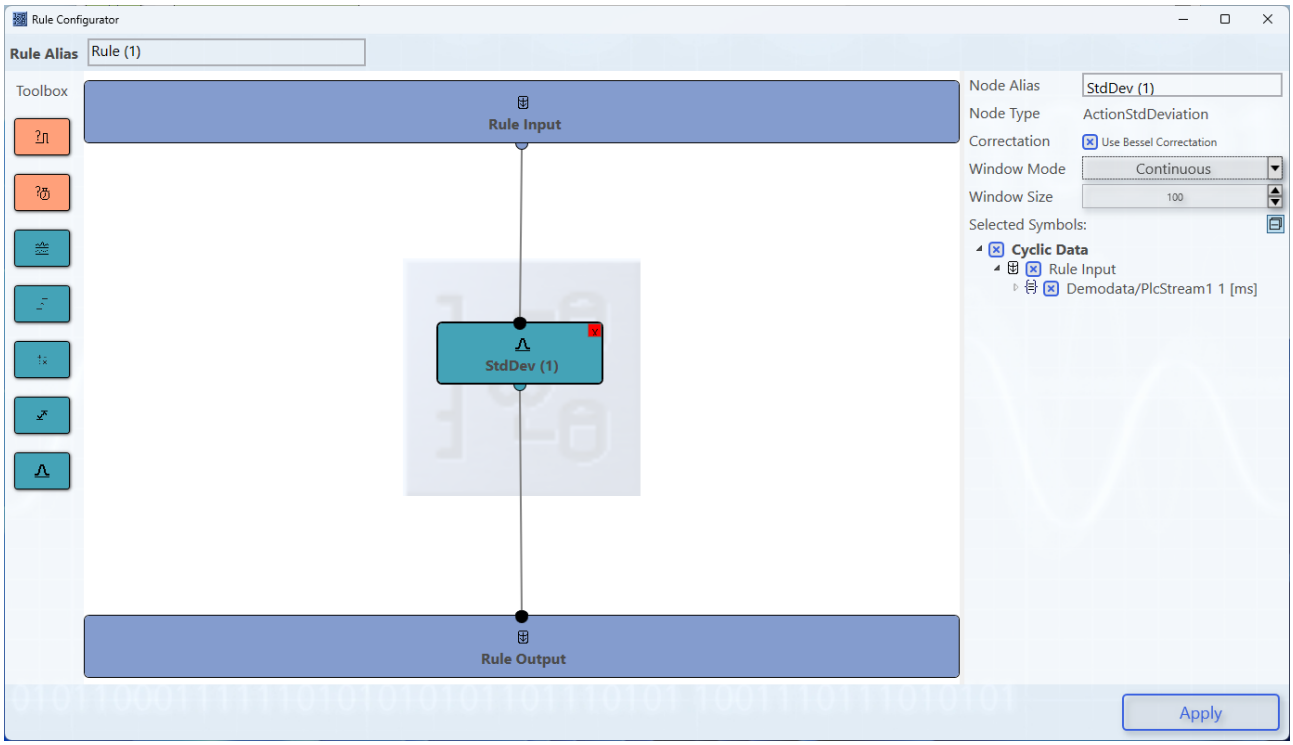


6.2.2.1.4.2.4.5 StandardDeviation

Calculation of the standard deviation for selected symbols. In the **Selected Symbols** area, you can select the variables for which the values are to be calculated. The calculation of the standard deviation can be configured according to the block.

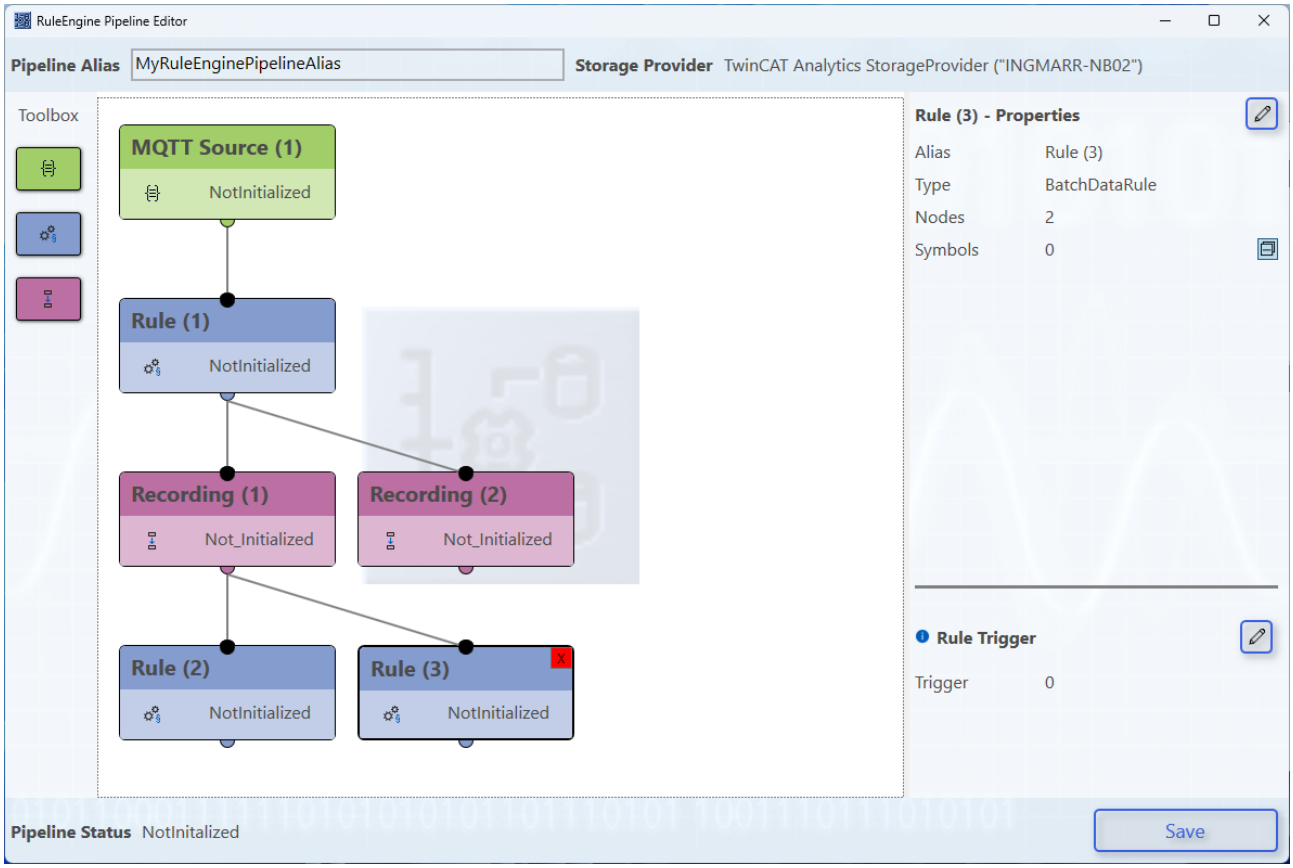
The following symbols are provided for each selected symbol:

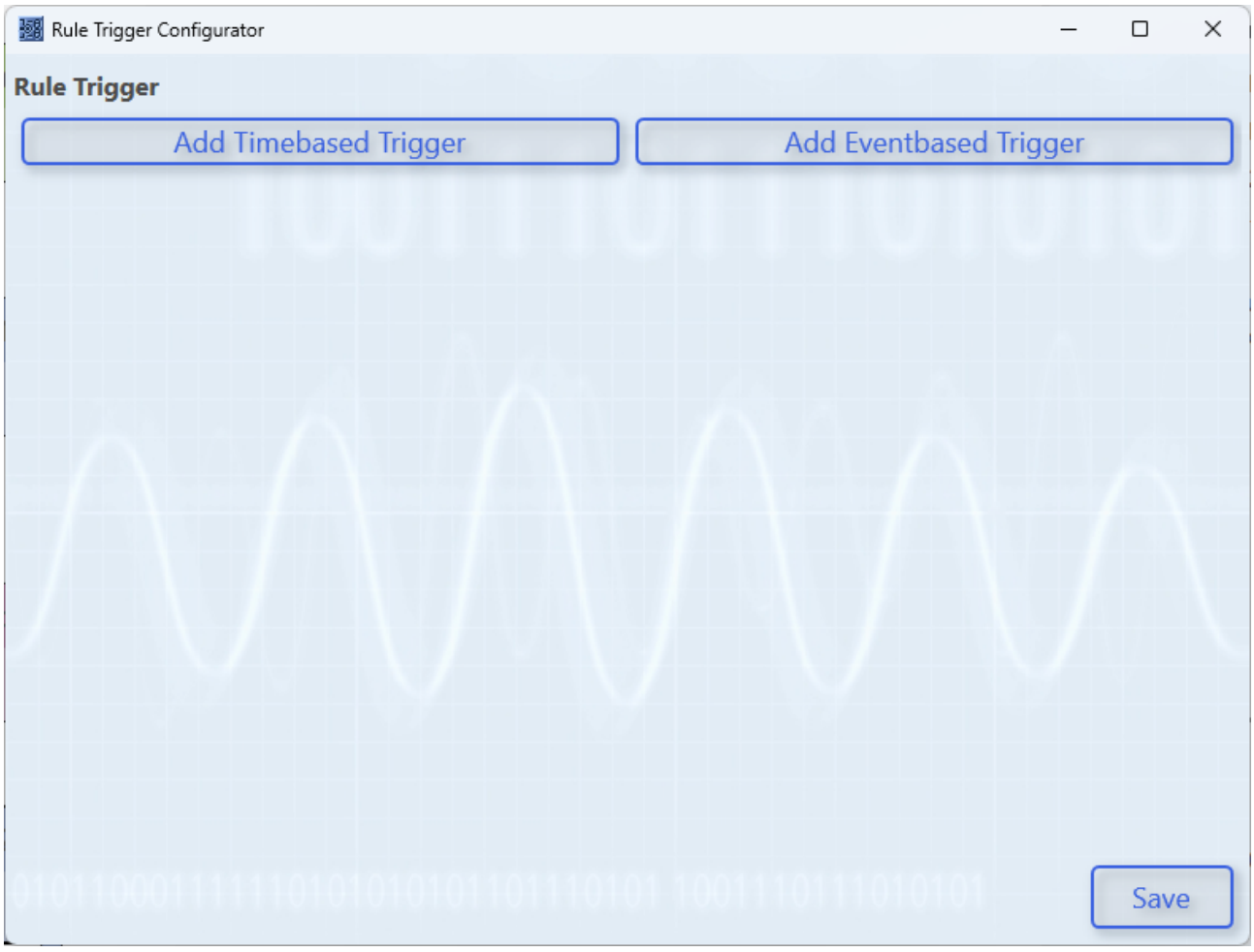
- **_STD**: Standard deviation of the symbol



6.2.2.1.4.2.5 Trigger

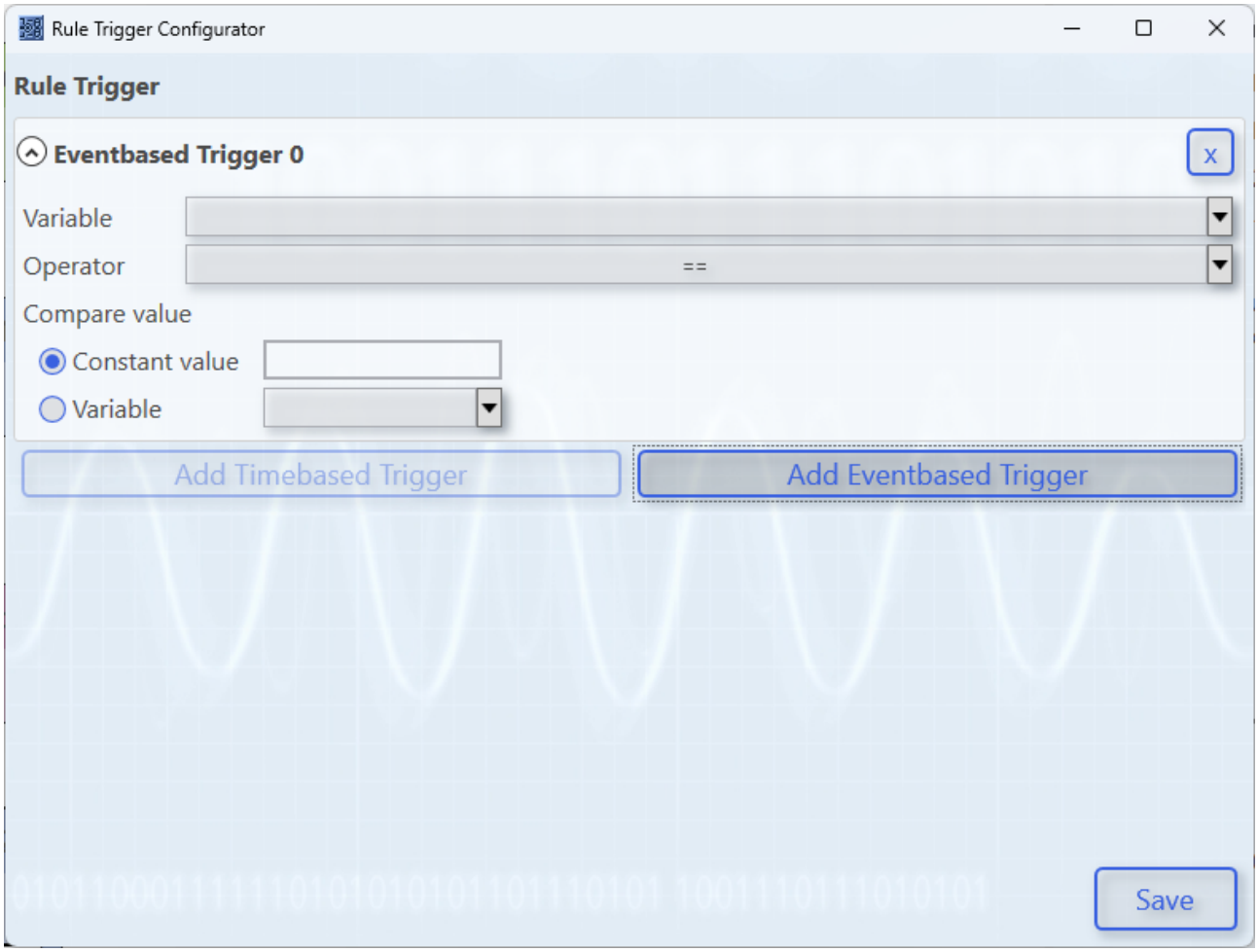
Triggers can be configured for a BatchDataRule (based on data from a recording). These triggers allow the rule to start time-based or event-based. A rule can also be triggered manually at any time. The triggers are configured in the Rule Trigger Configurator. This can be opened via the property window of the rule. The time-based and event-based triggers can be configured in the Rule Trigger Configurator. Currently, only time-based or event-based triggers can be used for a rule.


RuleEngine Pipeline Editor
Pipeline Alias
MyRuleEnginePipelineAlias
Storage Provider
TwinCAT Analytics StorageProvider ("INGMARR-NB02")
Toolbox
MQTT Source (1)
NotInitialized
Rule (1)
NotInitialized
Recording (1)
Not_Initialized
Recording (2)
Not_Initialized
Rule (2)
NotInitialized
Rule (3)
NotInitialized
Rule (3) - Properties
Alias
Rule (3)
Type
BatchDataRule
Nodes
2
Symbols
0
Rule Trigger
Trigger
0
Pipeline Status
NotInitalized
Save


Rule Trigger Configurator
Rule Trigger
Add Timebased Trigger
Add Eventbased Trigger
Save

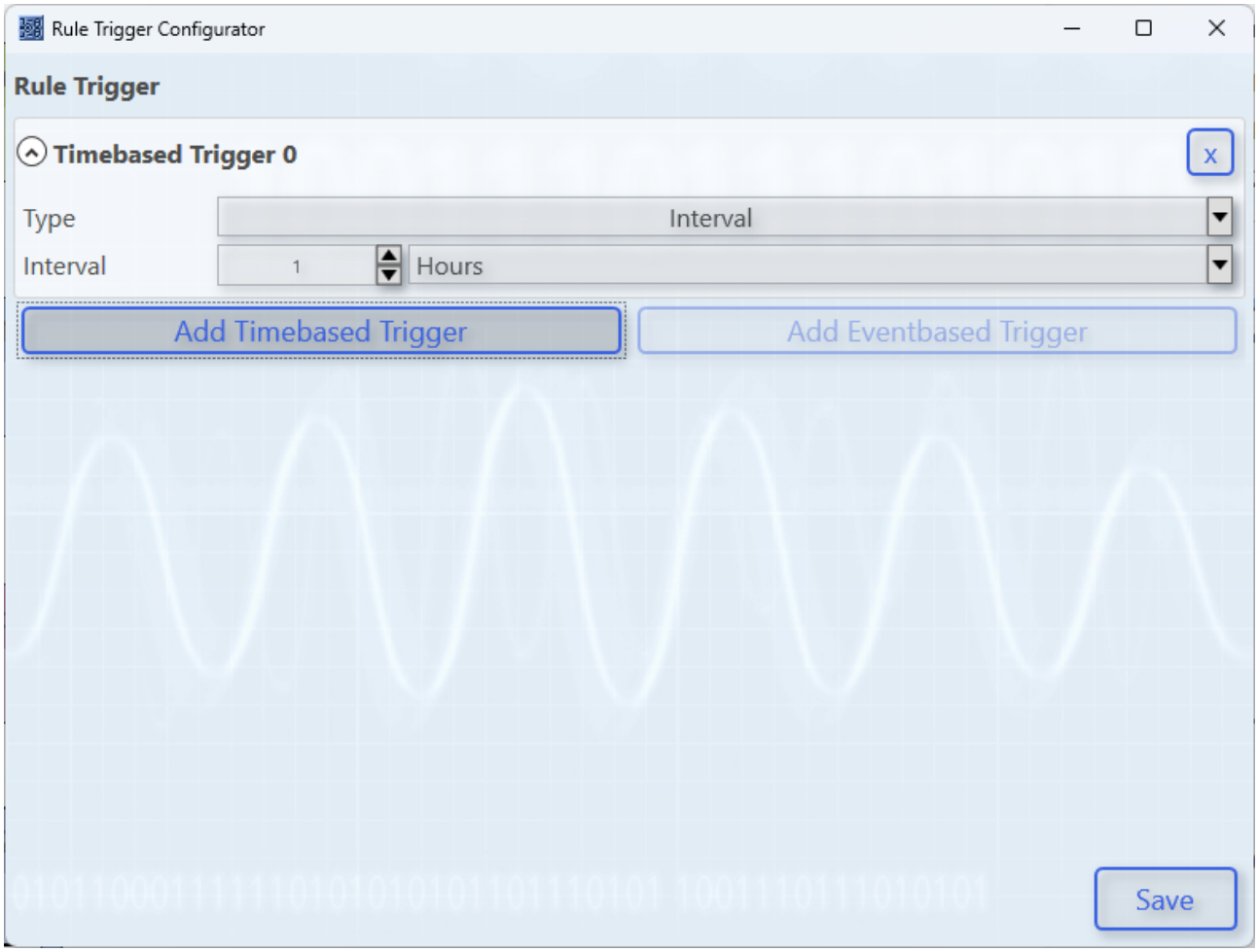
6.2.2.1.4.2.5.1 Eventbased

The configuration of an event-based trigger corresponds to an Eventbased-Condition [▶ 76] within the rule. The underlying data is the data that is currently being written to the recording that serves as the data source for the rule. It is not possible or necessary to configure the duration. If the trigger condition is fulfilled, the rule runs until the data has been processed accordingly and is then disabled until the next trigger.



6.2.2.1.4.2.5.2 Timebased

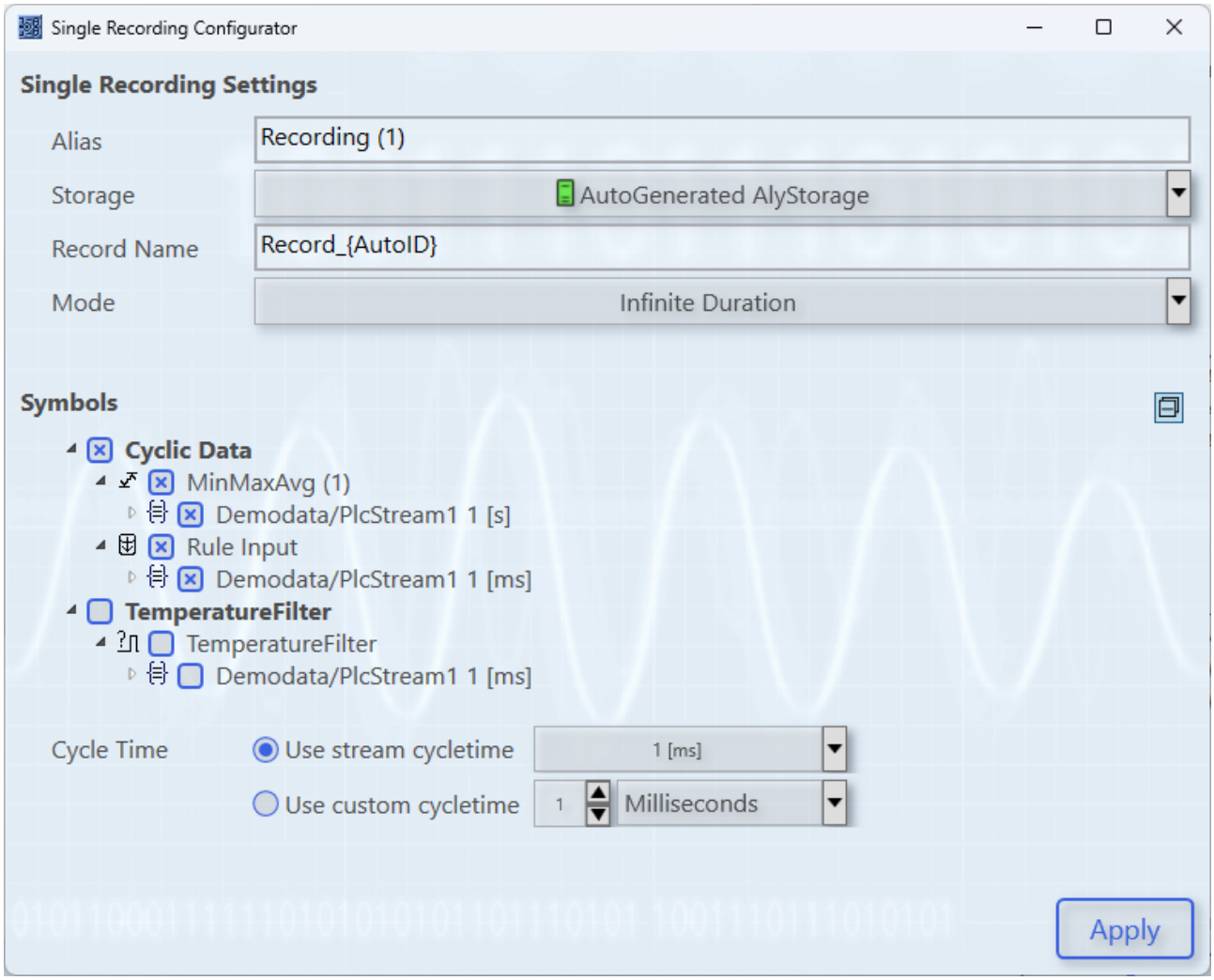
The configuration of a time-based trigger corresponds to a Timebased-Condition [▶ 77] within the rule. The underlying time is the system time of the device on which the Storage Provider is running. It is not possible or necessary to configure the duration. If the trigger condition is fulfilled, the rule runs until the data has been processed accordingly and is then disabled until the next trigger.



6.2.2.1.4.3 Editing recordings

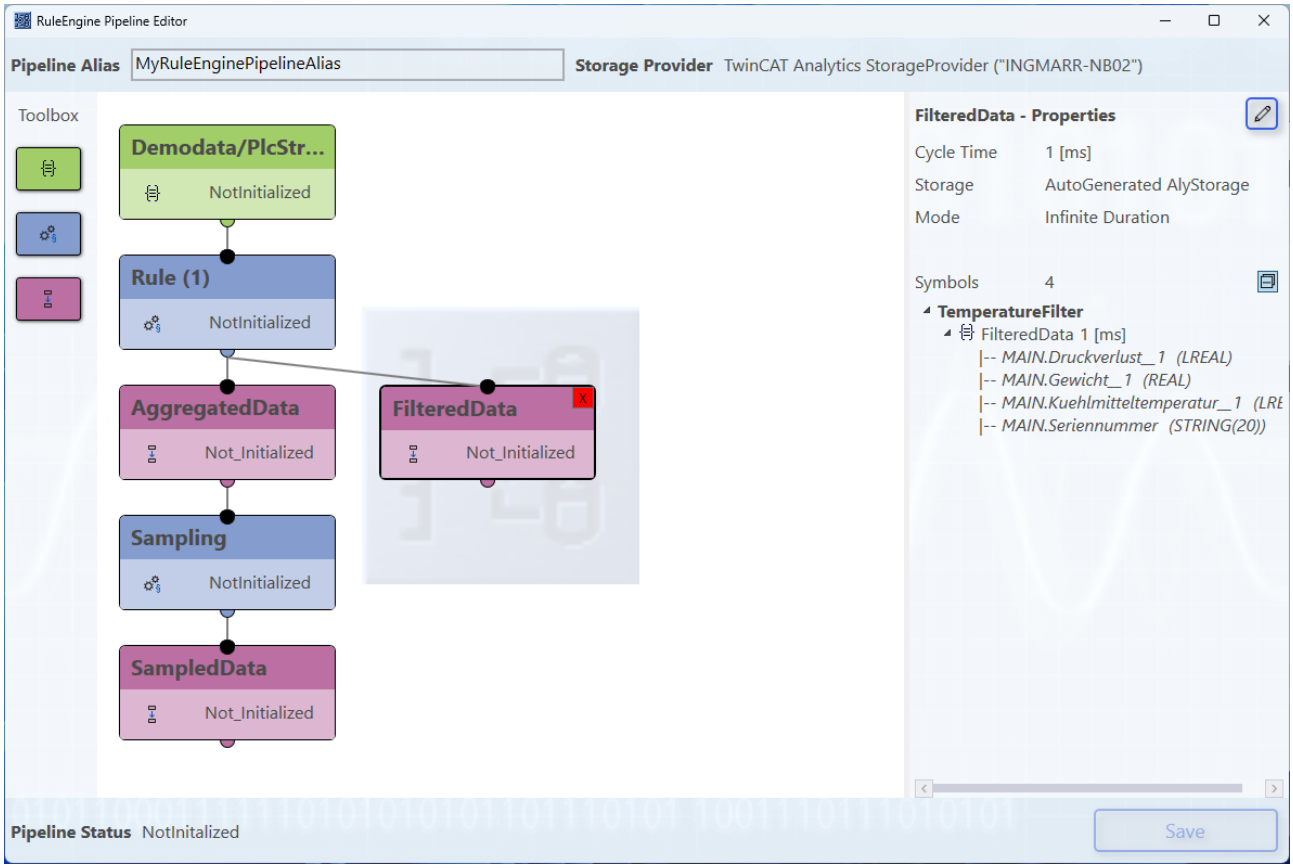
Recordings are edited in the Single Recording Configurator. The following settings are available here:

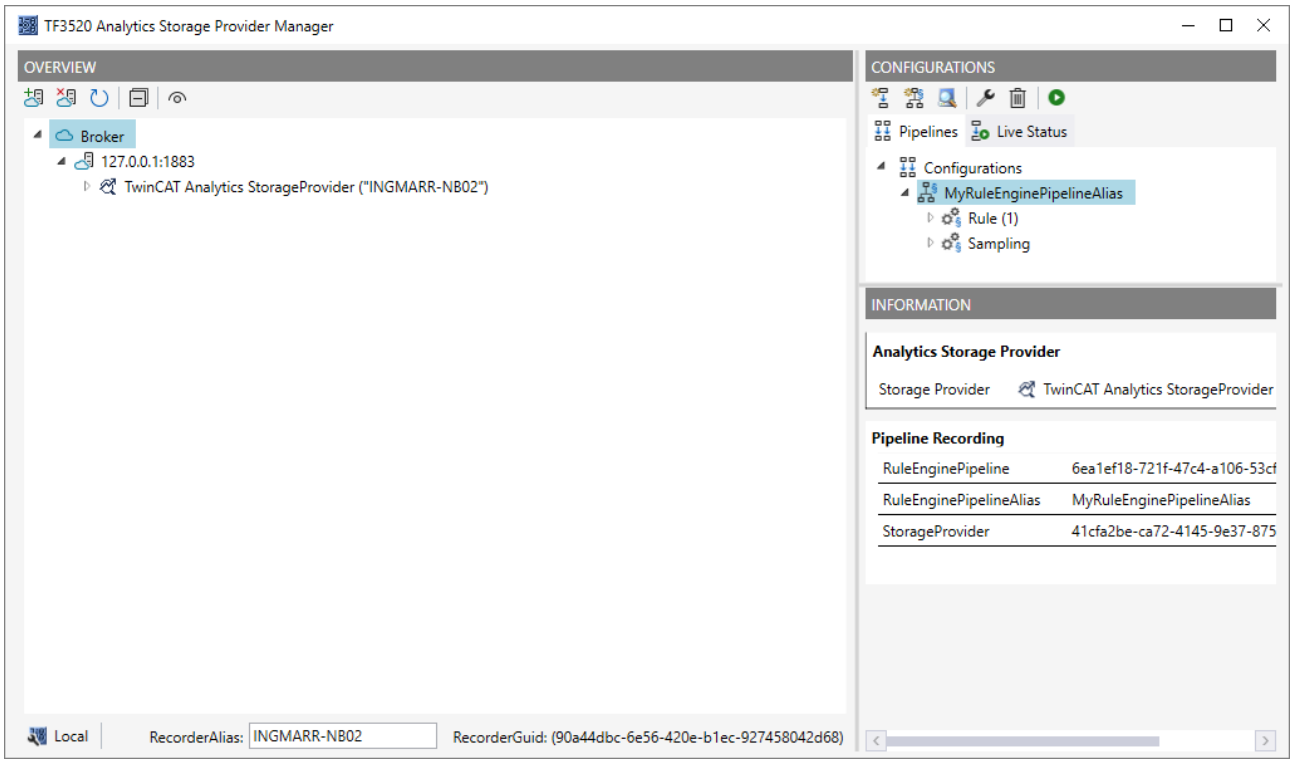
- **Alias**: Recording alias
- **Storage**: Data sink for recording (configured in the Storage Provider Configurator)
- **Record Name**: Name of the record
- **Mode**: Mode of the record
  - **Infinite Duration**: The record runs as long as the RuleEngine pipeline is running and data is available for the record.
  - **Timebased Ringbugger**: The record runs as long as the RuleEngine pipeline is running and data is available for the record. The data is stored in a time-limited ring buffer.
  - **Databased Ringbugger**: The record runs as long as the RuleEngine pipeline is running and data is available for the record. The data is stored in a ring buffer with a limited storage capacity.
- **Symbols**: Selection of symbols to be saved with the recording. You can choose between cyclic data or data filtered by conditions.
- **CycleTime**: Selection of the cycle time for recording. This selection can only be made if you want to save cyclic (unfiltered) data. You have the option of configuring a cycle time based on the variables or your own cycle time for recording. Note that data will be lost if the cycle time is greater than the minimum cycle time of the data. In this case, a message appears.



6.2.2.1.5 Saving a RuleEngine pipeline

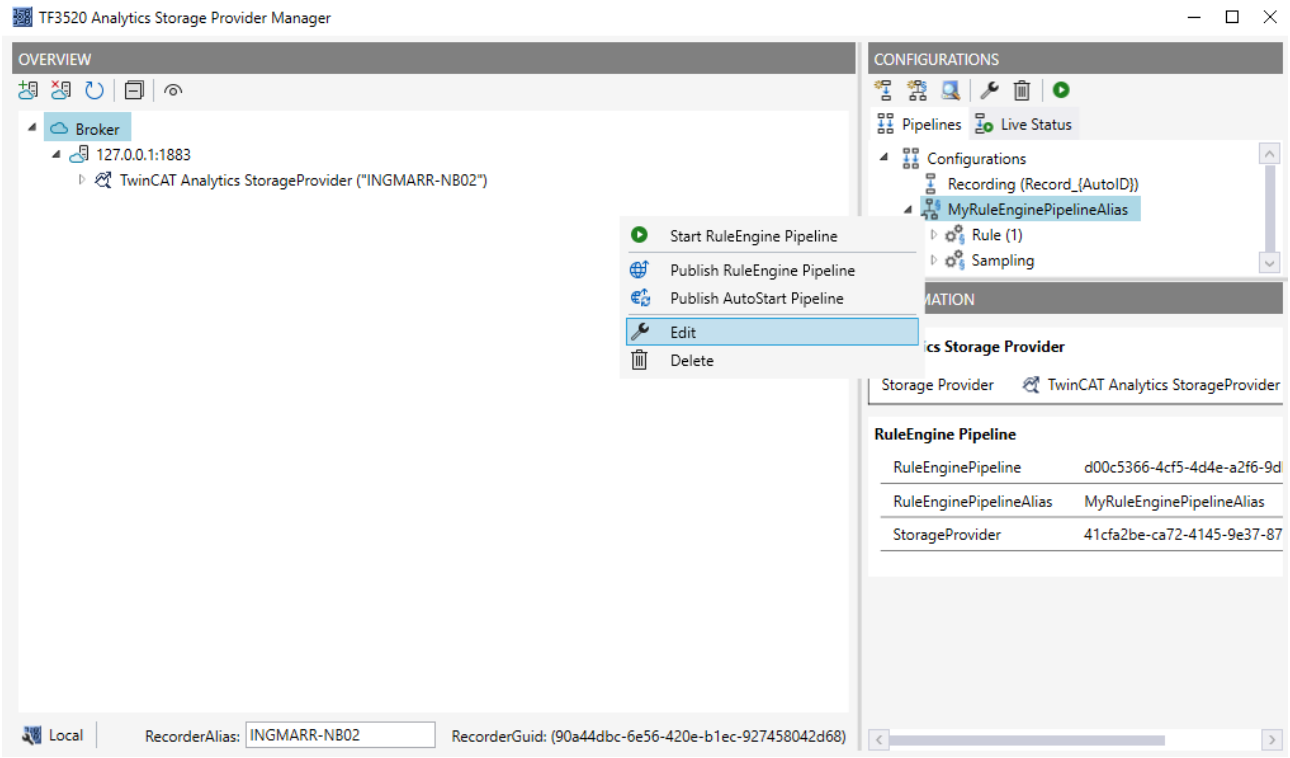
A configured RuleEngine pipeline can be saved using the **Save** button (bottom right) in the RuleEngine Pipeline Editor. Saving is only possible if the configuration is error-free. Any errors are displayed via a message and the incorrectly configured blocks are highlighted graphically.





6.2.2.1.6 Subsequent editing of a RuleEngine pipeline

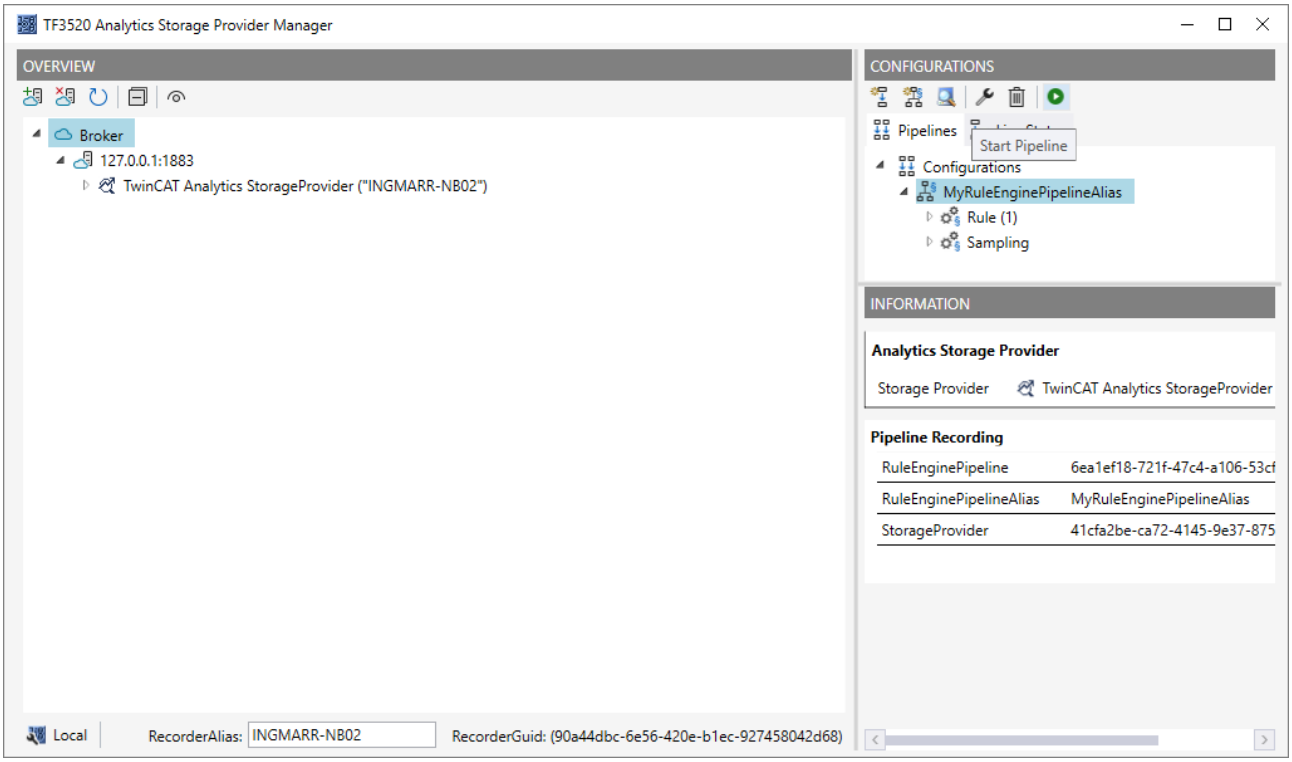
A RuleEngine pipeline that has been created can be edited and modified by right-clicking **Edit** in the context menu. Please note that existing data can no longer be used in the pipeline if the recordings contained in the pipeline are adjusted.



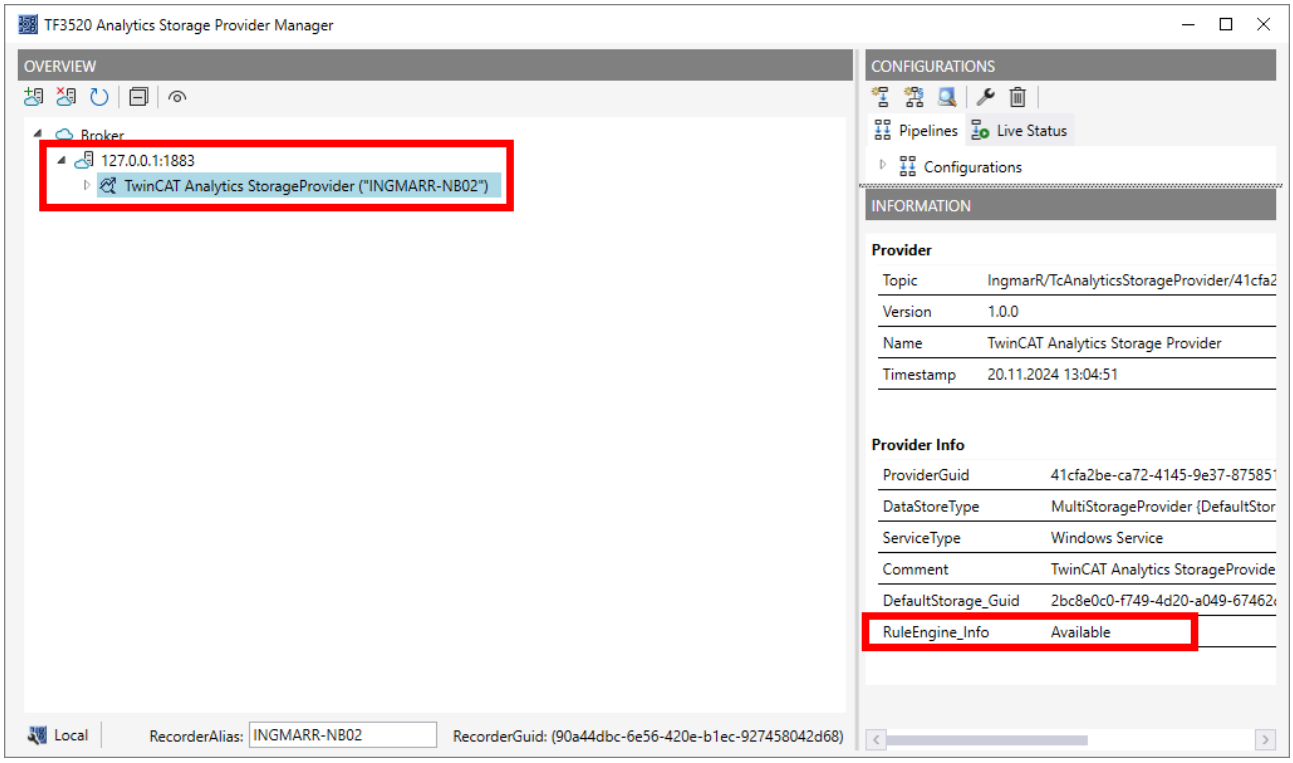
6.2.2.2 Application

6.2.2.2.1 Starting/stopping a RuleEngine pipeline

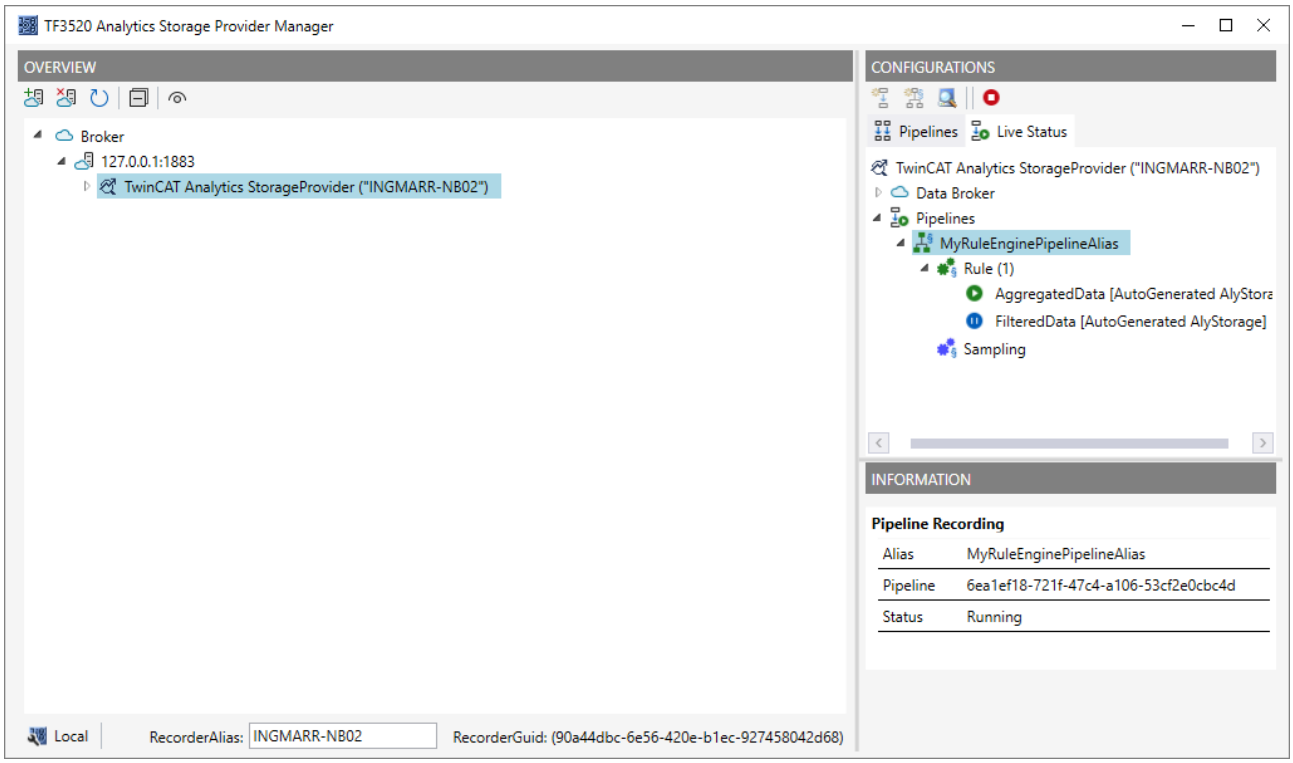
A configured RuleEngine pipeline can be started via the **Start** icon.



A start is only possible if the Storage Provider is running and the RuleEngine in the Storage Provider is running without errors. The status of the RuleEngine can be read via the information of the respective Storage Provider.

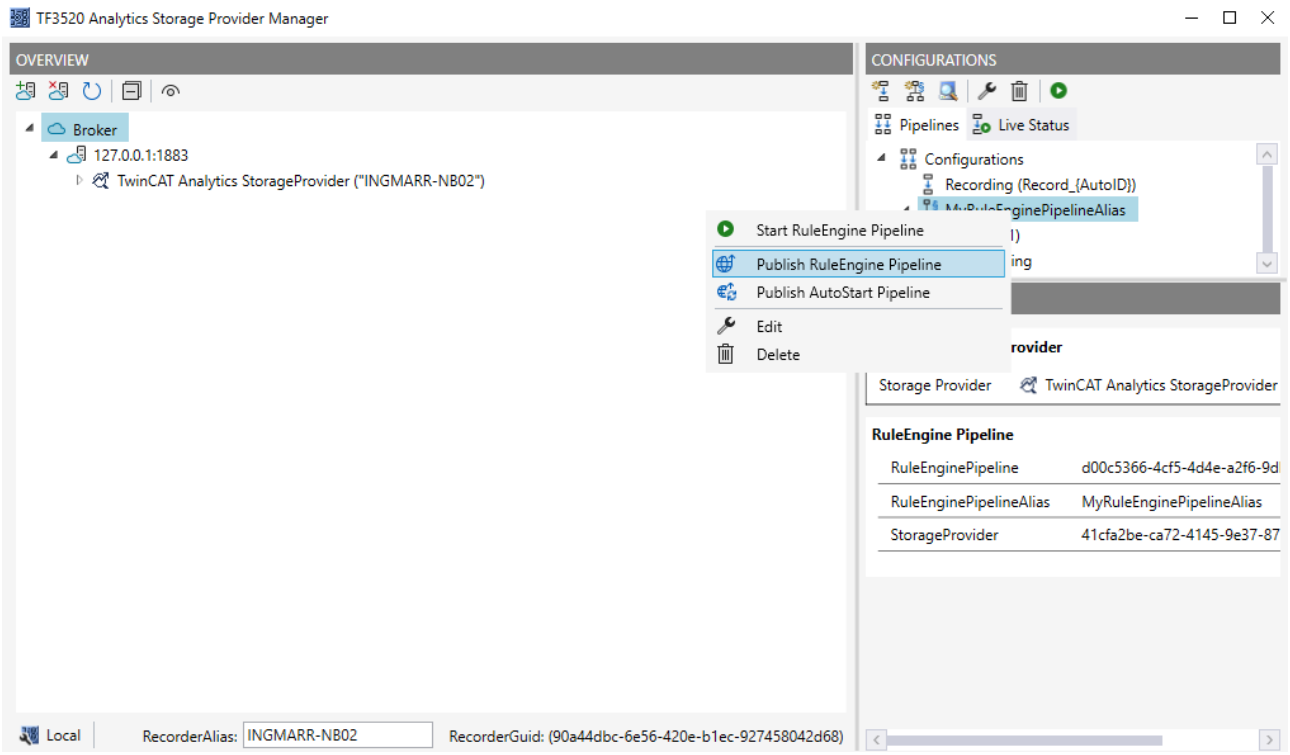


You can view RuleEngine pipelines that have been started in the Live Status tab and stop them there again.

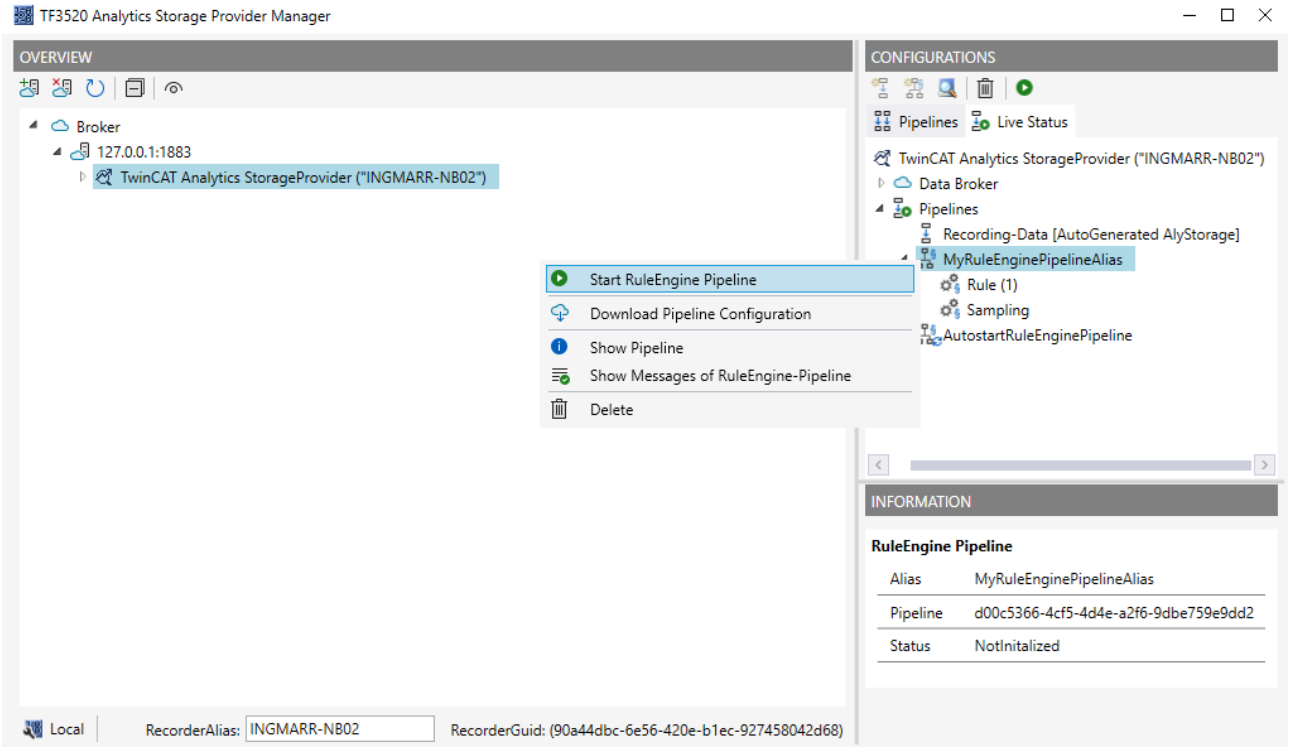


6.2.2.2.2 Publishing RuleEngine pipelines

Locally configured RuleEngine pipelines can be published to the stored Storage Provider. This means that the RuleEngine pipeline is also stored at the Storage Provider and is globally available. Other users can download, edit or start it. To publish, right-click on the locally configured pipeline in the **Publish RuleEngine Pipeline** context menu. In addition, RuleEngine pipelines can be published in Autostart mode. This means that the pipelines are started automatically after the Storage Provider Service is started.



All published pipelines can be viewed under **Live Status**. These can be started, deleted or downloaded.



6.2.2.2.3 RuleEngine pipeline states

A RuleEngine pipeline can have the following states:

	**NotInitalized:** The started RuleEngine pipeline was received by the Storage Provider but not processed.
	**Initalizing:** The RuleEngine pipeline is initialized. The corresponding rules are created.
	**IsStarting:** The RuleEngine pipeline is starting. Included rules are started.
	**Running:** The RuleEngine pipeline is running. All rules are at least in the Pending state and there is no error.
	**IsStopping:** The RuleEngine pipeline is stopping, all rules are stopping.
	**Stopped:** The RuleEngine pipeline is stopped, all rules are stopped.
	**Error:** There is an error in at least one rule.

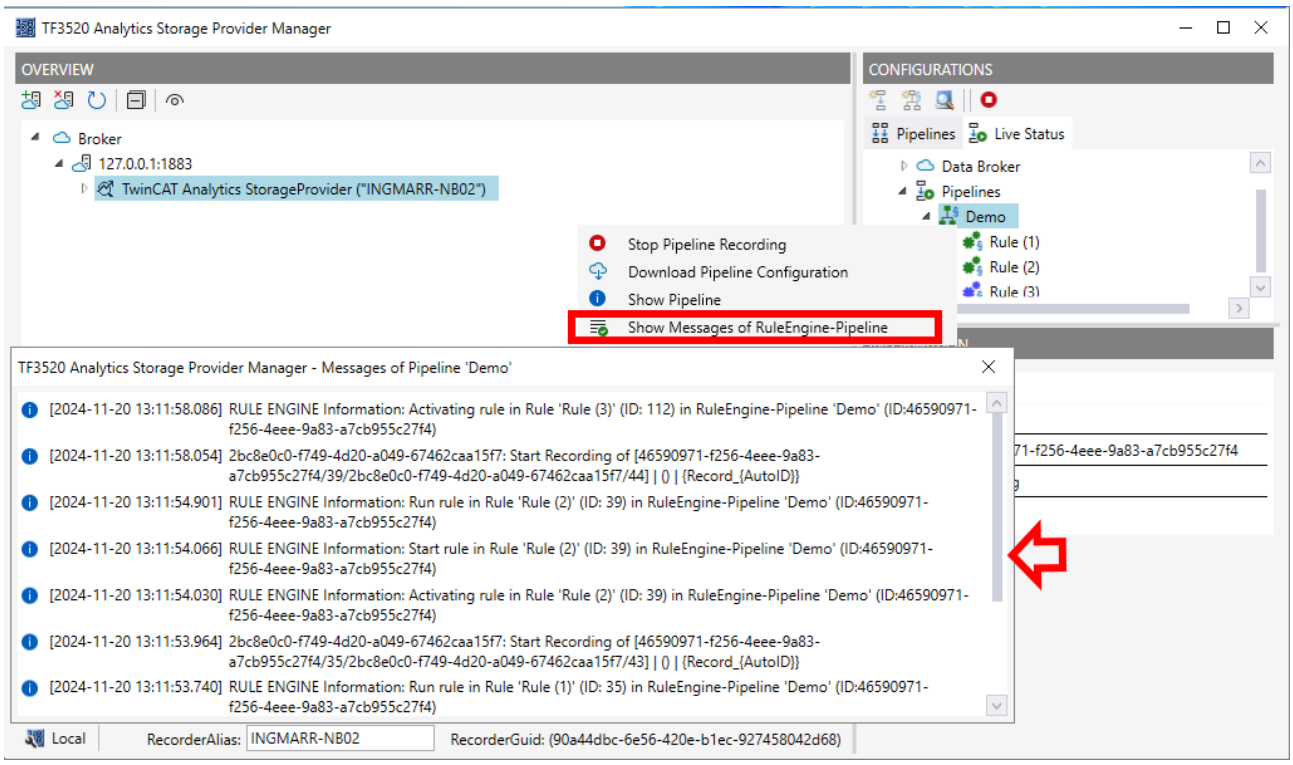
6.2.2.2.4 Rule states

You can also view the state of the rules from the RuleEngine pipeline in the Live Status. A rule has the following states:

	**NotInitialized:** The configuration of the rule has been read, but the rule has not yet been processed.
	**Initializing:** The rule is initializing. The necessary sources are generated in the RuleEngine.
	**Deactivated:** The rule is ready for activation. It is not currently running.
	**Activating:** The rule will be activated. Any triggers will be started.
	**Pending:** The rule is activated. Any triggers have been started. The rule can now be triggered manually. StreamingDataRules go directly from the Pending state to the Starting state.
	**Starting:** The rule is starting. The corresponding processing modules are generated and a connection to the data sources is established.
	**Running:** The rule runs and processes data.
	**Stopping:** The rule is stopping. Stopping is called when the RuleEngine pipeline will be stopped. BatchDataRules also stop when all data has been processed. The rule then switches to the Pending state.
	**Deactivating:** The rule will be deactivated. It is no longer possible to start up using a trigger.
	**Invalid:** The rule is in an invalid state and must be restarted.
	**Error:** The rule is in an error state. This can happen, for example, if a data source (Analytics Logger) is not available. In this case, the rule restarts automatically as soon as the data source is available again. For all other errors, the rule must be restarted manually. This can be done via the Storage Provider Manager or the API or PLC library.

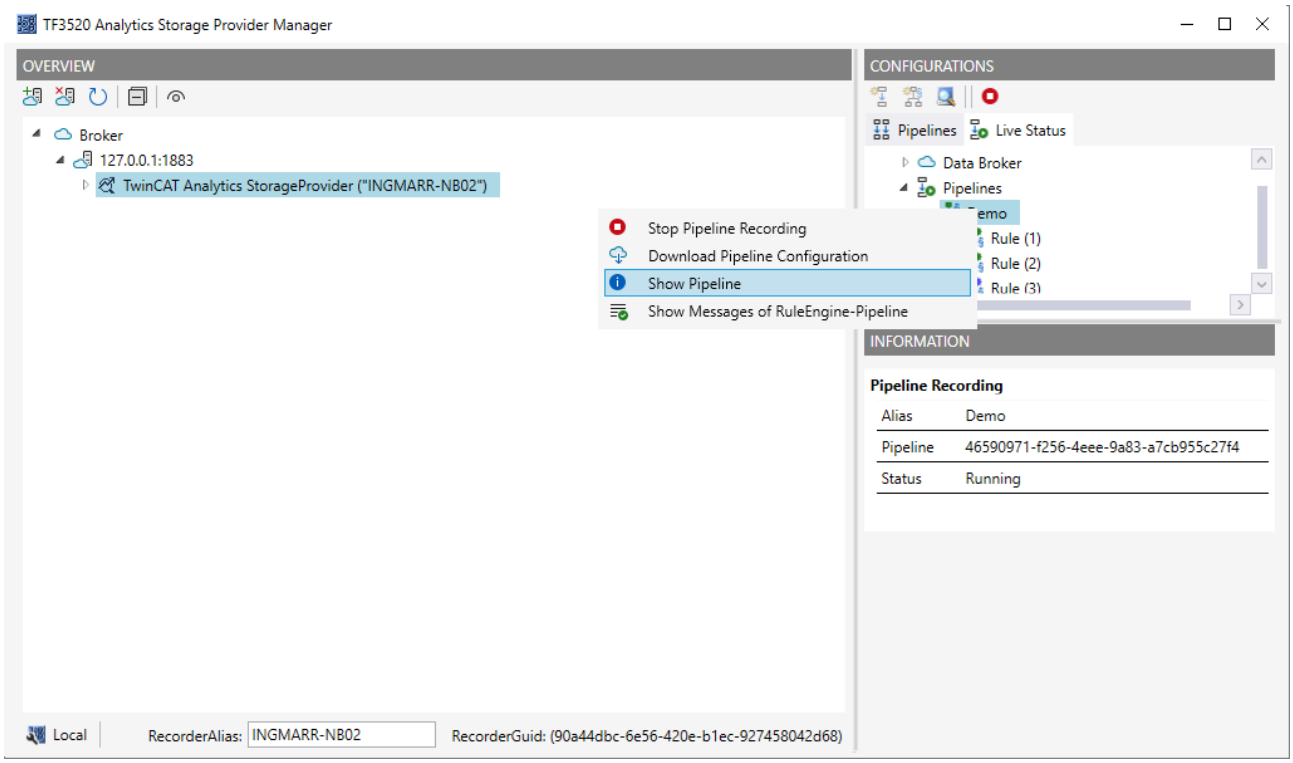
6.2.2.2.5 Messages from RuleEngine pipelines

The messages from RuleEngine pipelines provide more detailed information about errors and status information. The messages can be viewed via the context menu of a running RuleEngine pipeline. To do this, click on **Show Messages of RuleEngine-Pipeline**. A window with all available messages opens. If there are no messages in the window, these messages are no longer available. More accurate information can be found in the log from the Storage Provider, if this is activated.

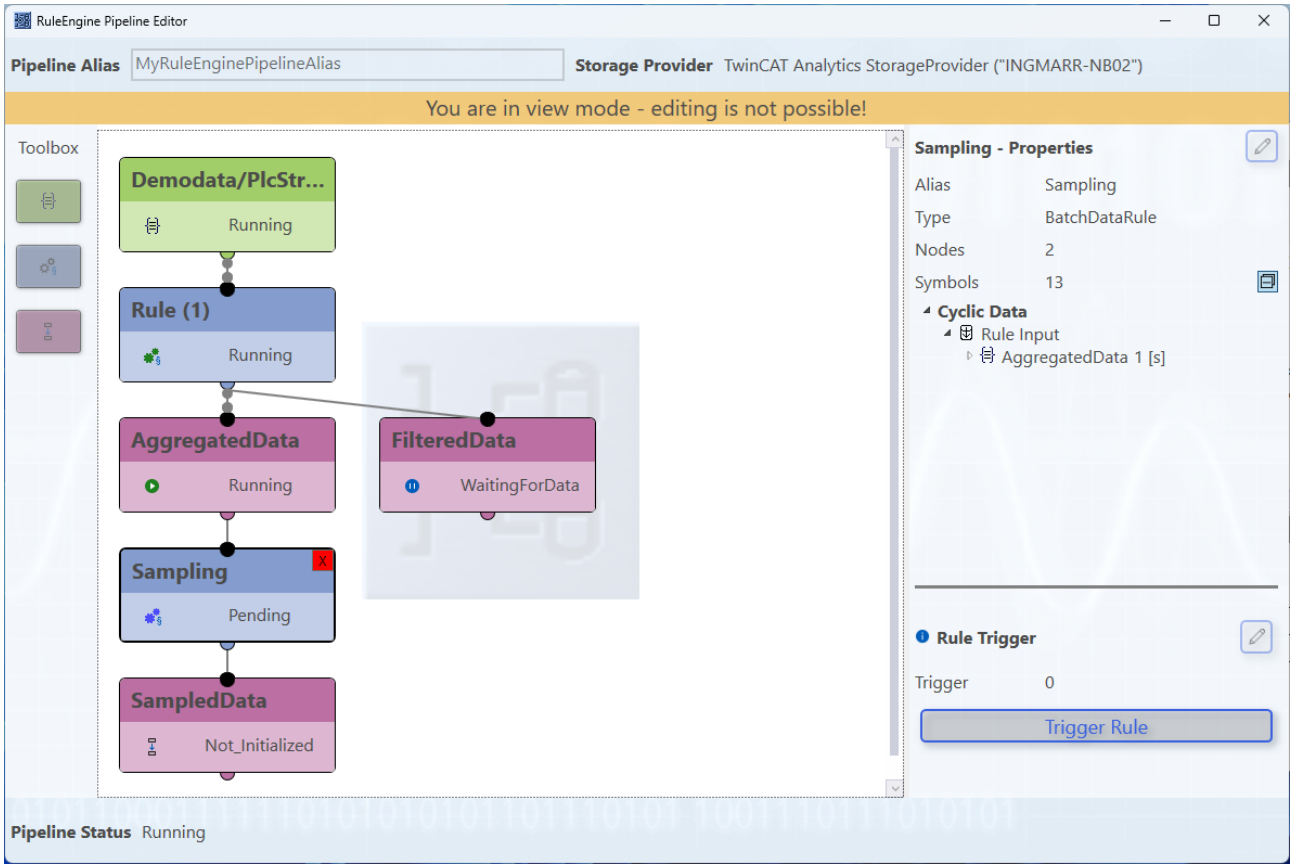


6.2.2.2.6 Triggering and restarting rules

Rules can be triggered or restarted via the RuleEngine Pipeline Editor. The editor must be opened via a running RuleEngine pipeline (**context menu > Show Pipeline**).

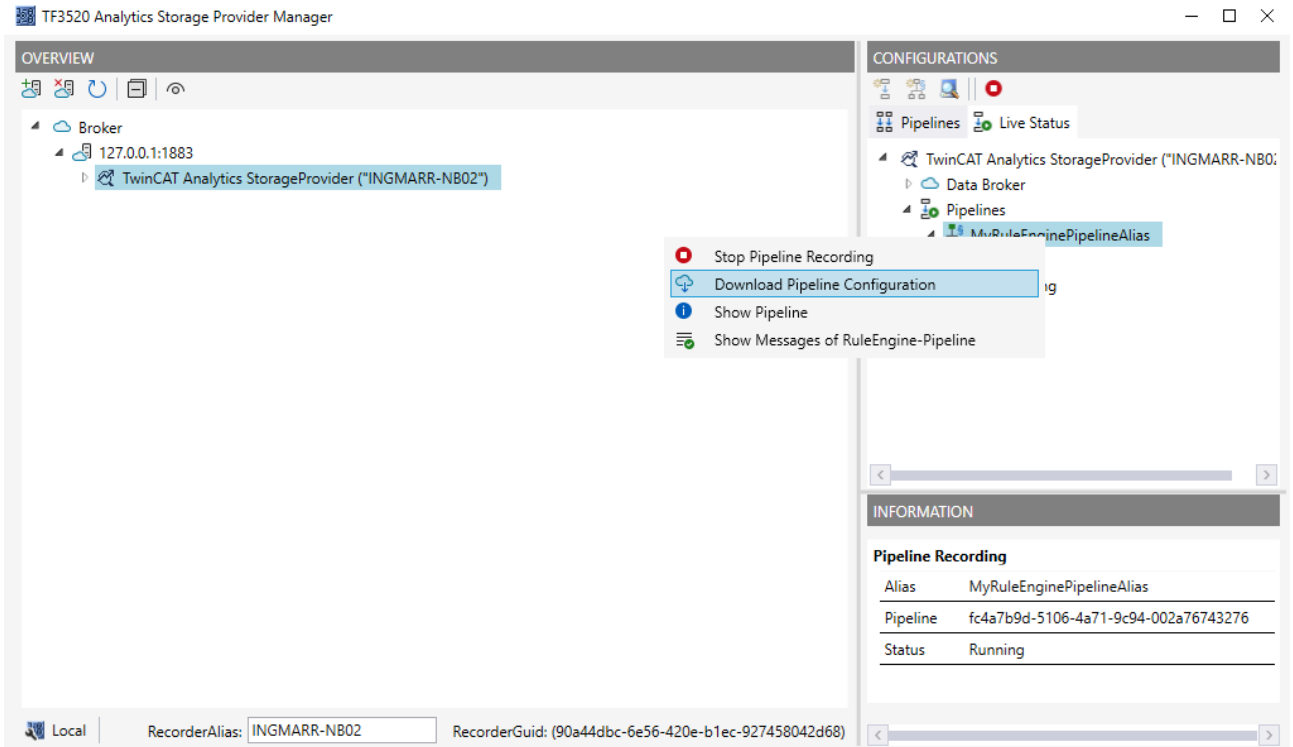


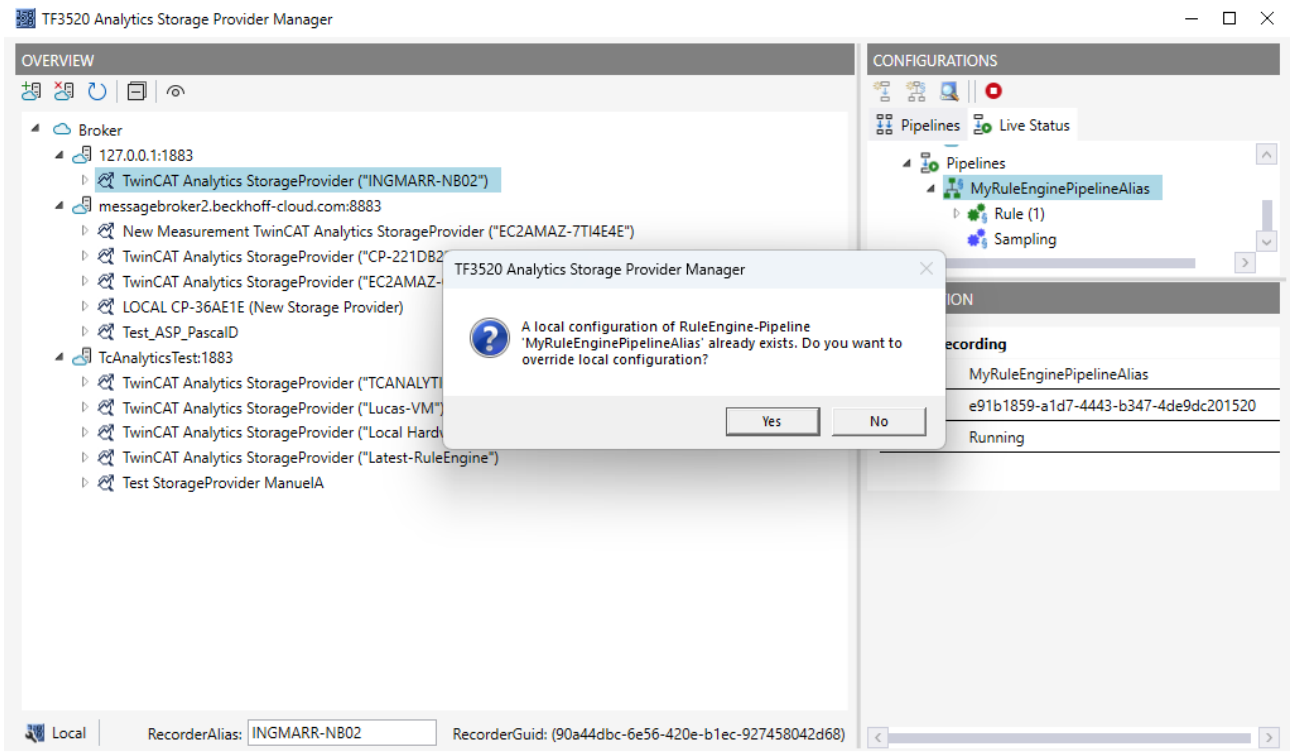
The RuleEngine Pipeline Editor opens in view mode. Processing is not possible there. In view mode, the status of the individual elements can be viewed and the corresponding rule can be triggered or restarted. Triggering is only possible when the rule is in the Pending state. A restart can only be performed if the rule is in an error state or has not been initialized.

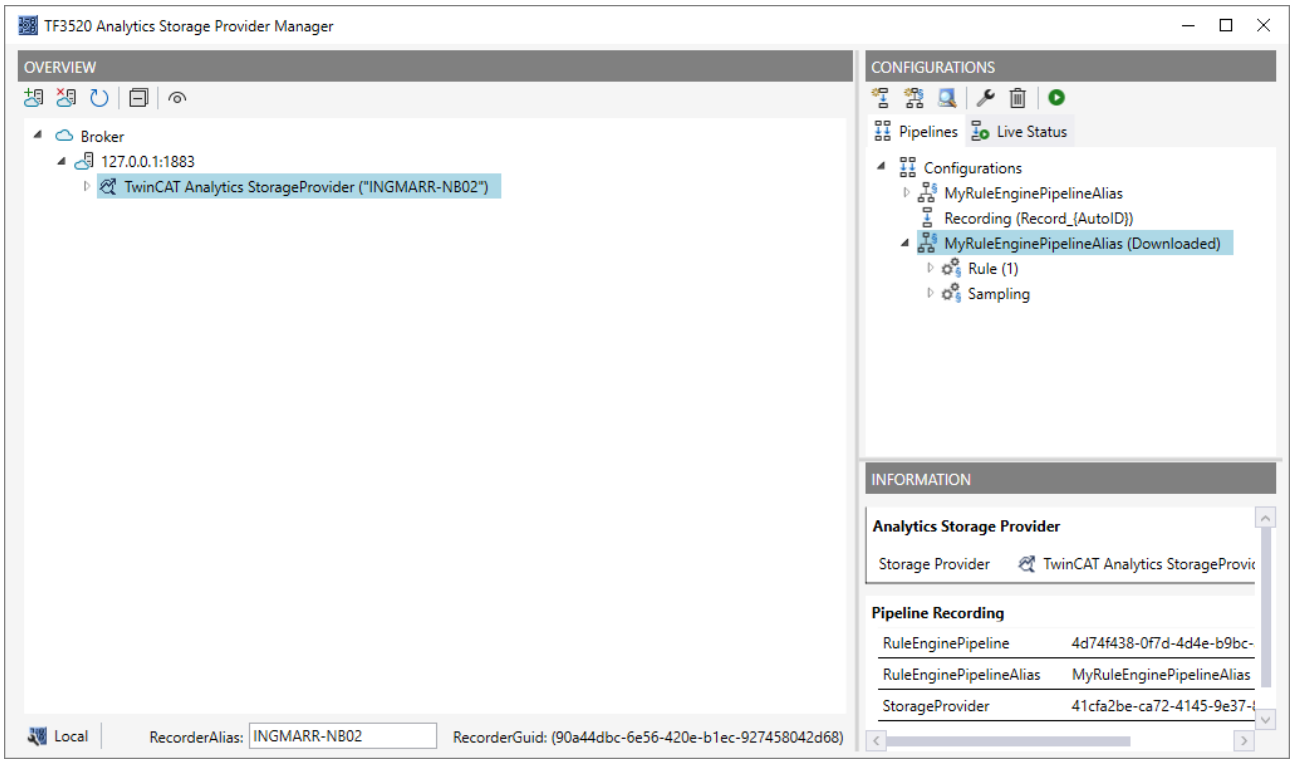


6.2.2.2.7 Download RuleEngine pipelines

A copy of a running RuleEngine pipeline can be downloaded to the local manager by **right-clicking > Download**. If a RuleEngine pipeline with the same pipeline ID already exists, you can choose to overwrite it or create a copy with a new pipeline ID.

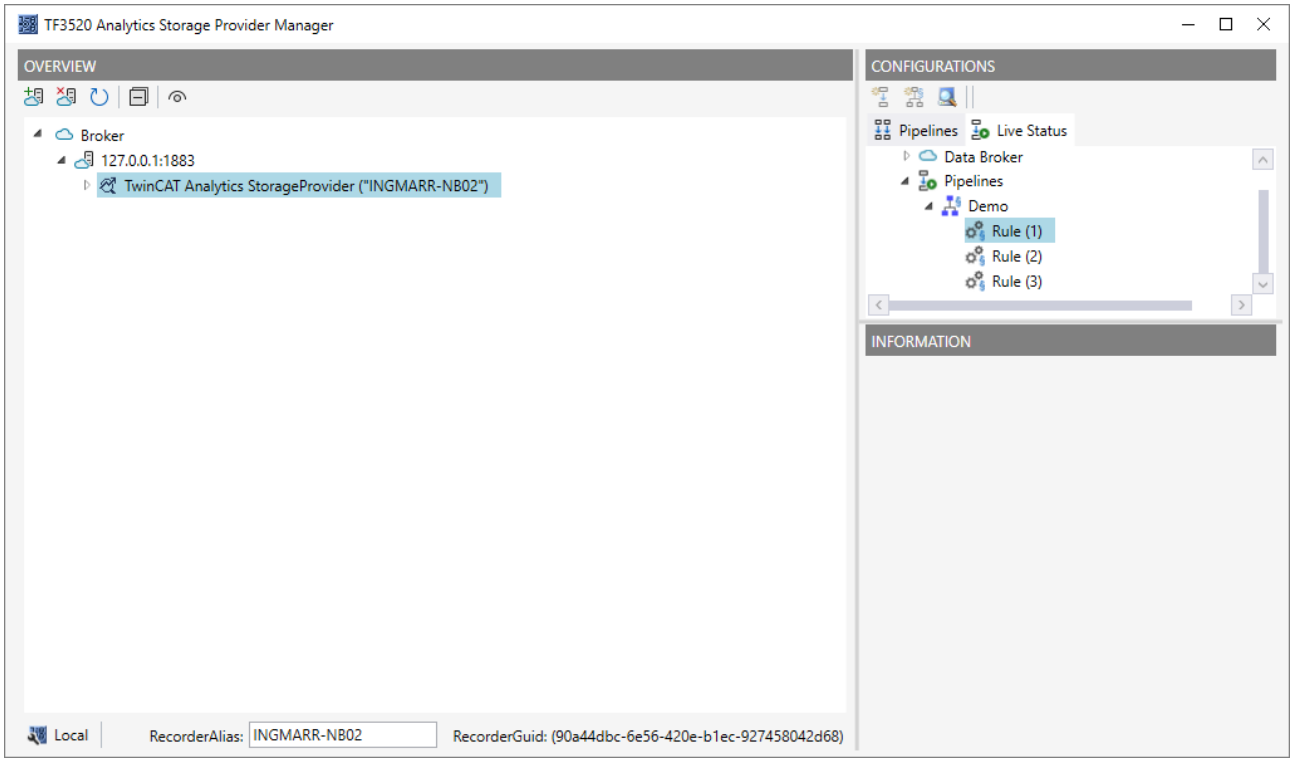


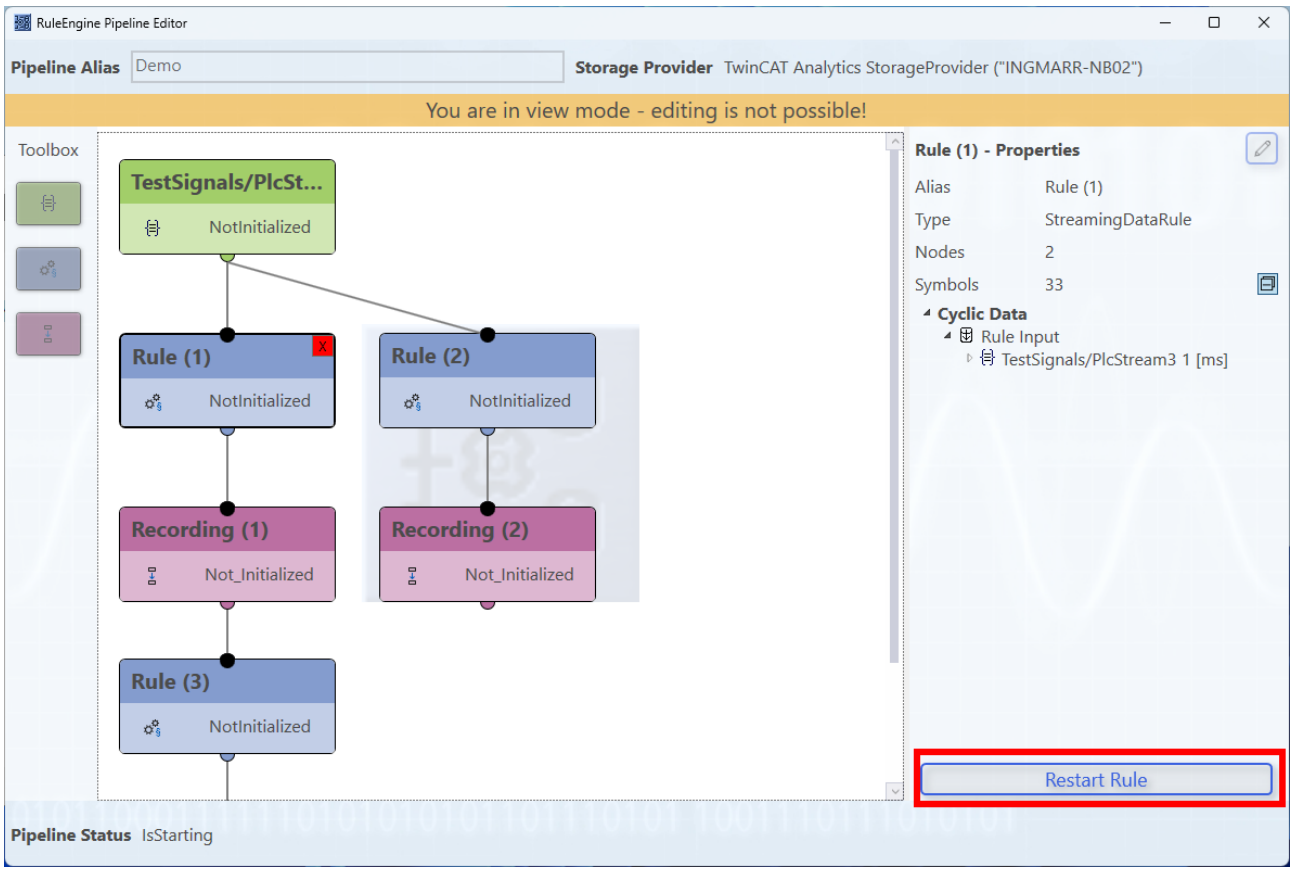




6.2.2.2.8 RuleEngine error case

If an error occurs within the RuleEngine that it cannot resolve on its own, the RuleEngine is restarted. All existing RuleEngine pipelines are shut down for this purpose. After the RuleEngine is restarted, the RuleEngine pipelines are created again. However, the rules contained therein are not automatically restarted. This must be done manually [▶ 91] using the RuleEngine Pipeline Editor.

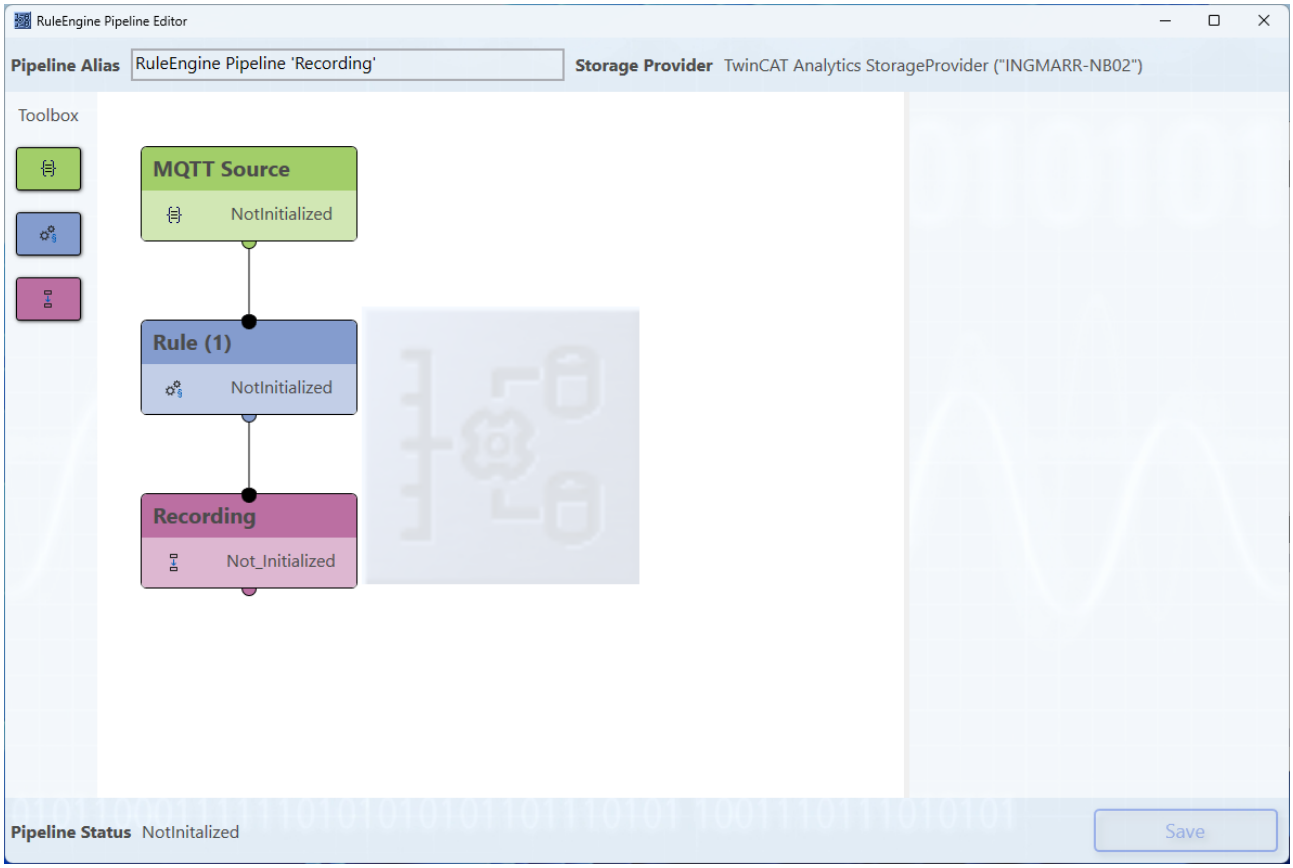




For a more detailed description of the error, see the messages or the log of the Storage Provider.

6.2.2.3 Converting old recordings

An existing pipeline (previous recording) can be converted into a RuleEngine by right-clicking **Convert to RuleEngine-Pipeline**. The new configuration opens in the RuleEngine Pipeline Editor and can be edited there.



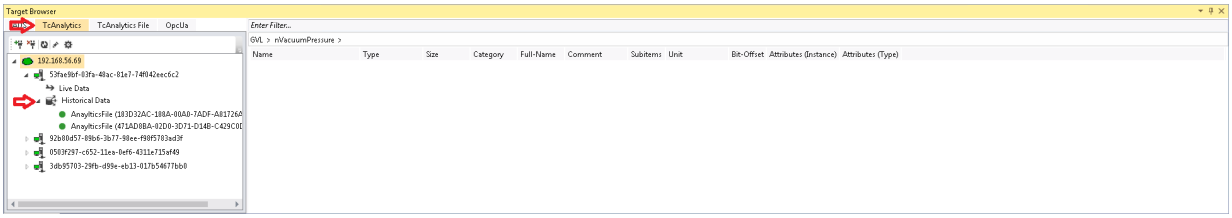
6.3 Working with Historical Data

Historical Data can be analysed with the Analytics Workbench or the Analytics Service Tool. To see your recorded data, you need the TwinCAT Target Browser.

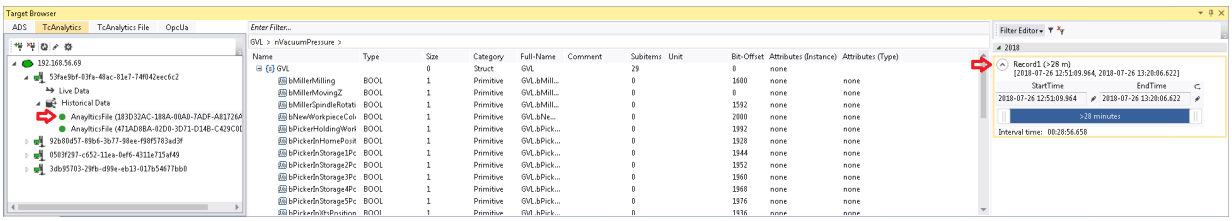
Selection of data from the TwinCAT Target Browser

The historical data can be pulled directly from the Target Browser to an input of an analysis algorithm.

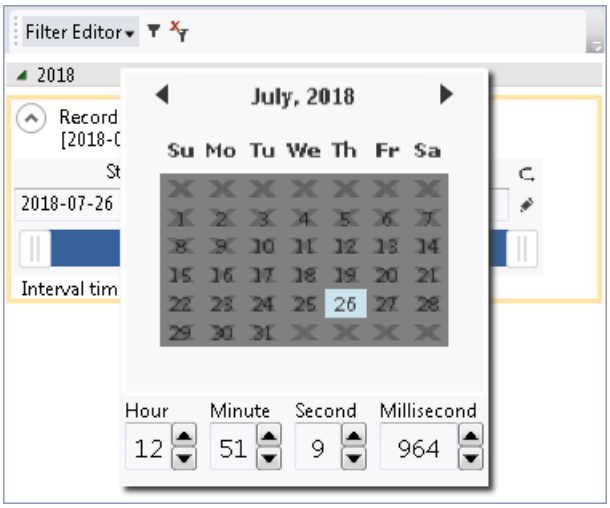
1. First, you need to click **TcAnalytics** in the left corner of Target Browser. There you can see your configured broker, which lists live and historical data from your various devices. This should look like the following figure.



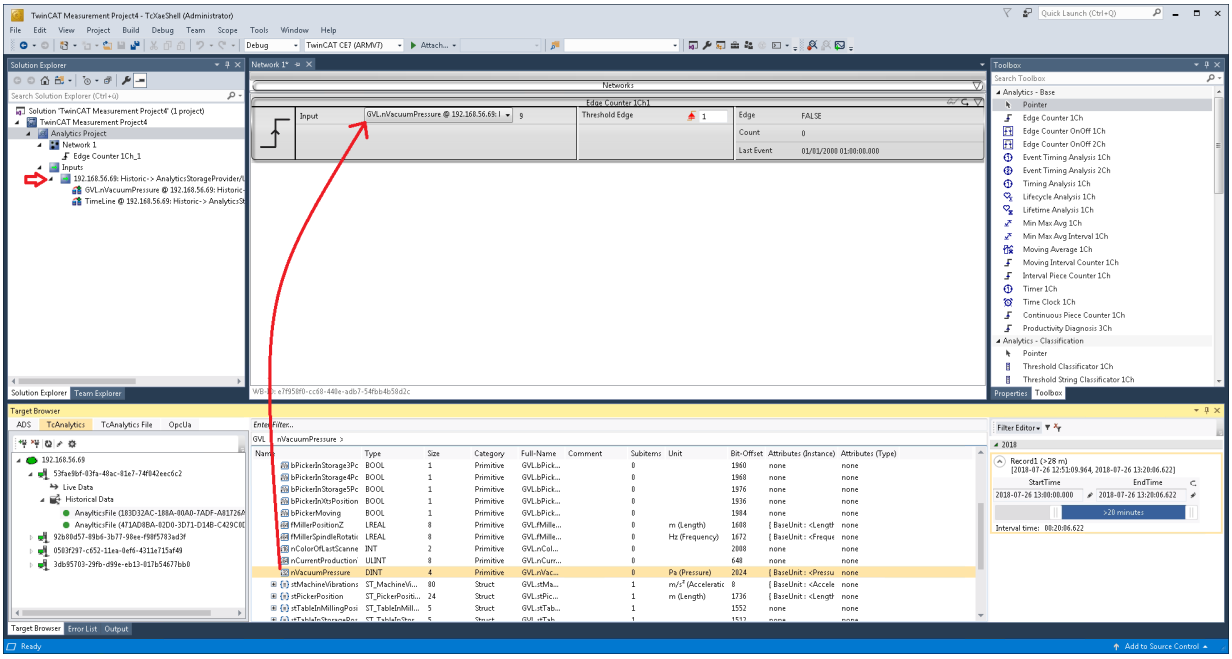
2. Go to the historical stream you created and select the recording to be analyzed. All your records are listed in the **Record** window on the right. The last recording is selected by default.



3. When you record live, the time range of the recording is updated every few seconds. The entire time range of a recording is used by default. You can also edit the start and end time to analyze your desired data area. This can be done with a slider, text fields or in a graphical calendar view. If you click on the symbol to the right of the text fields, the calendar view will be displayed.



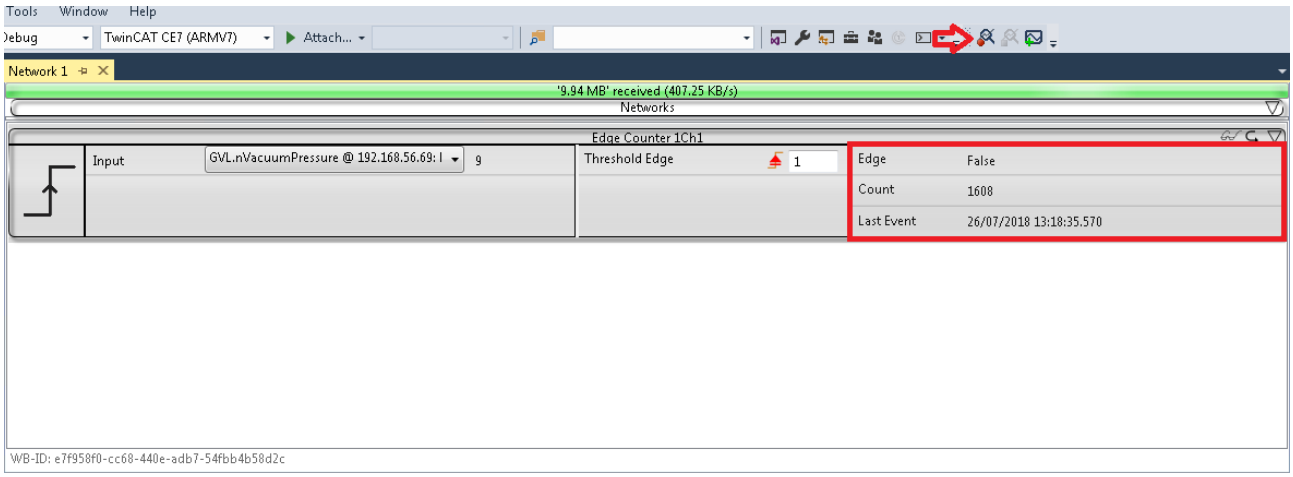
4. After these steps, you can drag and drop a symbol to an input of an algorithm just as you do with the symbols of the live data.



⇨ A new input source for your historical stream is then generated and can be displayed in the Solution Explorer of your Visual Studio®. First, the dragged symbol and a timestamp of the current device time are listed under this stream. Also new drawn symbols of this stream are listed there.

Analyse your historical data in the Analytics Configurator

To analyse your historical data press on the Start Analytics button. In contrast to analysing live data, a green progress bar appears. The speed of your analysis depends on your record length, the amount and size of your symbols as well as on your broadband speed to the broker. The analysis stops automatically when the progress bar ends. The results will remain visible.



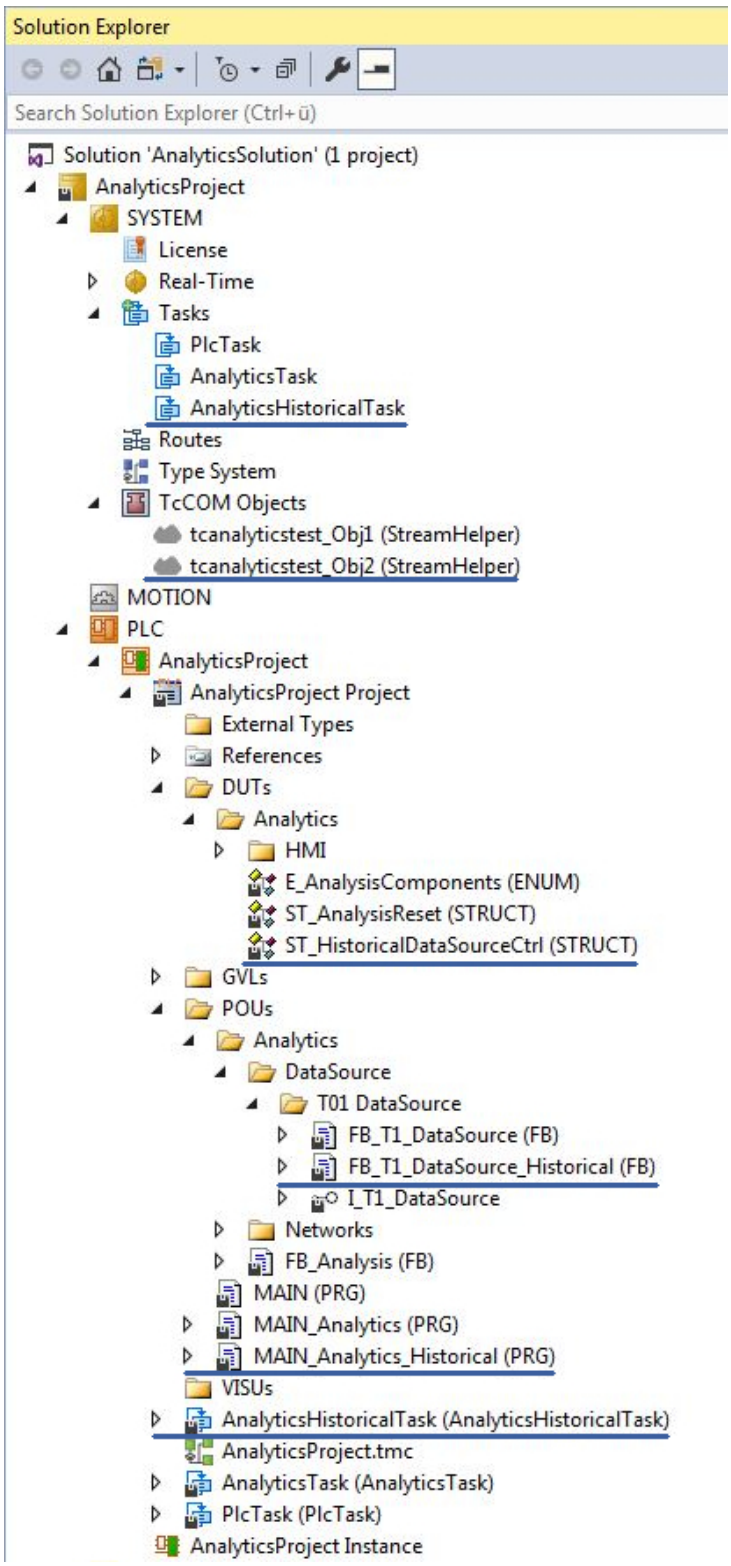
Analysis of your historical data in your Analytics Runtime

You can provide the configuration with your historical data to an Analytics Runtime (PLC). In addition to the historical data, the live data is also analyzed. This allows you to switch between them and not lose live data by streaming historical data. The reason for this is that they are separated into two different tasks. The start of the analysis of historical data must be triggered.

Computing time for historical data

Unlike the Analytics Configurator, the analysis of historical data in the PLC takes a similar amount of time as the original recording of the data. Depending on the packet size and the set sampling rate, the processing of the data can be shortened compared to the recording. However, cycle overruns due to excessively large packets must be taken into account.

Main differences of the folder structure in the created PLC project:

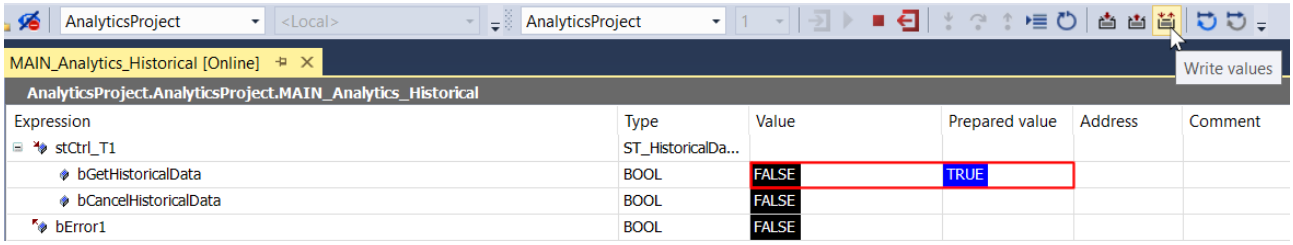


NOTICE

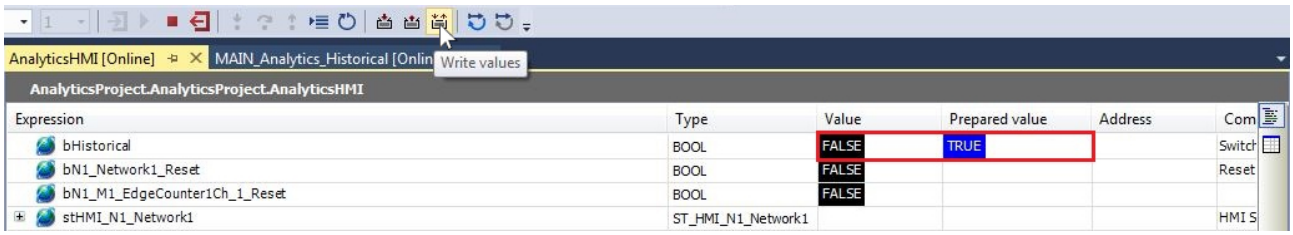
Implementation of the logic in your TwinCAT HMI

The preparation and writing of values in your PLC are for testing purposes. It is recommended to implement this and other logic in the PLC code with interactions from your TwinCAT HMI application if required.

You can start historical data analysis by triggering **bGetHistoricalData** in **stCtrl_T1**. The cancellation takes place by triggering **bCancelHistoricalData**.
This can be done in the **MAIN_Analytics_Historical** file as shown in the following figure:



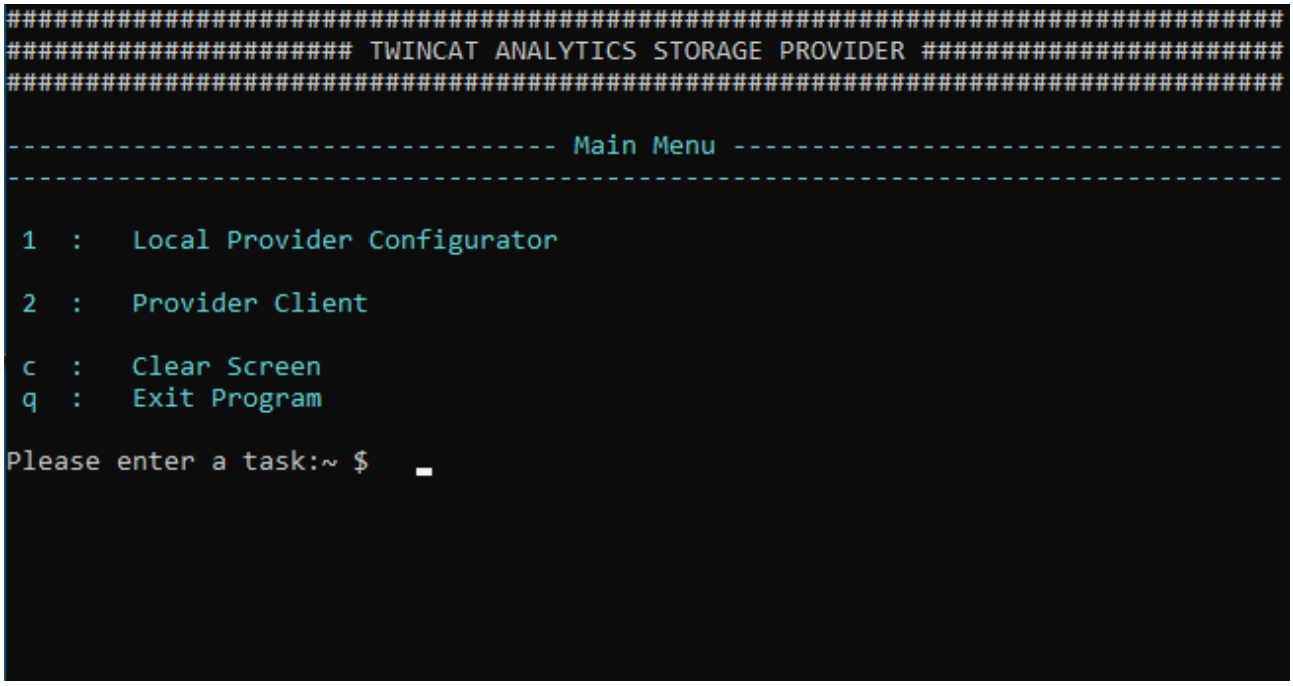
To switch between live and historical data results for your HMI dashboard, you can set the **bHistorical** symbol in the **AnalyticsHMI** GVL. With this option, you do not need any additional controls to display historical data (of course, you can also use your own controls for historical data). The analysis of the live data is not interrupted by calling up some historical data. After viewing the historical data, you can switch back to the current live results. This change only affects the variables in your GVL.



6.4 Console Configurator/Client

In addition to the graphical configurator and the recorder, the Analytics Storage Provider can also be operated via a console. This means that configuration and access to a Storage Provider can also be performed under TwinCAT/BSD in addition to Windows. In addition, the console application can be used to generate Batch files for control [▶ 104] of the Analytics Storage Provider.

After launching the console client, there are four options to choose from:



1	Opens the console Configurator [▶ 100] for the local Analytics Storage Provider
2	Opens the Analytics Storage Provider Client [▶ 101]
c	Clears the console configurator/client history
q	Closes the console configurator/client

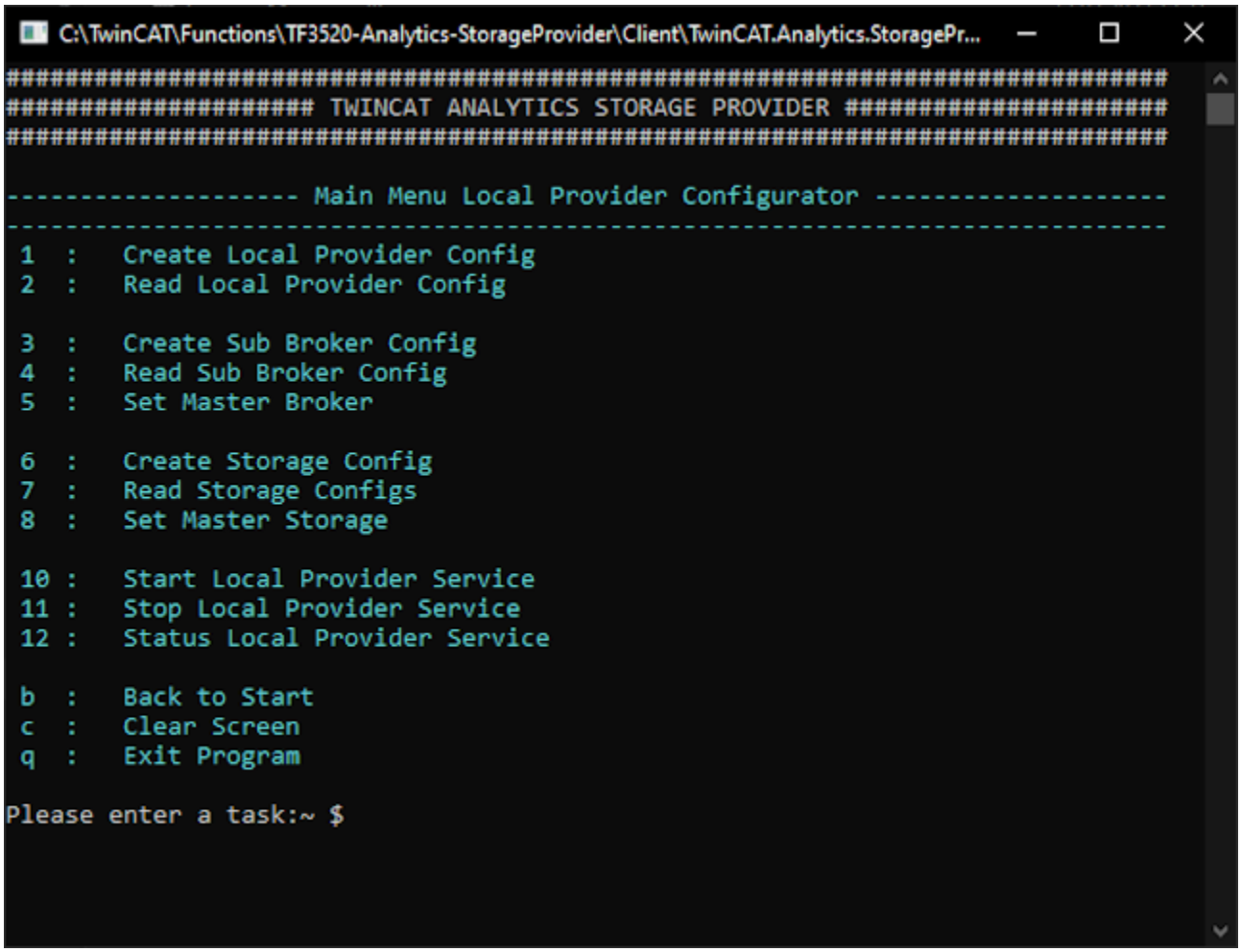
By entering one of the identifiers and confirming with the **[Enter]** key, the corresponding function is executed.

6.4.1 Configurator

In this menu, the local Analytics Storage Provider can be configured. The following additional inputs are available for this purpose:

1	Starts a dialog for configuring the local Analytics Storage Provider
2	Outputs the configuration of the local Analytics Storage Provider
3	Starts a dialog for configuring a Sub Message Broker
4	Returns all Sub Message Broker configurations
5	Specify the Master Sub Message Broker (commands can only be received via this broker)
6	Starts a dialog for configuring a Storage
7	Returns all Storage configurations
8	Specifying the Master Storage
10	Starts the Analytics Storage Provider with the local configuration
11	Stops the local Analytics Storage Provider
12	Displays the status of the Analytics Storage Provider
b	Switches back to the main menu

The configuration parameters are the same as in the graphical configurator of the Analytics Storage Provider. To create the configuration, the configuration parameters are received after entering:



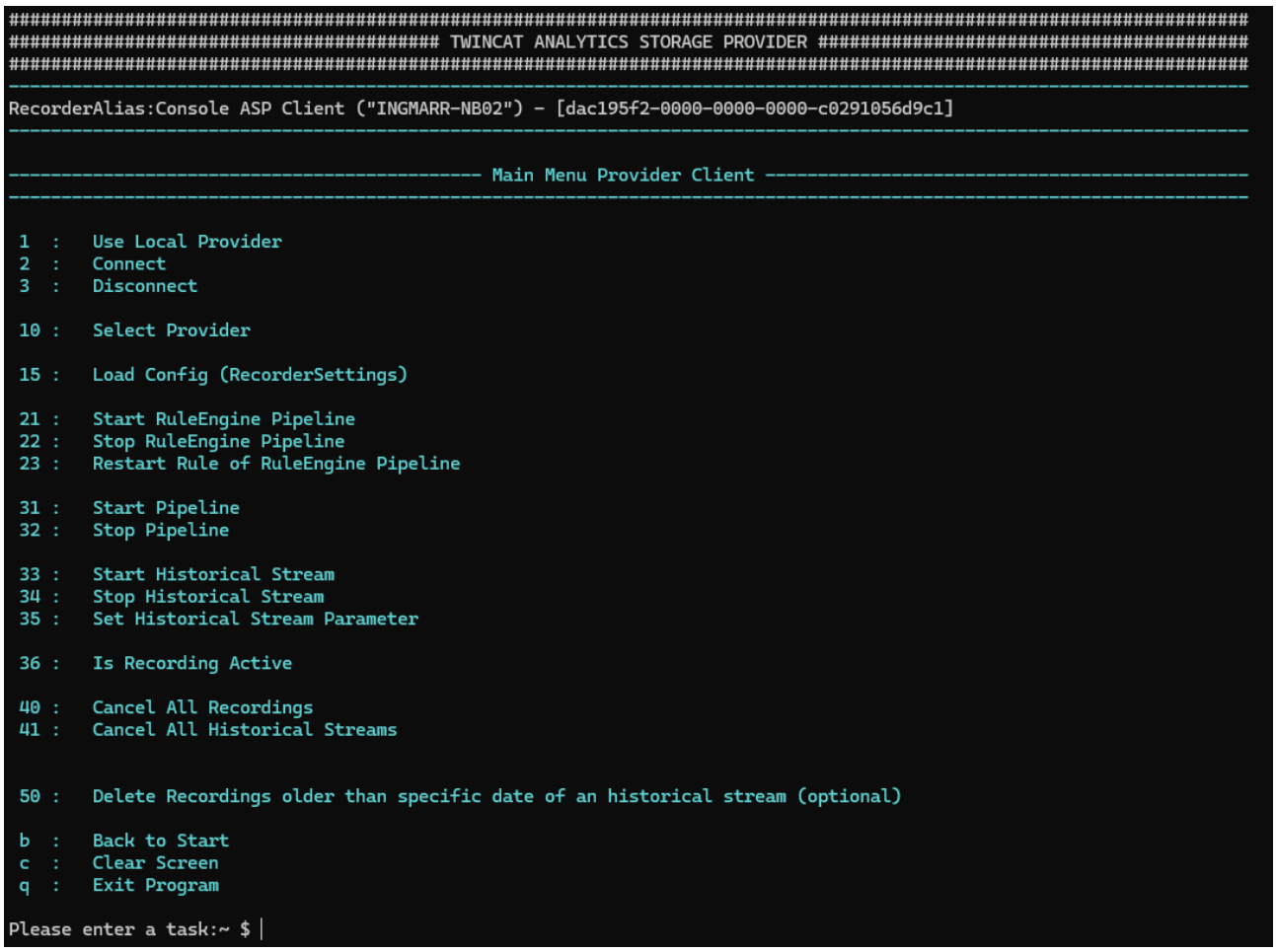
By pressing the **[ESC]** key, the dialog can be aborted at any time and you can return to the configuration menu.

6.4.2 Client

In the Analytics Storage Provider Client it is possible to connect to an Analytics Storage Provider. This does not necessarily have to run locally on the device, but can also be addressed via an external MQTT broker. Thus, it is not mandatory to configure an Analytics Storage Provider locally to use the client. Recordings can then be started and stopped on a connected Analytics Storage Provider, as well as historical data streams to a configurable MQTT topic.

The following inputs are available for this:

1	Establishes a connection with the MQTT broker from the local configuration and selects the configured Analytics Storage Provider.
2	Starts a dialog to connect to an MQTT broker.
3	Closes the connection to the current MQTT broker.
10	Provides the Analytics Storage Providers that are available under the MQTT broker for selection.
15	Reads a configuration file from an Analytics Storage Provider Recorder, with the recordings configured in it.
21	Starts a RuleEngine pipeline.
22	Stops a RuleEngine pipeline.
23	Restarts a rule of a RuleEngine pipeline.
31	Starts a recording.
32	Stops a recording based on the alias and the MQTT topic.
33	Starts a historical data stream.
34	Stops a historical data stream based on the result MQTT topic.
35	Updates streaming parameters of a running historical data stream.
36	Checks whether a recording is active.
40	Stops all active recordings.
41	Stops all historical data streams.
50	Deletes recordings that are older than a certain date. Optionally, a data stream can be specified.



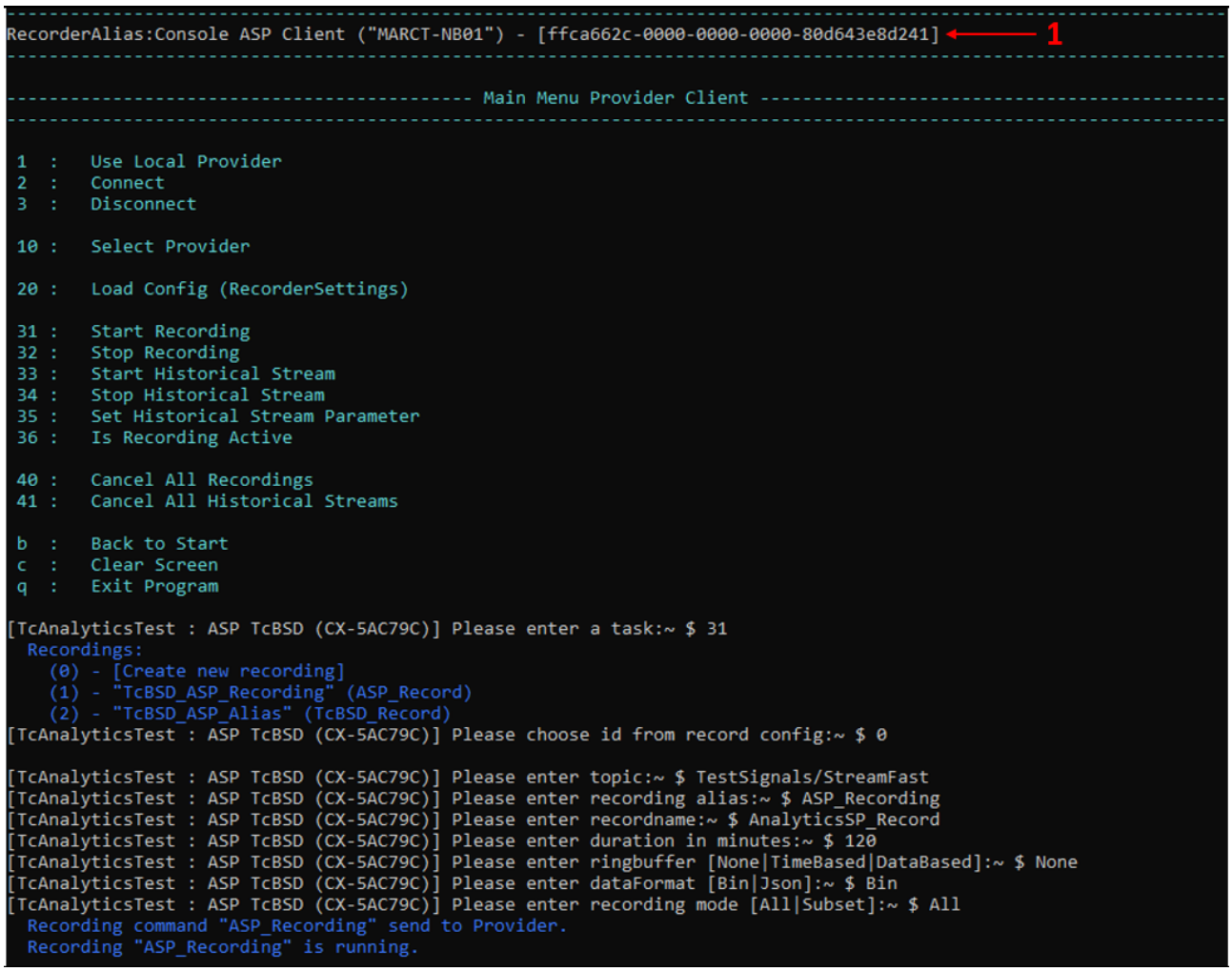
Use

To use the Analytics Storage Provider Client, a connection to an MQTT broker must first be established. Entering "2" starts a dialog in which the already configured MQTT brokers are presented for selection. There a new MQTT broker can be configured and connected by entering "0". Then, by entering "10" in the client main menu, an Analytics Storage Provider can be selected, which is available under the MQTT broker. Alternatively, by entering "1" in the client main menu, a connection to the MQTT broker and Analytics Storage Provider can be established directly from the local configuration file.

After the connection is successfully established, information about the connected MQTT broker and the selected Analytics Storage Provider is displayed in the prompter display before the prompt:

(MQTT broker: Analytics Storage Provider)

To start recordings, a dialog is started by entering "31". It is possible to start a recording that has already been created, provided that recordings have already been configured or read in via a recorder configuration file. In addition to the listed recordings, a new recording can also be configured and started by entering "0":



The configuration parameters correspond to the known parameters from the graphical Analytics Storage Provider Recorder. The default values can be deleted if necessary and replaced by individual entries. With the **recording mode** after the input "Subset" a subset of the symbolism can be defined by the recording data. Immediately after configuration, a command to start recording is sent to the connected Analytics Storage Provider. Running recordings can be stopped by entering "32". They are referenced by the MQTT topic from which the data comes and by the recording alias. If a recording is to be stopped by another client, the corresponding **Recorder Guid** must also be specified. The **Recorder Guid** is displayed together with the **Recorder Alias** above the input options in the client main menu (red 1).

Recording configurations created in the console client are not persisted. So after closing the client, the list of recording configurations is no longer available. Therefore, reading recorder configuration files (enter "20" in the client main menu) can be very helpful. The configuration file of a recorder is stored on Windows systems under the path *C:\Users*** *\AppData\Roaming\Beckhoff\TwinCAT Analytics Storage Provider* (replace *** with the corresponding user).

The historized data of the Analytics Storage Provider can be transmitted as a data stream to a definable result MQTT topic via the input "33". This also starts a dialog in which a previously configured data stream can be started. By entering "0", a new historical data stream can also be configured:

```
 1  :   Use Local Provider
 2  :   Connect
 3  :   Disconnect

 10 :   Select Provider

 20 :   Load Config (RecorderSettings)

 31 :   Start Recording
 32 :   Stop Recording
 33 :   Start Historical Stream
 34 :   Stop Historical Stream
 35 :   Set Historical Stream Parameter
 36 :   Is Recording Active

 40 :   Cancel All Recordings
 41 :   Cancel All Historical Streams

 b  :   Back to Start
 c  :   Clear Screen
 q  :   Exit Program

[TcAnalyticsTest : ASP TcBSD (CX-5AC79C)] Please enter a task:~ $ 33
  GetHistorical Cmds:
    (0) - [Create new GetHistorical cmd]
    (1) - "RecordID:29 | Topic:TestSignals/StreamFast"
    (2) - "RecordID:39 | Topic:TestSignals/StreamFast"
[TcAnalyticsTest : ASP TcBSD (CX-5AC79C)] Please choose id from cmd:~ $ 0

[TcAnalyticsTest : ASP TcBSD (CX-5AC79C)] Please enter systemID:~ $ cff7975b-b34d-43f7-755d-95cf135f50db
[TcAnalyticsTest : ASP TcBSD (CX-5AC79C)] Please enter topic:~ $ TestSignals/StreamFast
[TcAnalyticsTest : ASP TcBSD (CX-5AC79C)] Please enter symbol layout:~ $ 52a5066f-3c94-d853-f02b-bce62b4a6dea
[TcAnalyticsTest : ASP TcBSD (CX-5AC79C)] Please enter record id:~ $ 1
[TcAnalyticsTest : ASP TcBSD (CX-5AC79C)] Please enter start time in ns:~ $ 13331821883950000
[TcAnalyticsTest : ASP TcBSD (CX-5AC79C)] Please enter end time in ns:~ $ 133318220040054000
[TcAnalyticsTest : ASP TcBSD (CX-5AC79C)] Please enter maxSampleCnt default:~ $ 5000
[TcAnalyticsTest : ASP TcBSD (CX-5AC79C)] Please enter user sampletime default:~ $ -1
[TcAnalyticsTest : ASP TcBSD (CX-5AC79C)] Please enter outputFormat [Bin|Json]:~ $ Bin
[TcAnalyticsTest : ASP TcBSD (CX-5AC79C)] Please enter result topic:~ $ TcBSD_AnalyticsSP_ResultTopic
[TcAnalyticsTest : ASP TcBSD (CX-5AC79C)] Please enter symbol mode [All|Subset]:~ $ All
  GetHistoricalStream command for "TcBSD_AnalyticsSP_ResultTopic" send to Provider.
```

The parameters define a historized recording, whereby the parameter **result topic** defines the MQTT topic to which the data is to be streamed. After configuration, a command to start the historical stream is automatically sent to the Analytics Storage Provider.

By entering "35" in the client main menu, the parameters of an active historical stream can be adjusted. The historical stream is referenced by its result MQTT topic. The parameters can be used, for example, to adjust the speed or packet size of the data stream while it is running. Canceling a historical data stream is possible by entering "34" and specifying the result MQTT topic.

RuleEngine pipelines

It is also possible to start RuleEngine pipelines that have already been published. The entry "21" must be made for this. The appropriate RuleEngine pipeline can then be selected. RuleEngine pipelines can also be stopped (input "22"). It is also possible to restart individual rules (input "23").

```
[127.0.0.1 : TwinCAT Analytics StorageProvider ("INGMARR-NB02")] Please enter a task:~ $ 21
  RuleEngine Pipelines:
    (1) - "MyRuleEnginePipelineAlias" (d00c5366-4cf5-4d4e-a2f6-9dbe759e9dd2)
    (2) - "AutostartRuleEnginePipeline" (92bcfd4e-0a05-48a5-b914-80f680d48e92)
[127.0.0.1 : TwinCAT Analytics StorageProvider ("INGMARR-NB02")] Please choose id from pipeline configs:~ $ 1

  RuleEngine pipeline command send to Provider.
```

6.4.3 Batch files for control

The console client can be used to create batch files to control the Analytics Storage Provider. Some parameters are provided for this purpose:

`-Help / -H / -?`	Returns a description of all parameters

Parameters for the configuration settings:

-CreateASPConfig	Create a new Analytics Storage Provider settings XML
-MainTopic <mainTopic>	Analytics Storage Provider Main Topic
-Comment <comment>	Analytics Storage Provider comment
-EventLogTrace <True|False>	Trace to the event log
-DebugLog <True|False>	Additional DebugLog
-StorageType <type>	Storage type (**ANALYTICSFILE**, **AZURESQL**, **AZUREBLOB**)
-StorageConnString <connString>	Connection string or path to memory
-TlsType <Tls1.0|Tls1.1|Tls1.2>	Tls type (for AzureBlob)
-MaxDuration <duration (sec)>	Maximum duration of a TAY file
-MaxWriteLen <writeLen (bytes)>	Maximum length of a data packet

Configuration parameters:

-LocalProvider	Use the connection settings of the locally installed Analytics Storage Provider
-ConfigFile <path>	Use all configurations from the configuration file of an Analytics Storage Provider Recorder window
-ProviderGuid <guid>	Provider of the Analytics Storage Provider to be used
-ConfigCmdID <id>	ID number of the preconfigured recording in the configuration file
-ConfigCmdAlias <alias>	Alias of the preconfigured recording in the configuration file

Connection parameters:

-Broker /-Host <hostname>	Host name or IP address of the broker used
-Port <port>	Broker port (default value: 1883)
-User <username>	Username for the connection
-Password / -Pwd <password>	Password for the connection
-CA <path>	Path to the CA certificate for the connection
-Cert <path>	Path to the certificate for the connection
-Key_Cert <path>	Path to the key file for the connection
-Key_Pwd <password>	Password for the key file for the connection

Function parameters:

-StartRecord	Sends a StartRecord command
-StopRecord	Sends a StopRecord command
-IsRecordingActive	Checks whether a recording is currently running
-GetHistorical	Sends a GetHistoricalData command
-StopHistorical	Sends a StopHistoricalData command
-UpdateHistorical	Sends a HistoricalUpdate command
-CancelAllRec	Sends a Cancel command to all active recordings
-CancelAllHist	Sends a Cancel command to all active historical data streams
-StartPipeline	Sends a StartRuleEngine pipeline command
-StopPipeline	Sends a StopRuleEngine pipeline command
-RestartRule	Sends a RestartRule command
-DeleteRecordingsOlderThan	Deletes recordings whose end time is older than a specified timestamp. Optionally, the topic of the historical stream can also be specified. Only the active historical streams are taken into account.

Recording start/stop parameters:

`-Alias <alias>`	Alias name of the recording
`-RecName <record>`	Alias name of the data set
`-Topic <topic>`	Topic to be included
`-DataFormat <Bin|Json>`	Data format of the live data stream
`-Duration <seconds>`	Recording duration
`-Ringbuffer <None|TimeBased|DataBased>`	Ring buffer mode (default value: **Default**)
`-RinbufferPara <minutes/MB>`	Parameters for the ring buffer (in seconds or megabytes)
`-Mode <All|Subset>`	Mode of recording. Takes all symbols and a subset of the symbols.
`-Symbols / -Sym <Symbol1,Symbol2>`	List of symbol subset as comma-separated list.
`-RecorderGuid <guid>`	Guid of the Analytics Storage Provider Recorder window.
`-Storage <guid>`	Guid from storage where to write.
`-SubBroker <guid>`	Guid from the Sub Broker from which the data is to be recorded.

Historical data stream start/stop parameters:

`-SystemID <systemID guid>`	System ID of the recorded data set.
`-Topic <topic>`	Topic of the recorded data set.
`-Layout <layout guid>`	Layout of the recorded data set.
`-RecordID <id>`	ID of the data set to be streamed.
`-StartTime <time ns>`	Start time of the data set to be streamed in nanoseconds.
`-EndTime <time ns>`	End time of the data set to be streamed in nanoseconds.
`-MaxSamples <samples>`	Maximum number of samples (default value: 5000)
`-UsrSampleTime <ms>`	Sampling rate. (Default value: -1; sampling rate of the recording)
`-DataFormat <Bin|Json>`	Data format of the data stream.
`-ResultTopic <topic>`	Result MQTT topic to which the data will be streamed.
`-Mode <All|Subset>`	Streaming mode. Streams all or a subset of the symbols.
`-Symbols / -Sym <Symbol1,Symbol2>`	List of symbol subset as comma-separated list.

Historical data stream update parameters:

`-MaxSamples <samples>`	Maximum number of samples (default value: 5000)
`-UsrSampleTime <ms>`	Sampling rate. (Default value: -1; sampling rate of the recording)
`-MaxPackSize <samples>`	Maximum message size in kilobytes
`-SendDuration <ms>`	Waiting time between sending messages in milliseconds.
`-ResultTopic <topic>`	Result MQTT topic to which the data will be streamed.

RuleEngine pipeline parameter:

`- PipelineGuid <guid>`	Guid of the RuleEngine pipeline.
`- RuleID <id>`	ID of the rule within a RuleEngine pipeline.

Delete recordings Parameter:

- DateTimeOlderThan <datetime>	Timestamp in the format "yyyy-MM-dd hh:mm". Any recording with an end time older than this timestamp will be deleted.
- HistoricalStreamTopic <topic>	Topic of the historical stream (optional).

Command line samples:

Create configuration:

```
TwinCAT.Analytics.StorageProvider.Client
     -CreateASPConfig
          -MainTopic Beckhoff/ASPTest
          -Comment Analytics Storage Provider (Test)
          -EventLogTrace False
          -DebugLog False
          -StorageType ANALYTICSFILE
               -StorageConnString C:\TwinCAT\Functions\TF3520-Analytics-StorageProvider\Storage
               -MaxDuration 120
               -MaxWriteLen 2048
          -Broker 172.17.62.135
               -Port 1883
               -User tcanalytics
               -Pwd 123
```

Start recording with local Analytics Storage Provider:

```
TwinCAT.Analytics.StorageProvider.Client
     -localprovider
     -startrecord
          -alias cmdTest
          -recname cmdRec1
          -topic TestSignals/TestStream
          -dataformat Bin
          -Duration 30
          -mode Subset
          -Symbols Variables.fCosine,Variables.fSine
```

Start configuration file of a recording:

```
TwinCAT.Analytics.StorageProvider.Client
     -ConfigFile "C:
\Users\User\AppData\Roaming\Beckhoff\TwinCAT Analytics Storage Provider\TcAnalyticsStorageProvider_R
ecorder.xml"
     -ProviderGuid 76141a7f-e580-4281-99d8-1b8a75ca014d
     -startrecord
     -ConfigCmdAlias cmdTest
```

Check recording status

```
TwinCAT.Analytics.StorageProvider.Client
     -Broker 172.17.62.135
          -Port 1883
          -User tcanalytics
          -Pwd 123
     -ProviderGuid 76141a7f-e580-4281-99d8-1b8a75ca014d
     -IsRecordingActive
          -alias cmdTest
          -recorderGuid a8e171d2-712d-bd8e-da15-7eef28b71ad2
```

Stop all recordings:

```
TwinCAT.Analytics.StorageProvider.Client
     -Broker 172.17.62.135
          -Port 1883
          -User tcanalytics
          -Pwd 123
     -ProviderGuid 76141a7f-e580-4281-99d8-1b8a75ca014d
     -CancelAllRec
```

Start historical data stream:

```
TwinCAT.Analytics.StorageProvider.Client
     -localprovider
     -GetHistorical
          -systemID c29ac2d4-76ce-ff44-4d7f-355ffbcca6bf
          -layout 9a8e171d-712d-bd8e-da15-7eef28b71ad2
          -topic TestSignals/TestStream
          -recordID 1
          -startTime 132696863612730000
          -endTime 132696864177720000
          -maxSamples 5000
          -usrSampleTime -1
          -resultTopic _TestSignals/TestStream/123
          -dataformat Bin
          -mode Subset -symbols Variables.fSine
```

Start RuleEngine pipeline:

```
TwinCAT.Analytics.StorageProvider.Client
     -localprovider
     -StartPipeline
          -PipelineGuid d00c5366-4cf5-4d4e-a2f6-9dbe759e9dd2
```

Stop RuleEngine pipeline:

```
TwinCAT.Analytics.StorageProvider.Client
     -localprovider
     -StopPipeline
          -PipelineGuid d00c5366-4cf5-4d4e-a2f6-9dbe759e9dd2
```

Start a special rule of a RuleEngine pipeline:

```
TwinCAT.Analytics.StorageProvider.Client
     -localprovider
     -RestartRule
          -PipelineGuid d00c5366-4cf5-4d4e-a2f6-9dbe759e9dd2
          -RuleID 2
```

Delete old recordings:

```
TwinCAT.Analytics.StorageProvider.Client
     -localprovider
     -DeleteRecordingsOlderThan
          -DateTimeOlderThan yyyy-MM-dd 00:00
          - HistoricalStreamTopic Beckhoff /TcAnalyticsStorageProvider/41cfa2be-
ca72-4145-9e37-875851502aa6/Historical/Stream_65
```

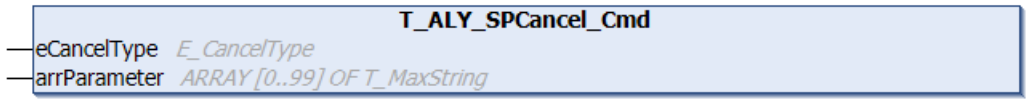
7 PLC API

7.1 Function blocks

7.1.1 Topic Architecture

7.1.1.1 Commands

7.1.1.1.1 T_ALY_SPCancel_Cmd



Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_SPCancel_Cmd EXTENDS T_ALY_JsonPayload
VAR_INPUT
    eCancelType : E_CancelType;
    arrParameter : ARRAY [0..99] OF T_MaxString;
END_VAR
```

Inheritence hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_SPCancel_Cmd

Inputs

Name	Type	Description
eCancelType	E_CancelType [▶ 141]	
arrParameter	ARRAY [0..99] OF T_MaxString	

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.1.2 T_ALY_SPGetHistorical_Cmd

Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_SPGetHistorical_Cmd EXTENDS T_ALY_JsonPayload
VAR_INPUT
    sTopic  : T_MaxString;
    sLayout : GUID;
    eMode   : E_SymbolMode := E_SymbolMode.All;
    eOutputFormat   : E_RawDataFormat := E_RawDataFormat.Bin;
    nMaxSampleCount : UDINT := 3000;
    nUserSampleTime : DINT := -1;
    nRecordID : DINT;
    nStartTimestamp : LINT;
    nEndTimestamp   : LINT;
    sResultTopic : T_MaxString;
    arrSymbol : ARRAY [0..255] OF T_ALY_Symbol;
END_VAR
```

Inheritence hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_SPGetHistorical_Cmd

Inputs

Name	Type	Description
sTopic	T_MaxString	Topic name of the recorded Live Stream
sLayout	GUID	Layout GUID of the recording
eMode	E_SymbolMode [▶ 145]	Get all symbols or only a subset
eOutputFormat	E_RawDataFormat [▶ 142]	Format of the returned data (actually only “Bin” supported)
nMaxSampleCount	UDINT	Max count of samples in one payload packet
nUserSampleTime	DINT	Sampletime in milliseconds of the returned stream. (-1 use the recorded sampletime)
nRecordID	DINT	Number of the record
nStartTimestamp	LINT	StartTime
nEndTimestamp	LINT	EndTime
sResultTopic	T_MaxString	Topicname of the result stream.
arrSymbol	ARRAY [0..255] OF T_ALY_Symbol [▶ 127]	If SymbolMode is Subset, only the list of this symbols will be returned

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.1.3 T_ALY_SPReadStreamRecords_Cmd

Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_SPReadStreamRecords_Cmd EXTENDS T_ALY_JsonPayload
VAR_INPUT
    sStreamTopic  : STRING(255);
    sStreamSystemID : GUID;
    sStreamLayout : GUID;
    nRecordStartIndex : DINT;
    nMaxRecordCount : DINT;
    sResultTopic : T_MaxString;
END_VAR
```

Inheritance hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_SPReadStreamRecords_Cmd

Inputs

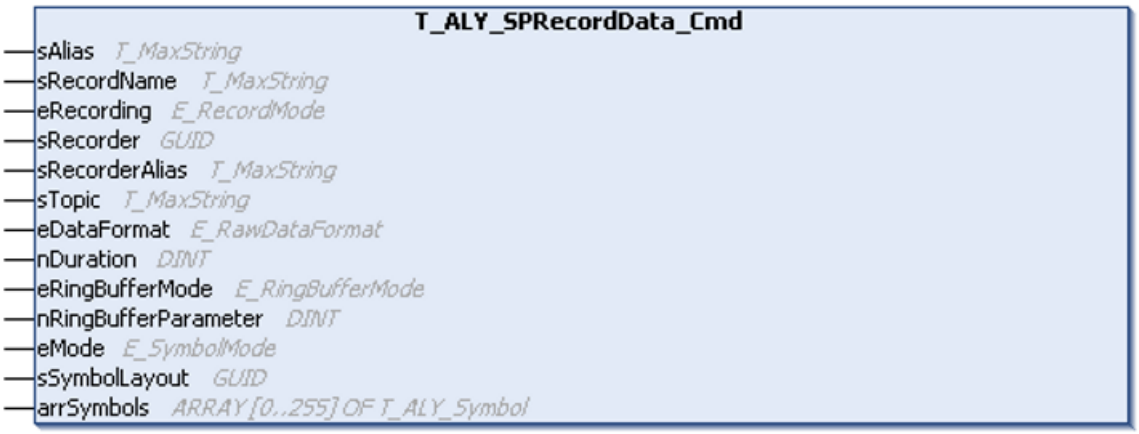
Name	Type	Description
sStreamTopic	STRING(255)	Topic name of the recorded live stream.
sStreamSystemID	GUID	SystemID of the target system from where the live stream was sent
sStreamLayout	GUID	Layout GUID of the recording
nRecordStartIndex	DINT	Start index of the first record to be read.
nMaxRecordCount	DINT	Total number of records to be read.
sResultTopic	T_MaxString	Topic name of the result stream

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.1.4 T_ALY_SPRecordData_Cmd



Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_SPRecordData_Cmd EXTENDS T_ALY_JsonPayload
VAR_INPUT
    sAlias  : T_MaxString;
    sRecordName : T_MaxString;
    eRecording : E_RecordMode;
    sRecorder : GUID;
    sRecorderAlias : T_MaxString;
    sTopic: T_MaxString;
    eDataFormat : E_RawDataFormat;
    nDuration : DINT;
    eRingBufferMode : E_RingBufferMode;
    nRingBufferParameter : DINT;
    eMode : E_SymbolMode;
    sSymbolLayout : GUID;
    arrSymbols : ARRAY [0..255] OF T_ALY_Symbol;
END_VAR
```

Inheritence hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_SPRecordData_Cmd

Inputs

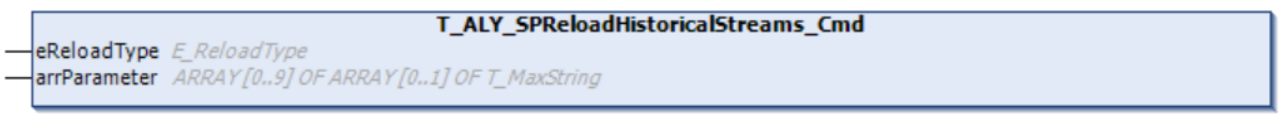
Name	Type	Description
sAlias	T_MaxString	Alias name for the Recording
sRecordName	T_MaxString	Name for this record
eRecording	E_RecordMode [▶ 143]	Start or Stop the recording
sRecorder	GUID	Individual GUID of the recorder
sRecorderAlias	T_MaxString	Alias name for the recorder
sTopic	T_MaxString	Topic name of the live stream
eDataFormat	E_RawDataFormat [▶ 142]	Store data format. (actually only Binary format is supported)
nDuration	DINT	Duration in minutes of the recording. (-1 unlimited)
eRingBufferMode	E_RingBufferMode [▶ 144]	Ringbuffer modus
nRingBufferParameter	DINT	TimeBased => Parameter in minutes DataBased => Parameter in Megabytes
eMode	E_SymbolMode [▶ 145]	Record all symbols or only a subset
sSymbolLayout	GUID	
arrSymbols	ARRAY [0..255] OF T_ALY_Symbol [▶ 127]	If SymbolMode is Subset, only the list of this symbols will be recorded

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.1.5 T_ALY_SPReloadHistoricalStreams_Cmd



Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_SPReloadHistoricalStreams_Cmd EXTENDS T_ALY_JsonPayload
VAR_INPUT
    eReloadType : E_ReloadType;
    arrParameter : ARRAY [0..9] OF ARRAY [0..1] OF T_MaxString;
END_VAR
```

Inheritance hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_SPReloadHistoricalStreams_Cmd

Inputs

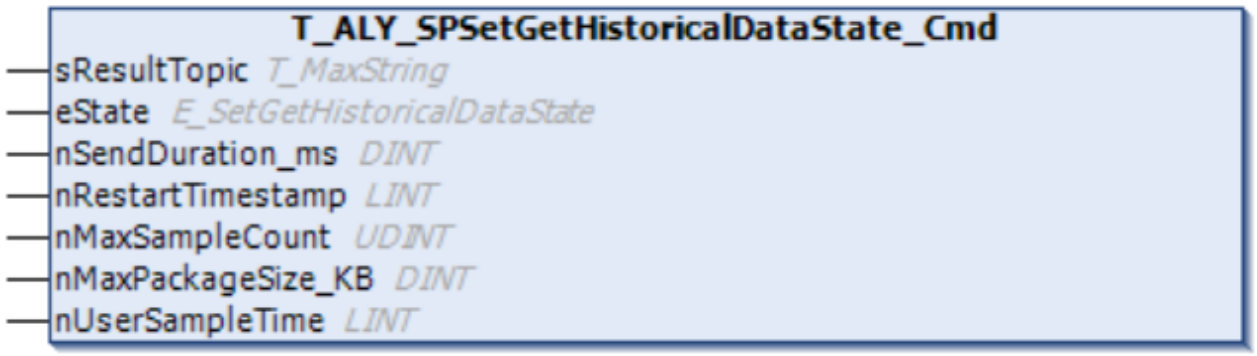
Name	Type	Description
eReloadType	E_ReloadType [▶ 144]	Update mode selection
arrParameter	ARRAY [0..9] OF ARRAY [0..1] OF T_MaxString	Additional parameters

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.1.6 T_ALY_SPSetGetHistoricalDataState_Cmd



Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_SPSetGetHistoricalDataState_Cmd EXTENDS T_ALY_JsonPayload
VAR_INPUT
    sResultTopic  : T_MaxString;
    eState : E_SetGetHistoricalDataState;
    nSendDuration_ms : DINT;
```

```
    nRestartTimestamp : LINT;
    nMaxSampleCount : UDINT;
    nMaxPackageSize_KB: DINT;
    nUserSampleTime : LINT;
END_VAR
```

Inheritance hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_SPSetGetHistoricalDataState_Cmd

Inputs

Name	Type	Description
sResultTopic	T_MaxString	Topic name of the result stream (used like a handle).
eSta.te	E_SetGetHistoricalDataState [▶ 144]	Historical stream state
nSendDuration_ms	DINT	Waiting time between sending the individual packages
nRestartTimestamp	LINT	Timestamp at which the result stream is continued.
nMaxSampleCount	UDINT	Maximum number of entries in a package
nMaxPackageSize_KB	DINT	Maximum size of a package
nUserSampleTime	LINT	Sample time in milliseconds of the returned stream (-1 uses the recorded sample time).

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.1.7 T_ALY_SPStorageCtrl_Cmd

Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_SPStorageCtrl_Cmd EXTENDS T_ALY_JsonPayload
VAR_INPUT
    eCtrlMode : E_ControlMode;
    sStorageGuid : GUID;
END_VAR
```

Inheritence hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_SPRecordData_Cmd

Inputs

Name	Type	Description
eCtrlMode	E_ControlMode [▶ 142]	Start, stop, etc. of the Storage.
sStorageGuid	GUID	Individual GUID of the Storage.

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.1.8 T_ALY_SPRuleEnginePipeline_Cmd

Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_SPRecordData_Cmd EXTENDS T_ALY_JsonPayload
VAR_INPUT
    sRuleEnginePipeline : GUID;
    eCmdType : E_PipelineCmdType;
    sRecorder : GUID;
    sRecorderAlias : T_MaxString;
END_VAR
```

Inheritance hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_SPRuleEnginePipeline_Cmd

Inputs

Name	Type	Description
sRuleEnginePipeline	GUID	Individual GUID of the RuleEngine pipeline.
eCmdType	E_PipelineCmdType [▶ 146]	Start or Stop the recording.
sRecorder	GUID	Individual GUID of the recorder.
sRecorderAlias	T_MaxString	Alias name for the recorder.

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.2 Descriptions

7.1.1.2.1 T_ALY_HistoricalStream_Desc

Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_HistoricalStream_Desc EXTENDS T_ALY_JsonPayload
VAR_OUTPUT
    sSource  : STRING(255);
    sStreamTopic : STRING(255);
    sStreamAlias : STRING(255);
    sStreamSystemID : GUID;
    sLayout : GUID;
    nCycleTime: UDINT;
    nDataSize : UDINT;
    arrRecords : ARRAY [0..cMaxRecordCount] OF T_RecordTimestamps;
END_VAR
```

Inheritance hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_HistoricalStream_Desc

Outputs

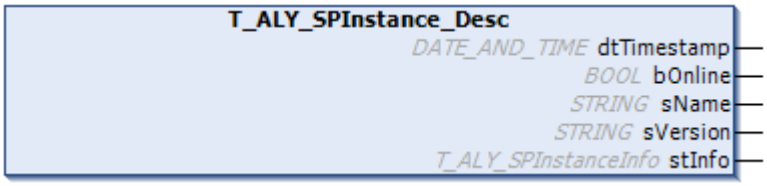
Name	Type	Description
sSource	STRING(255)	Data source name
sStreamTopic	STRING(255)	Topic name of the recorded stream
sStreamAlias	STRING(255)	Alias name of the stream
sStreamSystemID	GUID	SystemID GUID of the stream
sLayout	GUID	Layout GUID of the recording
nCycleTime	UDINT	Cycle time of the recording
nDataSize	UDINT	Data size of an entry of the recording
arrRecords	ARRAY [0..cMaxRecordCount] OF T_RecordTimestamps [▶ 128]	Timestamp of the various recordings

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.2.2 T_ALY_SPInstance_Desc



Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_SPInstance_Desc EXTENDS T_ALY_JsonPayload
VAR_OUTPUT
    dtTimestamp : DATE_AND_TIME;
    bOnline     : BOOL;
    sName       : STRING;
    sVersion    : STRING;
    stInfo      : T_ALY_SPInstanceInfo;
END_VAR
```

Inheritance hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_SPInstance_Desc

Outputs

Name	Type	Description
dtTimestamp	DATE_AND_TIME	Start time of the Storage Provider Service
bOnline	BOOL	Indicates whether the service is online.
sName	STRING	IoT Device name "TwinCAT Analytics Storage Provider"
sVersion	STRING	Version of the Storage Provider
stInfo	T_ALY_SPInstanceInfo [▶ 124]	Detailed description of the Storage Provider

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.2.3 T_ALY_SPRecordData_Desc

Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_SPRecordData_Desc EXTENDS T_ALY_JsonPayload
VAR_OUTPUT
    sRecordDataGuid    : GUID;
    nStartTimestamp    : LINT;
    eStatus            : E_RecordingState;
    nRecordID          : LINT;
    stRecord           : T_ALY_SPRecordData_Cmd;
END_VAR
```

Inheritance hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_SPRecordData_Desc

Outputs

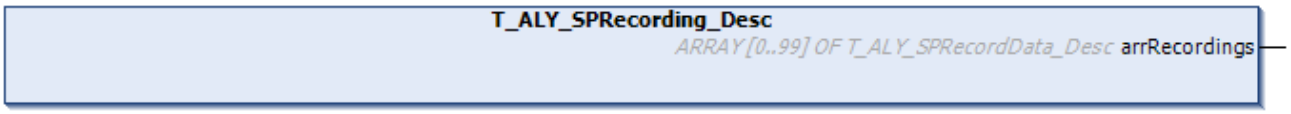
Name	Type	Description
sRecordDataGuid	GUID	GUID "Key" to identify the recording
nStartTimestamp	LINT	Start time of the recording.
eStatus	E_RecordingState [▶ 143]	Recording state
nRecordID	LINT	Recording ID
stRecord	T_ALY_SPRecordData_Cmd [▶ 113]	Associated recording command

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.2.4 T_ALY_SPRecording_Desc



Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_SPRecording_Desc EXTENDS T_ALY_JsonPayload
VAR_OUTPUT
    aRecordings : ARRAY [0..99] OF T_ALY_SPRecordData_Desc;
END_VAR
```

Inheritance hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_SPRecording_Desc

Outputs

Name	Type	Description
aRecordings	ARRAY [0..99] OF T_ALY_SPRecordData_Desc [▶ 120]	List of all current recordings

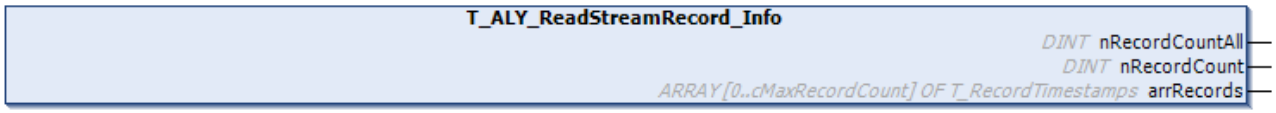
Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.3 Info

7.1.1.3.1 T_ALY_ReadStreamRecord_Info



Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_HistoricalStream_Desc EXTENDS T_ALY_JsonPayload
VAR_OUTPUT
    nRecordCountAll: UDINT;
    nRecordCount : UDINT;
    arrRecords : ARRAY [0..cMaxRecordCount] OF T_RecordTimestamps;
END_VAR
```

Inheritance hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_ReadStreamRecord_Info

Outputs

Name	Type	Description
nRecordCountAll	UDINT	Number of all existing records
nRecordCount	UDINT	Number of records read out
arrRecords	ARRAY [0..cMaxRecordCount] OF T_RecordTimestamps [▶ 128]	Timestamp of the records read out

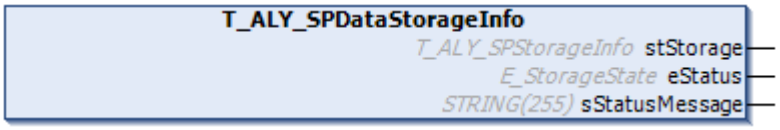
Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.4 SubTypes

7.1.1.4.1 T_ALY_SPDataStorageInfo



Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_SPDataStorageInfo EXTENDS T_ALY_JsonPayload
VAR_OUTPUT
    stStorage        : T_ALY_SPStorageInfo;
    eStatus          : E_StorageState;
    sStatusMessage   : STRING(255);
END_VAR
```

Inheritance hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_SPDataStorageInfo

Outputs

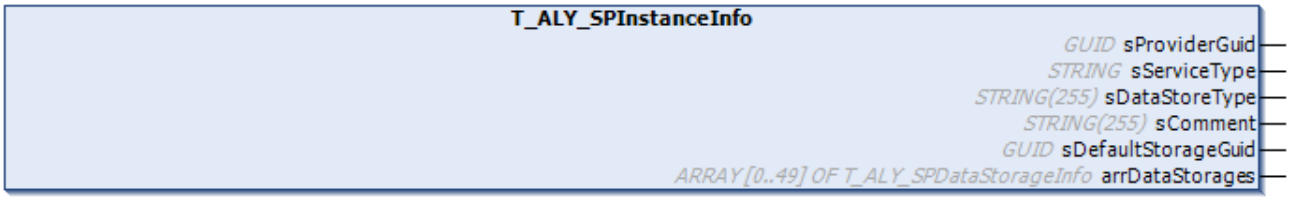
Name	Type	Description
stStorage	T_ALY_SPStorageInfo [▶ 125]	Detailed Storage information
eStatus	E_StorageState [▶ 145]	Status of the Storage
sStatusMessage	STRING(255)	Storage status message

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.4.2 T_ALY_SPInstanceInfo



Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_SPInstanceInfo EXTENDS T_ALY_JsonPayload
VAR_OUTPUT
    sProviderGuid       : GUID;
    sServiceType        : STRING;
    sDataStoreType      : STRING(255);
    sComment            : STRING(255);
    sDefaultStorageGuid : GUID;
    arrDataStorages     : ARRAY [0..49] OF T_ALY_SPDataStorageInfo;
END_VAR
```

Inheritance hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_SPInstanceInfo

Outputs

Name	Type	Description
sProviderGuid	GUID	Individual GUID of a Storage Provider instance
sServiceType	STRING	Service type
sDataStoreType	STRING(255)	Storage type
sComment	STRING(255)	Comment on the Storage Provider instance
sDefaultStorageGuid	GUID	Storage GUID of the Standard Storage
arrDataStorages	ARRAY [0..49] OF T_ALY_SPDataStorageInfo [▶ 123]	List of configured Storages

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.4.3 T_ALY_SPStorageInfo

Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_SPStorageInfo EXTENDS T_ALY_JsonPayload
VAR_OUTPUT
    sStorageGuid       : GUID;
    sStorageName        : STRING;
    eDataStorageType      : E_DataStorageType;
    sComment            : STRING(255);
END_VAR
```

Inheritance hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_SPStorageInfo

Outputs

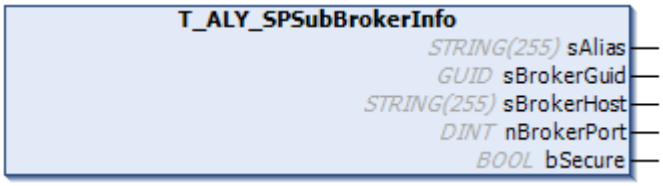
Name	Type	Description
sStorageGuid	GUID	Individual GUID of a Storage
sStorageName	STRING	Name of the Storage
eDataStorageType	E_DataStorageType [▶ 142]	Storage type
sComment	STRING(255)	Comment on Storage

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.4.4 T_ALY_SPSubBrokerInfo



Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_SPSubBrokerInfo EXTENDS T_ALY_JsonPayload
VAR_OUTPUT
    sAlias          : STRING(255);
    sBrokerGuid     : GUID;
    sBrokerHost     : STRING(255);
    nBrokerPort     : DINT;
    bSecure         : BOOL;
END_VAR
```

Inheritance hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_SPSubBrokerInfo

Outputs

Name	Type	Description
sAlias	STRING(255)	Alias name of the broker configuration
sBrokerGuid	GUID	Individual GUID of the broker configuration
sBrokerHost	STRING(255)	Broker Host Name
nBrokerPort	DINT	Broker port
bSecure	BOOL	TRUE if communication is established via certificates.

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.4.5 T_ALY_Symbol

Syntax

Definition:

```
FUNCTION_BLOCK T_ALY_Symbol EXTENDS T_ALY_JsonPayload
VAR_INPUT
    sName  : T_MaxString;
    sBaseType  : T_MaxString;
    nBitOffset : UDINT;
    nBitSize   : UDINT;
END_VAR
```

Inheritence hierarchy

T_ALY_JsonPayload [▶ 129]

T_ALY_Symbol

Inputs

Name	Type	Description
sName	T_MaxString	Name of the symbol
sBaseType	T_MaxString	DataType of the symbol
nBitOffset	UDINT	BitOffset of the symbol
nBitSize	UDINT	BitSize of the symbol

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.4.6 T_RecordTimestamps

Syntax

Definition:

```
FUNCTION_BLOCK T_RecordTimestamps EXTENDS T_ALY_JsonPayload
VAR_OUTPUT
    nRecordID  : DINT;
    sAlias   : STRING(255);
    nStartTimestamp : LINT;
    nEndTimestamp   : LINT;
END_VAR
```

Inheritance hierarchy

T_ALY_JsonPayload [▶ 129]

T_RecordTimestamps

Outputs

Name	Type	Description
nRecordID	DINT	Recording number
sAlias	STRING(255)	Alias name of the recording
nStartTimestamp	LINT	Start timestamp of the recording
nEndTimestamp	LINT	End timestamp of the recording

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.1.5 Base Types

7.1.1.5.1 T_ALY_JsonPayload

Syntax

Definition:

```
FUNCTION_BLOCK INTERNAL T_ALY_JsonPayload
```

Methods

Name	Definition location	Description
Reset		Reset all values in the payload FB.
Init_JsonValue	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON object.
Init_String	Inherited from T_ALY_JsonPayload [▶ 129]	Initialization of the FB with JSON string.
GetJsonLength	Inherited from T_ALY_JsonPayload [▶ 129]	Get the length of the JSON payload.
GetJsonString	Inherited from T_ALY_JsonPayload [▶ 129]	Get JSON payload as string.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.2 FB_ALY_StorageProvider

The FB_ALY_StorageProvider is a client FB for communication with a Storage Provider instance. The FB provides methods to trigger historical data or start/stop recordings.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALY_StorageProvider
VAR_INPUT
    stConfig : ST_ALY_SP_Config;
END_VAR
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    ipResultMessage  : I_TcMessage;
    eConnectionState : ETcIotMqttClientState;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALY_SP_Config [▶ 140]	Structure for the configuration of the FB.

Outputs

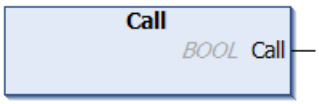
Name	Type	Description
bBusy	BOOL	TRUE as soon as a method of the function block is active.
bError	BOOL	Becomes TRUE when an error situation occurs.
ipResultMessage	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.
eConnectionState	ETcIotMqttClientState	Specifies the state of the connection between client and broker as an enumeration ETcIotMqttClientState.

Methods

Name	Definition location	Description
Call [▶ 131]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
Cancel [▶ 131]	Local	Method for aborting activities of the TwinCAT Analytics Storage Provider.
GetHistoricalData [▶ 132]	Local	Method for requesting historical data.
GetInstanceInfo [▶ 132]	Local	Method for receiving the instance information of the Storage Provider.
GetRecordingInfoByAlias [▶ 133]	Local	Method for receiving recording information.
GetRecordingInfoByKey [▶ 133]	Local	Method for receiving recording information.
ReadHistoricalStreams [▶ 134]	Local	Method for reading all historical streams.
ReadStreamRecords [▶ 134]	Local	Method for reading all records of a historical stream.
ReadSubBroker [▶ 135]	Local	Method for reading all declared message brokers.
ResetCommunication [▶ 136]	Local	Method to reset the MQTT connection to the broker.
RestartPipelineRule [▶ 136]	Loca	Restarts a rule of a pipeline.
SendCommand [▶ 136]	Local	Generic method for sending various commands.
SetHistoricalDataState [▶ 137]	Local	Method for setting various parameters of a historical stream.
StartPipeline [▶ 137]	Local	Starts recording a live MQTT binary stream.
StartRecord [▶ 138]	Local	Starts recording a live MQTT binary stream.
StartRecordEx [▶ 138]	Local	Starts recording a live MQTT binary stream.
StopPipeline [▶ 139]	Local	Stops the selected recording.
StopRecord [▶ 139]	Local	Stops the selected recording.
StorageControlling [▶ 140]	Local	Method for controlling the declared storages.

Development Environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.25	PC or CX (x64, x86, Arm®)	Tc3_AnalyticsStorageProvider

7.1.2.1 Call



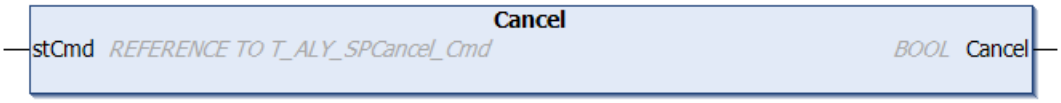
Syntax

```
METHOD Call : BOOL
```

Return value

Name	Type	Description
Call	BOOL	

7.1.2.2 Cancel



Syntax

```
METHOD Cancel : BOOL
VAR_INPUT
    stCmd : REFERENCE TO T_ALY_SPCancel_Cmd;
END_VAR
```

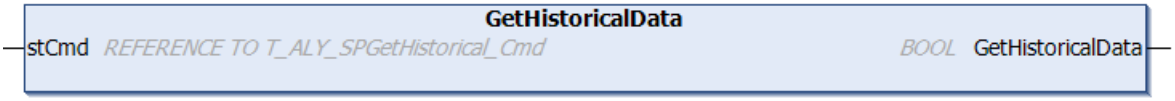
Inputs

Name	Type	Description
stCmd	REFERENCE TO T_ALY_SPCancel_Cmd [▶ 109]	JSON command to cancel operations of the TwinCAT Analytics Storage Provider.

Return value

Name	Type	Description
Cancel	BOOL	Is TRUE if done

7.1.2.3 GetHistoricalData



Syntax

```
METHOD GetHistoricalData : BOOL
VAR_INPUT
    stCmd : REFERENCE TO T_ALY_SPHistorical_Cmd;
END_VAR
```

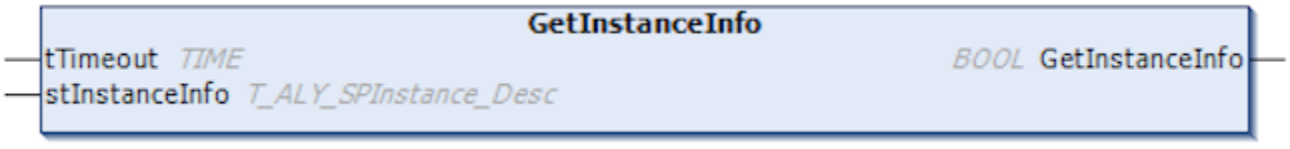
Inputs

Name	Type	Description
stCmd	REFERENCE TO T_ALY_SPGetHistorical_Cmd [▶ 110]	JSON command to get historical data from TwinCAT Analytics Storage Provider.

Return value

Name	Type	Description
GetHistoricalData	BOOL	Is TRUE if done

7.1.2.4 GetInstanceInfo



Syntax

```
METHOD GetInstanceInfo : BOOL
VAR_INPUT
    tTimeout : TIME;
    stInstanceInfo : T_ALY_SPInstance_Desc;
END_VAR
```

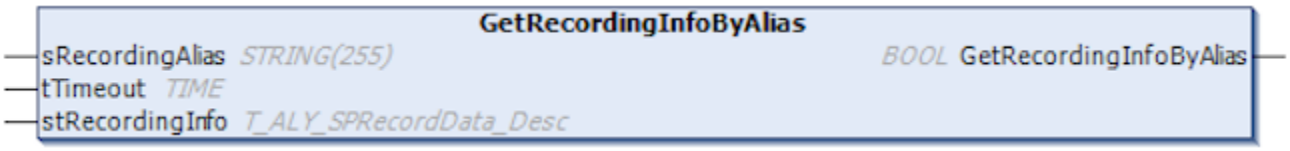
Inputs

Name	Type	Description
tTimeout	TIME	Duration until the procedure is aborted.
stInstanceInfo	T_ALY_SPInstance_Desc [▶ 119]	JSON description of the Storage Provider instance.

Return value

Name	Type	Description
GetInstanceInfo	BOOL	Is TRUE when completed

7.1.2.5 GetRecordingInfoByAlias



Syntax

```
METHOD GetRecordingInfoByAlias : BOOL
VAR_INPUT
    sRecordingAlias : STRING(255);
    tTimeout        : TIME;
    stRecordingInfo : T_ALY_SPRecordData_Desc;
END_VAR
```

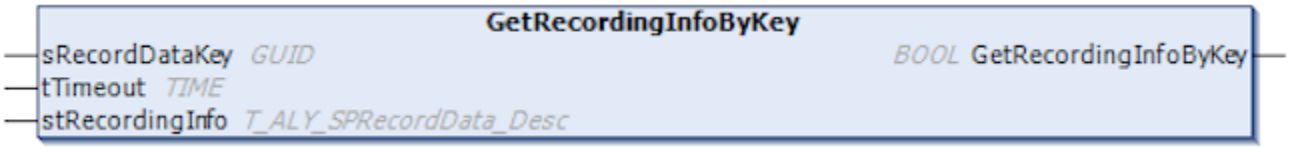
Inputs

Name	Type	Description
sRecordingAlias	STRING(255)	Search criterion "Alias"
tTimeout	TIME	Duration until the procedure is aborted.
stRecordingInfo	T_ALY_SPRecordData_Desc [▶ 120]	JSON description of the recording.

Return value

Name	Type	Description
GetRecordingInfoByAlias	BOOL	Is TRUE when completed

7.1.2.6 GetRecordingInfoByKey



Syntax

```
METHOD GetRecordingInfoByKey : BOOL
VAR_INPUT
    sRecordDataKey  : GUID;
    tTimeout        : TIME;
    stRecordingInfo : T_ALY_SPRecordData_Desc;
END_VAR
```

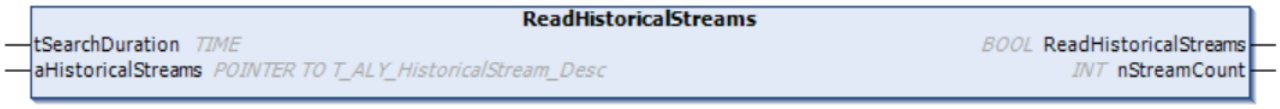
Inputs

Name	Type	Description
sRecordDataKey	GUID	Search criterion "RecordDataKey"
tTimeout	TIME	Duration until the procedure is aborted.
stRecordingInfo	T_ALY_SPRecordData_Desc [▶ 120]	JSON description of the recording.

Return value

Name	Type	Description
GetRecordingInfoByKey	BOOL	Is TRUE when completed

7.1.2.7 ReadHistoricalStreams



Syntax

```
METHOD GetHistoricalData : BOOL
VAR_INPUT
tSearchDuration : TIME := TIME#5s0ms
    aHistoricalStreams : POINTER TO T_ALY_HistoricalStream_Desc;
nStreamCount : INT
END_VAR
```

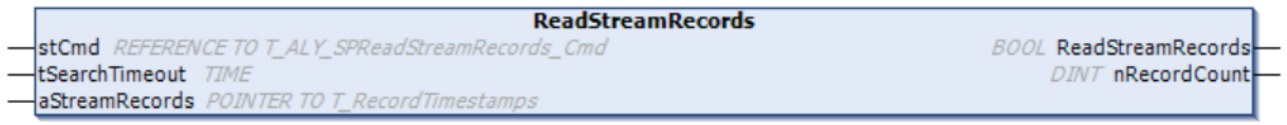
Inputs

Name	Type	Description
tSearchDuration	TIME	Time period in which to wait for feedback.
aHistoricalStreams	POINTER TO T_ALY_HistoricalStream_Desc [▶ 118]	Description of the different historical streams

Return value

Name	Type	Description
ReadHistoricalStreams	BOOL	Is TRUE when completed
nStreamCount	INT	Number of streams read out

7.1.2.8 ReadStreamRecords



Syntax

```
METHOD ReadStreamRecords : BOOL
VAR_INPUT
    stCmd : REFERENCE TO T_ALY_SPReadStreamRecords_Cmd;
tSearchTimeout : TIME := TIME#5s0ms;
```

```
aStreamRecords : POINTER TO T_RecordTimestamps;
nRecordCount : DINT;
END_VAR
```

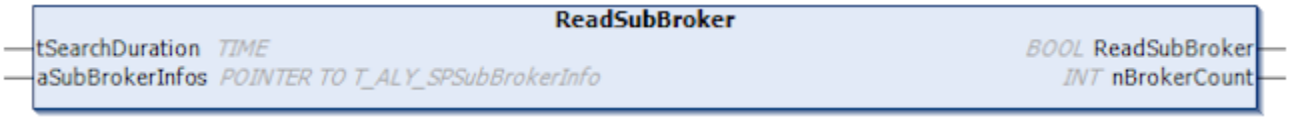
Inputs

Name	Type	Description
stCmd	REFERENCE TO T_ALY_SPReadStreamRecords_Cmd [▶ 111]	JSON command to get recordings of a historical stream from TwinCAT Analytics Storage Provider.
tSearchTimeout	TIME	Waiting time for the response.
aStreamRecords	POINTER TO T_RecordTimestamps [▶ 128]	Recordings read out

Return value

Name	Type	Description
ReadStreamRecords	BOOL	Is TRUE when completed
nRecordCount	DINT	Number of records read out

7.1.2.9 ReadSubBroker



Syntax

```
METHOD ReadSubBroker : BOOL
VAR_INPUT
    tSearchDuration    : TIME;
    aSubBrokerInfos    : POINTER TO T_ALY_SPSubBrokerInfo;
    nBrokerCount       : INT;
END_VAR
```

Inputs

Name	Type	Description
tSearchDuration	TIME	Duration until the search is completed
aSubBrokerInfo	POINTER TO T_ALY_SPSubBrokerInfo [▶ 126]	Address to an array in which the broker information found is stored.
nBrokerCount	INT	Number of brokers found

Return value

Name	Type	Description
ReadSubBroker	BOOL	Is TRUE when completed

7.1.2.10 ResetCommunication

Syntax

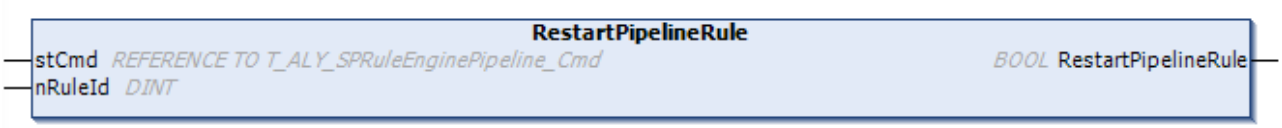
```
METHOD ResetCommunication : BOOL
VAR_INPUT

END_VAR
```

Return value

Name	Type	Description
ResetCommunication	BOOL	Is TRUE when completed

7.1.2.11 RestartPipelineRule



Syntax

```
METHOD RestartPipelineRule : BOOL
VAR_INPUT
    stCmd : REFERENCE TO T_ALY_SPRuleEnginePipeline_Cmd;
    nRuleId : DINT;
END_VAR
```

Inputs

Name	Type	Description
stCmd	REFERENCE TO T_ALY_SPRuleEnginePipeline_Cmd [▶ 117]	JSON command to start the recording a live stream.
nRuleId	DINT	ID of the rule to be restarted.

Return value

Name	Type	Description
RestartPipelineRule	BOOL	Is TRUE when completed.

7.1.2.12 SendCommand

Syntax

```
METHOD SendCommand : BOOL
VAR_INPUT
    stCmd : I_ALY_SPCommand;
END_VAR
```

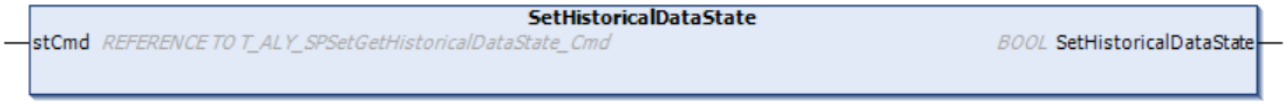
Inputs

Name	Type	Description
stCmd	I_ALY_SPCommand	JSON command to interact with the TwinCAT Analytics Storage Provider.

Return value

Name	Type	Description
SendCommand	BOOL	Is TRUE when completed

7.1.2.13 SetHistoricalDataState



Syntax

```
METHOD SetHistoricalDataState : BOOL
VAR_INPUT
    stCmd : REFERENCE TO T_ALY_SPGetSetHistoricalDataState_Cmd;
END_VAR
```

Inputs

Name	Type	Description
stCmd	REFERENCE TO T_ALY_SPGetSetHistoricalDataState_Cmd [▶ 115]	JSON command to set parameters of a started historical stream from the TwinCAT Analytics Storage Provider.

Return value

Name	Type	Description
SetHistoricalDataState	BOOL	Is TRUE when completed

7.1.2.14 StartPipeline

Syntax

```
METHOD StartPipeline : BOOL
VAR_INPUT
    stCmd : REFERENCE TO T_ALY_SPRuleEnginePipeline_Cmd;
END_VAR
```

Inputs

Name	Type	Description
stCmd	REFERENCE TO T_ALY_SPRuleEnginePipeline_Cmd [▶ 117]	JSON command to start the recording a live stream.

Return value

Name	Type	Description
StartPipeline	BOOL	Is TRUE when completed.

7.1.2.15 StartRecord

Syntax

```
METHOD StartRecord : BOOL
VAR_INPUT
    stCmd : REFERENCE TO T_ALY_SPRecordData_Cmd;
END_VAR
```

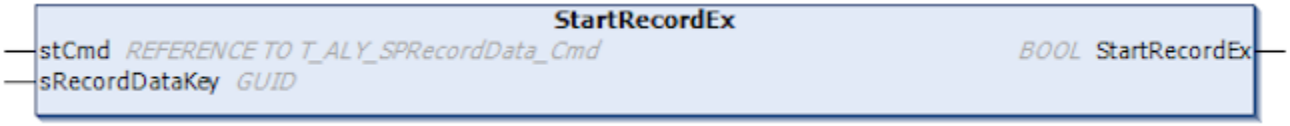
Inputs

Name	Type	Description
stCmd	REFERENCE TO T_ALY_SPRecordData_Cmd [▶ 113]	JSON command to start the recording a live stream.

Return value

Name	Type	Description
StartRecord	BOOL	Is TRUE if done

7.1.2.16 StartRecordEx



In contrast to the StartRecord [▶ 138] method, a RecordDataKey can be specified here. This key makes it easier to find the recording you have started in order to check the status of the recording. The GetRecordingInfoByKey [▶ 133] method can be used to retrieve the recording information.

Syntax

```
METHOD StartRecordEx : BOOL
VAR_INPUT
    stCmd : REFERENCE TO T_ALY_SPRecordData_Cmd;
    sRecordDataKey : GUID;
END_VAR
```

Inputs

Name	Type	Description
stCmd	REFERENCE TO T_ALY_SPRecordData_Cmd [▶ 113]	JSON command to start the recording a live stream.
sRecordDataKey	GUID	Guid "Key" to identify the recording that has been started.

Return value

Name	Type	Description
StartRecordEx	BOOL	Is TRUE when completed

7.1.2.17 StopPipeline

Syntax

```
METHOD StopPipeline : BOOL
VAR_INPUT
    stCmd : REFERENCE TO T_ALY_SPRuleEnginePipeline_Cmd;
END_VAR
```

Inputs

Name	Type	Description
stCmd	REFERENCE TO T_ALY_SPRuleEnginePipeline_Cmd [▶ 117]	JSON command to stop recording of a live stream.

Return value

Name	Type	Description
StopPipeline	BOOL	Is TRUE when completed.

7.1.2.18 StopRecord

Syntax

```
METHOD StopRecord : BOOL
VAR_INPUT
    stCmd : REFERENCE TO T_ALY_SPRecordData_Cmd;
END_VAR
```

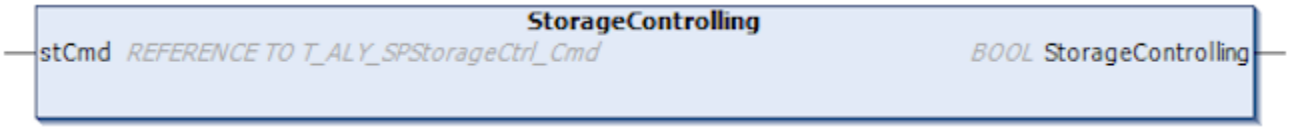
Inputs

Name	Type	Description
stCmd	REFERENCE TO T_ALY_SPRecordData_Cmd [▶ 113]	JSON command to stop recording of a live stream.

Return value

Name	Type	Description
StopRecord	BOOL	Is TRUE when completed.

7.1.2.19 StorageControlling



Syntax

```
METHOD StorageControlling : BOOL
VAR_INPUT
    stCmd : REFERENCE TO T_ALY_SPStorageCtrl_Cmd;
END_VAR
```

Inputs

Name	Type	Description
stCmd	REFERENCE TO T_ALY_SPStorageCtrl_Cmd [▶ 116]	JSON command to control the storages.

Return value

Name	Type	Description
StorageControlling	BOOL	Is TRUE when completed

7.2 Data types

7.2.1 ST_ALY_SP_Config

Syntax

Definition:

```
TYPE ST_Msg :
STRUCT
    sMainTopic     : T_MaxString;
    sProviderGuid  : GUID;
    stConnSettings : ST_ConnectionSettings
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Description
sMainTopic	T_MaxString	The main topic where the TwinCAT Analytics Storage Provider is located on the message broker
sProviderGuid	GUID	The individual GUID of the TwinCAT Analytics Storage Provider Instance
stConnSettings	ST_ConnectionSettings [▶ 141]	MQTT connection settings to connect with the message broker

7.2.2 ST_ConnectionSettings

Syntax

Definition:

```
TYPE ST_ConnectionSettings :
STRUCT
    sHostName   : T_MaxString;
    nHostPort   : UINT := 1883;
    sUserId     : T_MaxString;
    sPassword   : T_MaxString;
    bWithCertificate : BOOL := BOOL;
    sCA     : T_MaxString;
    sCert   : T_MaxString;
    sKey    : T_MaxString;
sKeyPwd : T_MaxString;
END_STRUCT
END_TYPE
```

Parameter

Name	Type	Descriptiom
sHostName	T_MaxString	sHostName can be specified as name or as IP address. If no information is provided, the local host is used.
nHostPort	UINT	The host port can be specified here. The default is 1883.
sUserId	T_MaxString	Optionally, a user name can be specified.
sPassword	T_MaxString	A password for the user name can be entered here.
bWithCertificate	BOOL	If TRUE the certificates will be used for communication
sCA	T_MaxString	Certificate of the certificate authority (CA)
sCert	T_MaxString	Client certificate to be used for authentication at the broker
sKey	T_MaxString	Private key of the client
sKeyPwd	T_MaxString	Password of the private key, if applicable

7.2.3 E_CancelType

Syntax

Definition:

```
TYPE E_CancelType :
(
    HistoricalData := 0,
    AllRecordData
)INT;
END_TYPE
```

Parameter

Name	Descriptiom
HistoricalData	Canceled the selected historical data stream
AllRecordData	Canceled all running recordings

7.2.4 E_ControlMode

Syntax

Definition:

```
TYPE E_ControlMode :
(
    Start := 0,
    Stop
    DeleteSettings
)INT;
END_TYPE
```

Parameters

Name	Description
Start	Starting the "Storage" is triggered.
Stop	Stopping the "Storage" is triggered.
DeleteSettings	"Storage" configuration should be deleted (only works if the "Storage" is not online).

7.2.5 E_DataStorageType

Syntax

Definition:

```
TYPE E_DataStorageType :
(
    Empty := 0,
    AnalyticsFile
    AzureBlob
    MsSQL_Binary
    InfluxDB
    MsSQL_Plain
    CSVFile
)INT;
END_TYPE
```

Parameters

Name	Description
Empty	Unknown "Storage" type
AnalytticsFile	Analytics File (TwinCAT Analytics own data format)
AzureBlob	Microsoft Azure Blob
MsSQL_Binary	Microsoft SQL Server (data in binary format)
InfluxDB	Influx 2.x database
MsSQL_Plain	Microsoft SQL Server (data in plain text)
CSVFile	CSV file

7.2.6 E_RawDataFormat

Syntax

Definition:

```
TYPE E_RawDataFormat :
(
    Bin := 0,
    Json
)INT;
END_TYPE
```

Parameter

Name	Description
Bin	Analytics Binary Stream Format
Json	TwinCAT Json Format (actually not supported)

7.2.7 E_RecordingState

Syntax

Definition:

```
TYPE E_RecordingState :
(
    Not_Initialized := 0,
    Initializing
    RecordingCanceled
    Running
    Running_QueueHyst
    Stopping_ReceivingDataStopped
    RecordingDone
    WaitingForData
    Error
)INT;
END_TYPE
```

Parameters

Name	Description
Not_Initialized	Recording triggered. Waits for input data description for initialization.
Initialized	Recording successfully initialized.
RecordingCanceled	Recording canceled.
Running	Recording running. Input data is saved.
Running_QueueHyst	Recording running. Input data cannot be saved fast enough. Data loss!
Stopping_ReceivingDataStopped	Recording is stopped.
RecordingDone	Recording is done.
WaitingForData	Recording running. No data arrives from the Analytics Logger.
Error	An error has occurred during recording.

7.2.8 E_RecordMode

Syntax

Definition:

```
TYPE E_RecordMode :
(
    Start := 0,
    Stop
)INT;
END_TYPE
```

Parameter

Name	Description
Start	Starts the recording of the configured record
Stop	Stops the recording

7.2.9 E_ReloadType

Syntax

Definition:

```
TYPE E_ReloadType :
(
    All := 0,
    Specific
)INT;
END_TYPE
```

Parameters

Name	Description
All	All records are read in again.
Specific	Only one specific record will be reread.

7.2.10 E_RingBufferMode

Syntax

Definition:

```
TYPE E_RingBufferMode:
(
    None := 0,
    TimeBased,
    DataBased
)INT;
END_TYPE
```

Parameter

Name	Descriptiom
None	Recording without ringbuffer mode
TimeBased	Ringbuffer based on a given time periode
DataBased	Ringbuffer based on a given max data size

7.2.11 E_SetGetHistoricalDataState

Syntax

Definition:

```
TYPE E_SetGetHistoricalDataState :
(
    Pause,
    Continue_,
Restart,
Stop,
Update
)INT;
END_TYPE
```

Parameters

Name	Description
Break	Playback of the recording is paused.
Continue_	Playback of the recording continues.
Restart	Playback of the recording is restarted.
Stop	Playback of the recording is stopped.
Update	Parameters for playing the recording are updated.

7.2.12 E_StorageState

Syntax

Definition:

```
TYPE E_StorageState :
(
    unknown := 0,
    error
    starting
    online
    shuttingDownt
    offline
)INT;
END_TYPE
```

Parameters

Name	Description
Unknown	Status of the "Storage" is unknown
Error	"Storage" is in the error state. No more requests will be processed.
Starting	The "Storage" starts up and connects.
Online	The "Storage" is running and ready for requests.
ShuttingDown	The "Storage" is shut down.
Offline	The "Storage" is off and cannot be reached.

7.2.13 E_SymbolMode

Syntax

Definition:

```
TYPE E_SymbolMode :
(
    All := 0,
    Subset
)INT;
END_TYPE
```

Parameter

Name	Descriptiom
All	All symbols of the stream will be used
Subset	Only a subset of symbols will be used

7.2.14 E_PipelineCmdType

Syntax

Definition:

```
TYPE E_PipelineCmdType :
(
    RestartRule := 5,
    Start := 6,
    Stop :=  7
)INT;
END_TYPE
```

Parameters

Name	Description
RestartRule	Restarts a specific rule of a RuleEnginePipeline.
Start	Starts a RuleEnginePipeline
Stop	Stops a RuleEnginePipeline

8 Samples

8.1 PLC Client

This PLC sample shows the use of the TwinCAT Analytics Storage Provider library. The sample code shows reading and writing. For the sample to work coherently, both the use of the Analytics Logger for sending measured data to an MQTT Message Broker and the import of historical data via the Analytics Stream Helper are shown.

The basis is an appropriately set up native MQTT Message Broker and an Analytics Storage Provider service.

The PLC sample shows the following steps:

1. Analytics Logger: stream of variables from a Global Variable List to a MQTT Message Broker.
2. Analytics Storage Provider: starting and stopping stores and recordings, as well as reading recordings and historical data.
3. Analytics Stream Helper: receiving the historical data from the Analytics Storage Provider and mapping the data into a Global Variable List for the historical data.

Analytics Storage Provider GUID Glossary

When using the Storage Provider, various GUIDs occur that are required to identify services and data. The following describes where the GUIDs come from, what purpose they serve and where they can be viewed if necessary.

The analytics data stream sent by the Analytics Logger is basically described by three parameters:

1. **Topic** [STRING]
   Where is the data sent to?
2. **TwinCAT SystemID** [GUID]
   From which TwinCAT system is the data sent?
3. **LayoutID / Symbol Info ID** [GUID]
   What does the data look like? The GUID is a hash of the symbol information.

The above parameters are required to identify recordings at the Storage Provider.

Several Storage Providers can be connected to an MQTT Message Broker. Two parameters are required to identify a Storage Provider.

1. **MainTopic** [STRING](Where are the data/services provided?)
2. **ProviderGuid** [GUID](Unique identifier of the service)

RecorderGuid is used to recognize who has started a recording at the Storage Provider. This GUID is automatically generated at each Storage Provider Manager or client and attached to the StartRecord commands. In the PLC, this can be freely assigned at the StartRecord command.

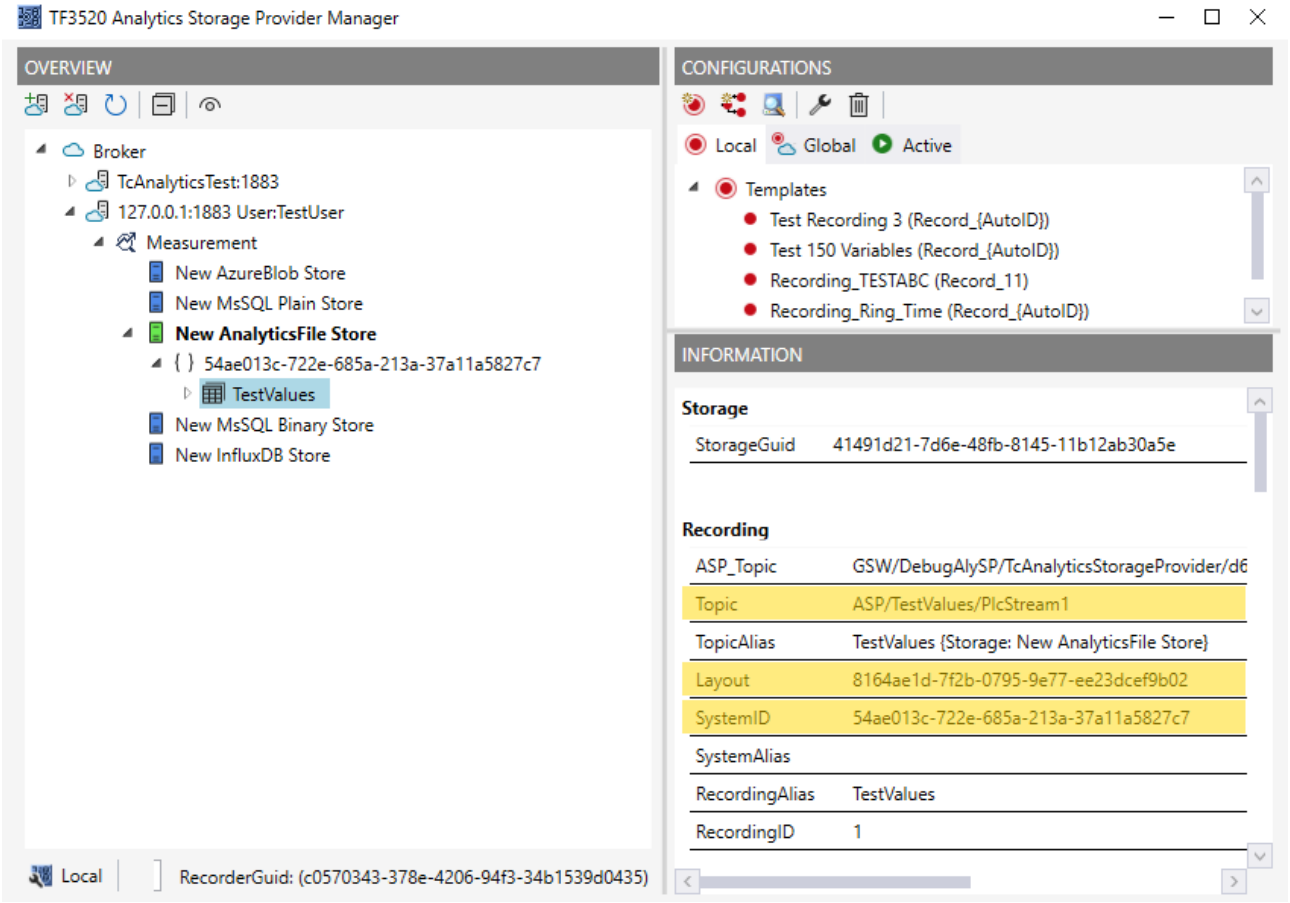
From version 3.2.14, three additional GUIDs can be specified on the StartRecord command.

1. **Storage** [GUID]
   This GUID specifies the storage in which the data is to be saved. The GUID is generated automatically in the Analytics Storage Provider Configurator. It can be read there or in the Analytics Storage Provider Manager. If no GUID is specified, the Master Storage is used.
2. **SubBroker/DataBroker** [GUID]
   This GUID specifies the data broker from which the Analytics Stream is to be received. The Analytics Storage Provider offers the option of recording from several message brokers. The GUID is generated automatically in the Analytics Storage Provider Configurator. It can be read there or in the Analytics Storage Provider Manager. If no GUID is specified, the Master Data Broker is used.

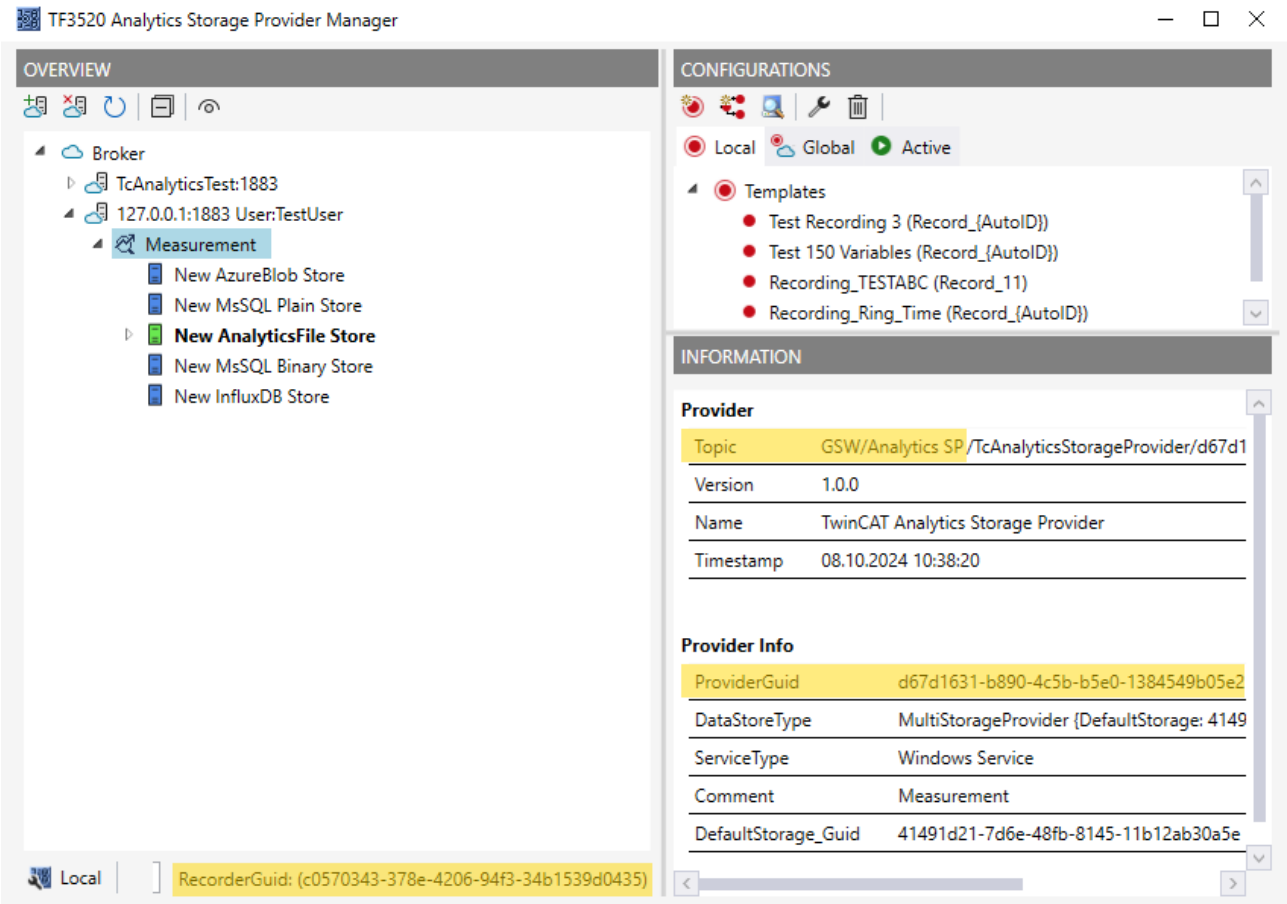
3. **DataKey** [GUID]
   The DataKey can be used to find, read and monitor recordings in progress. This DataKey can be freely selected in the PLC. If no DataKey is specified, a DataKey is automatically generated by the Storage Provider.

The following screenshots contain the parameters and GUIDs described above.

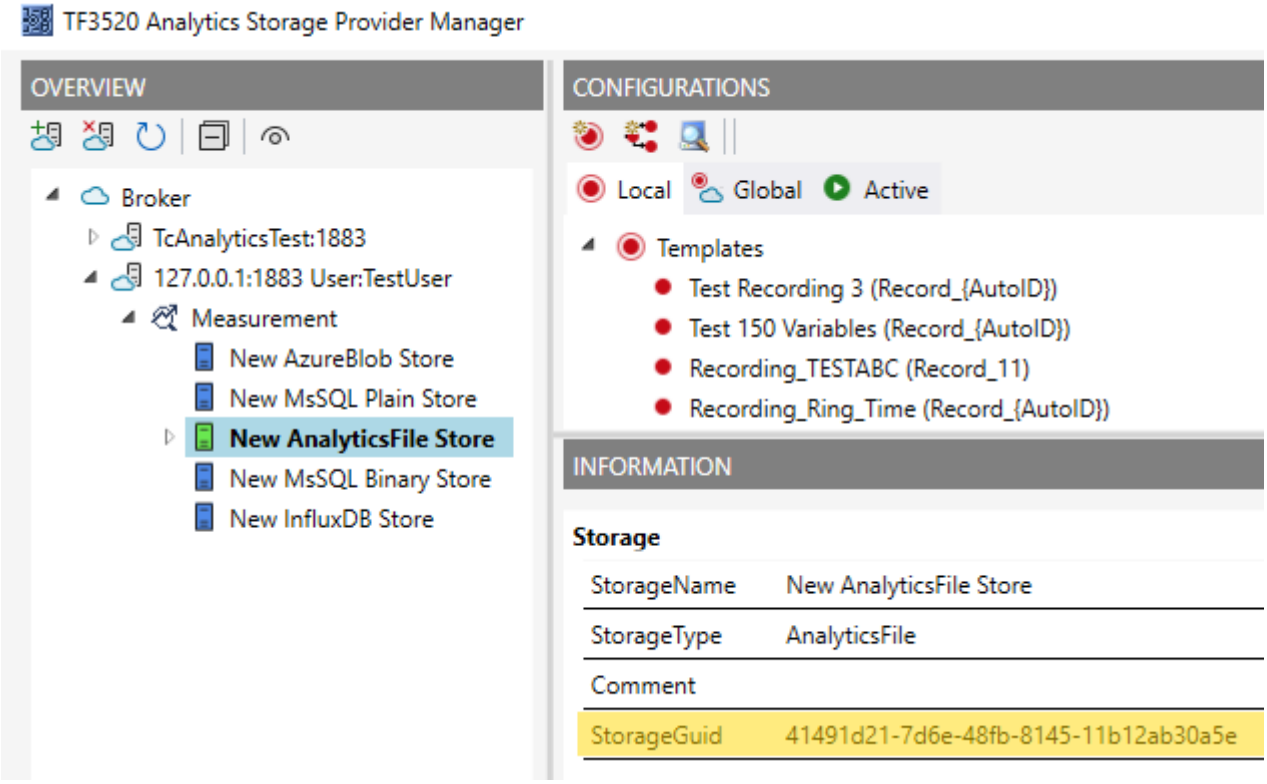
Topic / TwinCAT system ID / Layout (System Info ID)



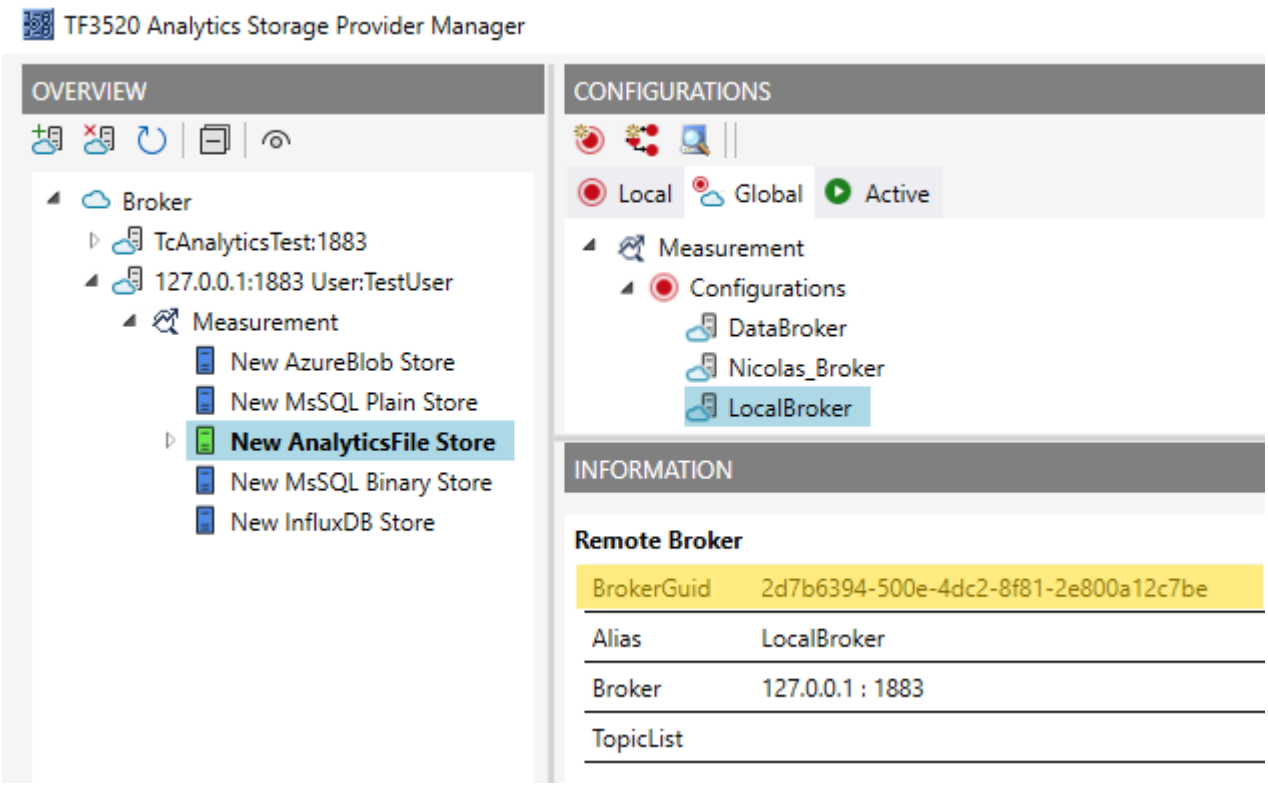
Storage Provider MainTopic / ProviderGuid / RecorderGuid



Storage GUID

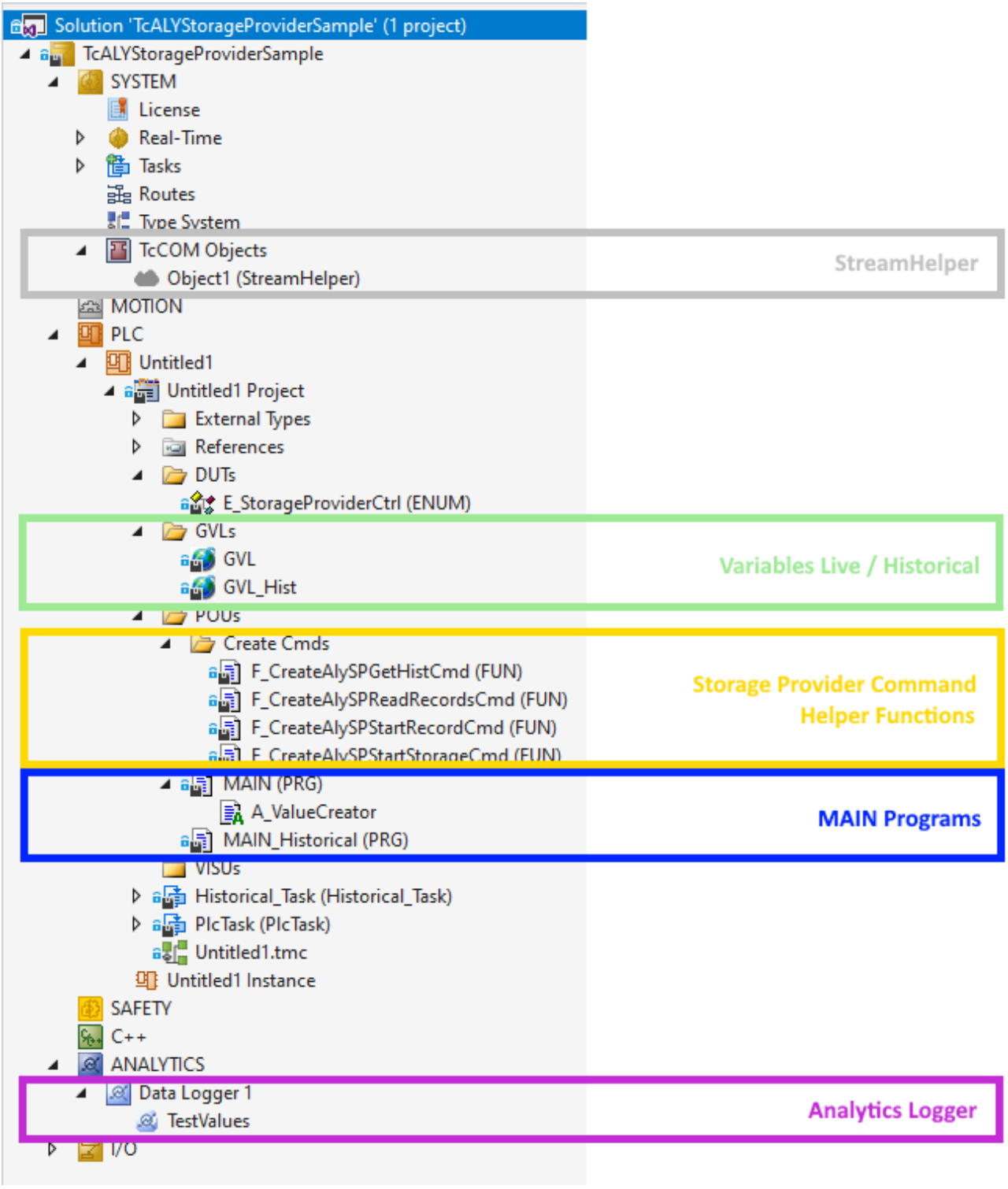


Data Broker GUID



Sample code architecture

All relevant parts of the configuration and the program code are marked in the following picture:



Stream Helper

For receiving the historical data sent by the Analytics Storage Provider via MQTT.

Variable Live/Historical

The GVL is for the live data and the GVL_Hist is for the historical data.

Storage Provider Command Helper Functions

These Helper Functions generate the commands for communication with the Storage Provider Service in JSON format.

MAIN program

The Main program invokes communication to the Analytics Storage Provider. The Main Historical program implements the mapping of historical data from the Stream Helper into the GVL_Hist.

Analytics Logger

Sends the variables of the GVL to an MQTT Message Broker.

Sample Start

NOTICE

Too little router memory can lead to system crashes

Increase the router memory in the real-time settings to 256 MB. It is also recommended to increase the maximum stack size of the global task configuration to 512 KB.

Before the sample can be started, you must set the MQTT Message Broker you are using in three different places.

Analytics Stream Helper:

Object | Context | Parameter (Init) | Parameter (Online) | Interfaces

	Name	Value	CS	Type	PTCID
-	MQTT				
	Host Name	172.17.62.145	☐	STRING(80)	0x020201...
	TcpPort	1883	☐	UINT	0x020201...
	Topic Prefix	_AlySPTest/ResultValues	☐	STRING(255)	0x020300...
	Client ID		☐	STRING(80)	0x020201...
	Username		☐	STRING(80)	0x020201...
	Password		☐	STRING(80)	0x020201...
	Max Handles	100	☐	UDINT	0x020300...
	Fifo Exponent	8	☐	UDINT	0x020300...
	Sender System Id		☐	STRING(80)	0x020300...
-	Tls				
	CaFile		☐	STRING(255)	0x020201...
	CertFile		☐	STRING(255)	0x020201...
	KeyFile		☐	STRING(255)	0x020201...
	KeyPw		☐	STRING(255)	0x020201...
	CrlFile		☐	STRING(255)	0x020201...
	PskId		☐	STRING(255)	0x020201...
	PskPwd		☐	STRING(255)	0x020201...
	Insecure	FALSE	☐	BOOL	0x020201...
	Version		☐	STRING(255)	0x020201...

Analytics Logger:

Context | Parameter (Init) | Data Area | TLS | Time Source

	Name	Value	CS	Type	PTCID
	Data Format	ANALYTICS_FORMAT_MQTT_BINARY		ANALYTICS_FORMAT	0x02020114
	Data Compression	ANALYTICS_COMPRESSION_RL		ANALYTICS_COMPRESS...	0x02030027
	Max. Compression Compare Width	ANALYTICS_COMP_WIDTH_8		ANALYTICS_COMPRESS...	0x02030030
	MQTT Host Name	172.17.62.145		STRING(80)	0x02020113
	MQTT Tcp Port	1883		UINT	0x02020103
	MQTT Main Topic	AlySPTest		STRING(255)	0x02030008
	MQTT Client ID			STRING(80)	0x02020101
	MQTT User Name			STRING(255)	0x02020106
	MQTT Password			STRING(80)	0x02020107

MAIN program:

```
MAIN
28  VAR
29      nState: INT;
30
31      fbALY_StorageProvider: FB_ALY_StorageProvider := (stConfig := (sMainTopic := 'GSW/Analytics SP',
32                                                                     sProviderGuid := STRING_TO_GUID('d67d1631-b890-4c5b-b5e0-1384549b05e2'),
33                                                                      stConnSettings := (sHostName := '172.17.64.145',
34                                                                                         sUserID := 'TestUser', sPassword := '123',
35                                                                                         sClientId := 'PLCClient')));
36
37      sLiveStreamTopic: T_MaxString := 'AlySPTest/TestValues';
38      sRecordingAlias: T_MaxString := 'PLC Recording';
39      sRecordName: T_MaxString := 'TestValueRec_{AutoID}';
40
41      stStorageCtrlCmd: T_ALY_SPStorageCtrl_Cmd;
42      stStartCmd: T_ALY_SPRecordData_Cmd;
43      stStopCmd: T_ALY_SPRecordData_Cmd;
44      stReadRecordsCmd: T_ALY_SPReadStreamRecords_Cmd;
45      stGetHistDataCmd: T_ALY_SPGetHistorical_Cmd;
46
47      sDataKey: GUID := STRING_TO_GUID('12345678-ABCD-1234-ABCD-123456789ABC');
48  END_VAR
```

You must then change the sMainTopic and the sProviderGUID for the FB_ALY_StorageProvider. This can be found as described above in this document.

```
MAIN
28  VAR
29      nState: INT;
30
31      fbALY_StorageProvider: FB_ALY_StorageProvider := (stConfig := (sMainTopic := 'GSW/Analytics SP',
32                                                                     sProviderGuid := STRING_TO_GUID('d67d1631-b890-4c5b-b5e0-1384549b05e2'),
33                                                                      stConnSettings := (sHostName := '172.17.64.145',
34                                                                                         sUserID := 'TestUser', sPassword := '123',
35                                                                                         sClientId := 'PLCClient')));
36
37      sLiveStreamTopic: T_MaxString := 'AlySPTest/TestValues';
38      sRecordingAlias: T_MaxString := 'PLC Recording';
39      sRecordName: T_MaxString := 'TestValueRec_{AutoID}';
40
41      stStorageCtrlCmd: T_ALY_SPStorageCtrl_Cmd;
42      stStartCmd: T_ALY_SPRecordData_Cmd;
43      stStopCmd: T_ALY_SPRecordData_Cmd;
44      stReadRecordsCmd: T_ALY_SPReadStreamRecords_Cmd;
45      stGetHistDataCmd: T_ALY_SPGetHistorical_Cmd;
46
47      sDataKey: GUID := STRING_TO_GUID('12345678-ABCD-1234-ABCD-123456789ABC');
48  END_VAR
```

Now go to the MAIN program to control the sample. With the enum eCtrl you can set the action you would like to perform. The available options are:

- ReadASPDescription (also contains the storage description)
- ReadDataBroker
- StartStorage
- ShutdownStorage
- StartRecord
- StopRecord
- IsRecordingRunning
- ReadHistoricalStreams

- ReadRecords
- GetHistorical

With a rising edge at the variable bExecute the action selected in the enum is executed. If you have made more than one record, you can see this in the array aRecordInfo. With the index it is then possible to select the different records. The timespans are also displayed, you could theoretically still adjust these within the timespan. To do this, you would need to modify the logic of the sample in the helper function F_CreateAlySPGetHistCmd accordingly.

The Storage Provider Recorder GUID selected in the document above can optionally be set in the PLC in the F_CreateAlySPStartRecordCmd function. Theoretically, it can be any GUID, it is only used to identify the recorder.

Download: https://infosys.beckhoff.com/content/1033/tf3500_tc3_analytics_logger/Resources/11270100747.zip

9 Appendix

9.1 Glossary

The following table explains key terms in connection with the Storage Provider.

Name	Function	Unique identifier (automatically generated)	Descriptive parameters (configurable by the user)
Storage Provider Configurator	Software tool for configuring the local Storage Provider.		
Storage Provider Manager	Software tool for working with the Storage Provider. This can be used to start, stop and manage data recordings. The Storage Provider Manager can be used for both the local and the remote Storage Provider.		
Storage Provider CLI Client	Command line tool for interactions with the Storage Provider.		
Storage Provider PLC library	PLC library for interactions with the Storage Provider.		
Storage Provider	Software application for historicizing data such as from the Analytics Logger. Both the acquisition of the data to be stored and the provision of stored data is carried out via MQTT. The Storage Provider can be operated under Windows and FreeBSD®.	ProviderGuid	ProviderAlias
Storage	Data sink of a Storage Provider (e.g. MS SQL or CSV).	StorageGuid	StorageAlias
Message Broker	MQTT message broker via which data can be transmitted using the MQTT protocol.		
HostBroker	Central message broker on which the Storage Provider provides information and receives commands.	BrokerGuid	BrokerAlias
DataBroker	Additional message broker from which the Storage Provider can obtain data to be recorded.	BrokerGuid	BrokerAlias
Pipeline	Description of a data flow. Components are data sources (Input Sources), processing steps (Rules) and data storage (Recordings).	PipelineGuid	PipelineAlias (can be set before starting the pipeline)
RuleEngine	Processing unit within the Storage Provider.		
Rule	Processing rule within a pipeline that is used for filtering, aggregating and sampling data.	RuleID	RuleAlias
Recording	Recording configuration for data. This includes which data should be recorded in which storage.	RecordingID	RecordingAlias (can be set before starting the pipeline)
Record	Data recording based on a defined recording configuration (Recording).	RecordID, additionally RecordDataKey if recording is running	RecordAlias (can be set before starting the pipeline)
Input Source	Data source for a pipeline. This can be an MQTT livestream or a HistoricalStream.		
LiveStream	Data from the Analytics Logger, IoT Data Agent or EK9160.		
HistoricalStream	MQTT data stream from the Storage Provider, which contains all information about a recording and the associated records. One HistoricalStream is generated per recording.	HistStreamID	HistStreamAlias - this corresponds to the RecordingAlias

Name	Function	Unique identifier (automatically generated)	Descriptive parameters (configurable by the user)
Recorder	Identification of the client that communicates with the provider. It is provided when the pipeline is started in order to be able to trace who started the pipeline.	RecorderGuid	RecorderAlias

9.2 FAQ - frequently asked questions and answers

In this section frequently asked questions are answered, in order to facilitate your work with the TwinCAT Analytics Storage Provider (ASP). If you have any further questions, please contact our support team at support@beckhoff.com.

1. How can I manage the table schema of MS SQL with ASP? [▶ 157]
2. Can I control the Storage Provider in a programmable way? [▶ 157]
3. Is it also possible to save results from the Analytics Runtime? [▶ 157]
4. Are open source software components used in TwinCAT Measurement products? [▶ 157]
5. What factors influence the data throughput of the storage provider? [▶ 157]

How can I manage the table schema of MS SQL with ASP?

You don't have to worry about the table schema. This is done completely by the Analytics Storage Provider. You only have to specify on which database server the data should be stored. If you want to see data in your own table structure, you have to stream the data into a TwinCAT Analytics Runtime and have the TwinCAT Database Server write the data in your structure.

Can I control the Storage Provider in a programmable way?

Yes, via the PLC interface for the TwinCAT Storage Provider. You can start/stop recordings or retrieve historical data (raw data or result data).

Is it also possible to save results from the Analytics Runtime?

Yes, this is possible. For this purpose, you can choose to send the results to an MQTT Message Broker when generating the Analytics Runtime from the Analytics Workbench configurator. This data stream can be captured by the Storage Provider.

Are open source software components used in TwinCAT Measurement products?

Yes, various open source components are used.

Please see the information on the page Third-party components [▶ 158].

What factors influence the data throughput of the storage provider?

The data throughput depends on many influencing variables. Primarily of system and network resources. An overview:

- System properties (CPU, RAM)
- Writing speed and quality of the storage medium (SSD)
- Network properties
- Complexity of symbolism (data type, structures, arrays, etc.)
- Mode of historization (total symbolism allows higher throughput, a subset may be more costly depending on its size)
- Compression level of the stream (the stronger the compression, the higher the system load)
- The size of a sample
- Total size of a data packet (number of samples per packet)
- The number of parallel recordings that the storage provider manages

9.3 Third-party components

This software contains third-party components.
Please refer to the license file provided in the following folder for further information:
C:\Program Files(x86)\Beckhoff\Legal\TwinCAT-XAR-AnalyticsStorageProvider

10 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Download finder

Our download finder contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline:	+49 5246 963-157
e-mail:	support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline:	+49 5246 963-460
e-mail:	service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49 5246 963-0
e-mail:	info@beckhoff.com
web:	www.beckhoff.com

Trademark statements

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.

Third-party trademark statements

Arm, Arm9 and Cortex are trademarks or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

FreeBSD is a registered trademark of The FreeBSD Foundation and is used by Beckhoff with the permission of The FreeBSD Foundation.

Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

More Information:
www.beckhoff.com/tf3520

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com