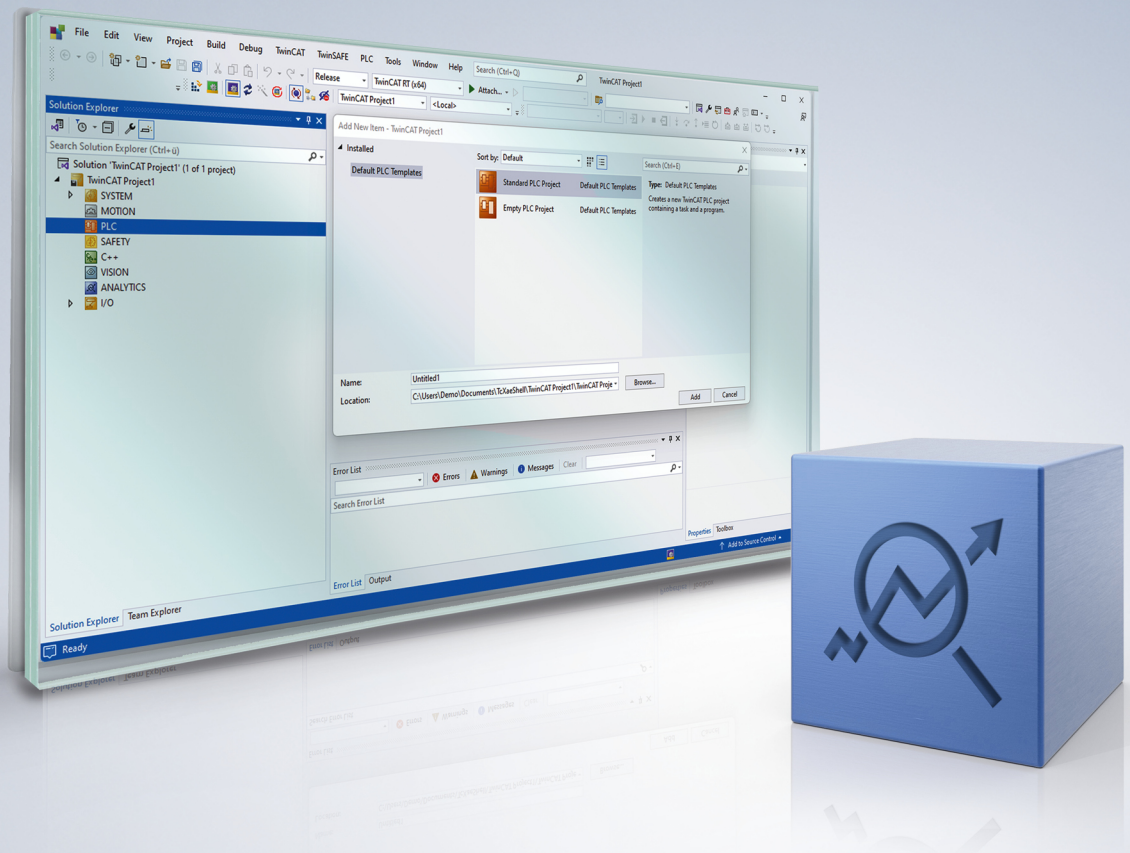
BECKHOFF New Automation Technology

Manual | EN

TF3500

TwinCAT 3 | Analytics Logger



2026-10-07 | Version: 1.6.0

Table of contents

1 Foreword 5
1.1 Notes on the documentation 5
1.2 For your safety 6
1.3 Notes on information security 7
2 Overview 8
3 Installation 9
3.1 System requirements 9
3.2 Installation under TwinCAT 4026 9
3.3 Licensing 9
4 Analytics Workflow - First Steps 12
4.1 Recording data from the machine 12
4.2 Communication 18
4.3 Historicize data 19
4.4 Analyse data 26
4.5 24h Analytics application 31
5 Technical introduction 41
5.1 Basic Concepts 41
5.2 MQTT basics 42
5.3 Data Compression 48
6 Configuration 49
6.1 Basic settings 49
6.1.1 Stream Sources 50
6.1.2 Stream Targets 53
6.1.3 Stream Overview 60
6.1.4 Device Info 60
6.1.5 Miscellaneous 61
6.2 Context settings 66
6.3 Stream settings 67
6.3.1 Symbol Selection 67
6.3.2 Trigger configuration 70
6.3.3 Online status 71
6.3.4 Data Handling 72
6.3.5 Stream Info 78
7 API 80
7.1 PLC 80
7.1.1 Analytics Communication Library 80
7.1.2 Obsolete 103
7.2 Automation Interface 104
8 Samples 105
9 Appendix 106
9.1 FAQ - frequently asked questions and answers 106
9.2 Cybersecurity risk analysis 106

10 Support and Service 108

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

© Beckhoff Automation GmbH & Co. KG, Germany.
The distribution and reproduction of this document, as well as the use and communication of its contents without express authorization, are prohibited.
Offenders will be held liable for the payment of damages. All rights reserved in the event that a patent, utility model, or design are registered.

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

The TwinCAT Analytics Logger records process and application data of the machine controller in synchronization with task cycles. The logger is characterized by its high performance as it operates directly in the real-time context of the TwinCAT controller.

The TwinCAT Analytics Logger may either act as an MQTT client and transmit the data to a MQTT message broker on a regular basis (called MQTT-mode) or store the data locally in a file on the hard disk of the machine controller (called file-mode). The configuration required is performed in Microsoft Visual Studio. All variables of the process image and the PLC application can be added easily to the configuration via check boxes without the need for programming.

When used as an MQTT Client the Logger is able to bypass short disconnects to the Message Broker using a ring buffer functionality to prevent loss of data temporarily. Used in file-mode, a ring buffer can also be configured, which may be useful in case of limited storage capacity or if there is no need for data to be recorded (henceforth referred to as logged) permanently, instead focusing on a fixed time interval.

The logged data may be used in various ways but its main intention is data analysis with TwinCAT Analytics as wells as data visualization with TwinCAT Scope.

Components

- Configuration surface in TwinCAT project tree
- Description files TcAnalytics.tmc and TcIotBase.tmc
- Drivers TcAnayltics.sys and TcIotDrivers.sys

List of key features

Functionality	TwinCAT 3 Analytics Logger as MQTT client	TwinCAT 3 Analytics Logger for local storage
Programable record control	Yes	Yes
Configuration Interface	Yes	Yes
RT Context	Yes	Yes
MQTT	Yes	No
Analytics Binary Format	Yes	Yes
JSON Format	No	No
File Storage	No	Yes
Ring Buffer	Yes	Yes
Authentication	Yes	No
Encryption	Yes	Yes
Compression	Yes	Yes

3 Installation

The TwinCAT Analytics Logger is installed on the TwinCAT Engineering system via a workload. Once configuration is complete, all necessary data is downloaded to the target system upon activation. No additional installation on the target system is necessary. To use the Analytics Logger, a “TC3 Analytics Logger” license is required for each target device. This can be either a perpetual license or a 7-day trial license.

General information on installation and licensing can be found in the following subchapters.

3.1 System requirements

Technical data	TF3500 | TwinCAT 3 Analytics Logger
Operating system	Windows 10, Windows 11, TwinCAT/BSD, Beckhoff RT Linux® PC (x86, x64, and Arm®)
Min. TwinCAT version	3.1.4022.31
Min. TwinCAT level	TC1100 | TwinCAT 3 I/O
Trial license	Available for 7 days without restriction

3.2 Installation under TwinCAT 4026

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 TF3500
```

You can use the following command to install the workloads of the TF3500 TwinCAT 3 Analytics Logger function.

```
tcpkg install TF3500.AnalyticsLogger.XAE
```

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.

3.3 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).

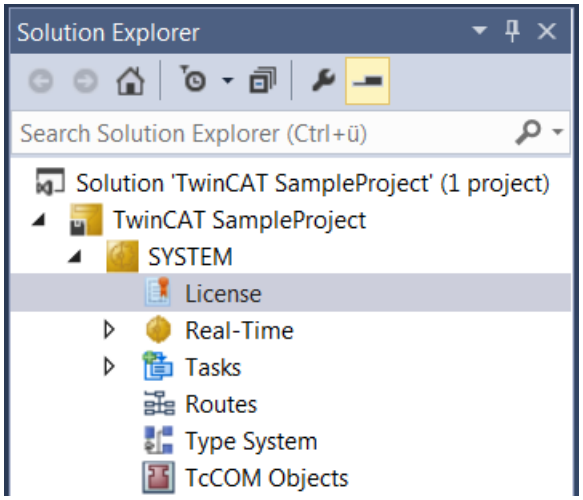
Licensing the full version of a TwinCAT 3 Function

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

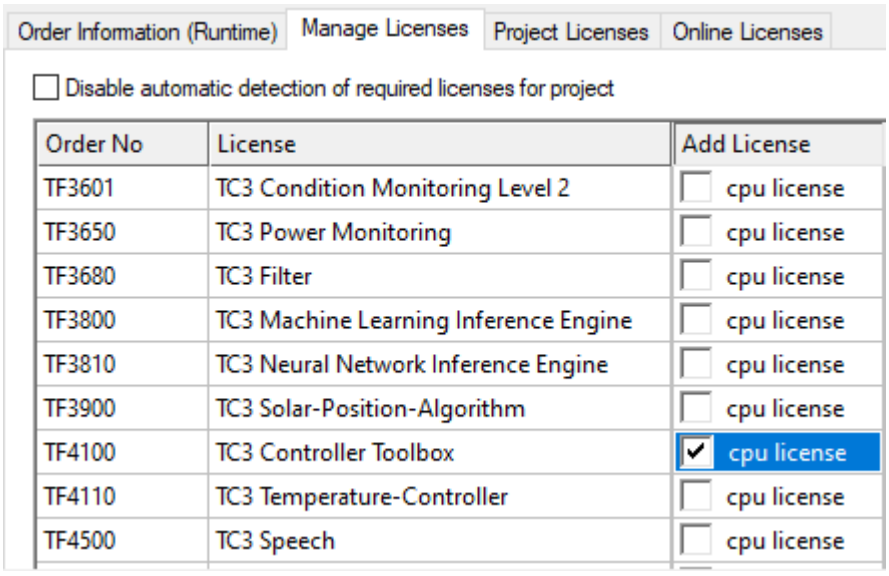
Licensing the 7-day test version of a TwinCAT 3 Function

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.



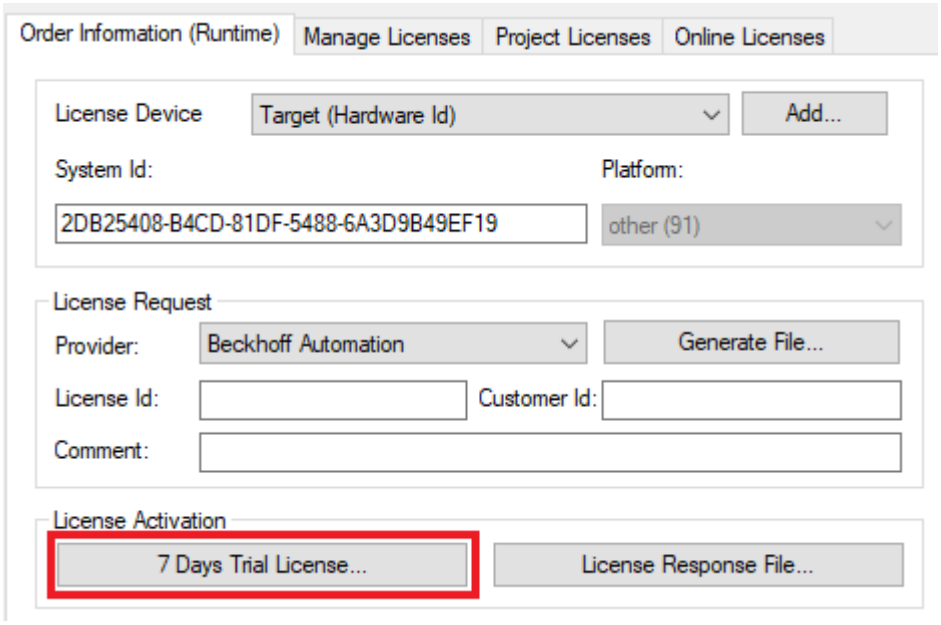
   ⇨ The TwinCAT 3 license manager opens.
5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. "TF4100 TC3 Controller Toolbox").



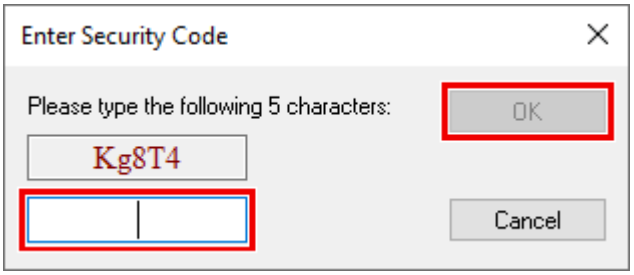
Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	cpu license
TF3650	TC3 Power Monitoring	cpu license
TF3680	TC3 Filter	cpu license
TF3800	TC3 Machine Learning Inference Engine	cpu license
TF3810	TC3 Neural Network Inference Engine	cpu license
TF3900	TC3 Solar-Position-Algorithm	cpu 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.

4 Analytics Workflow - First Steps

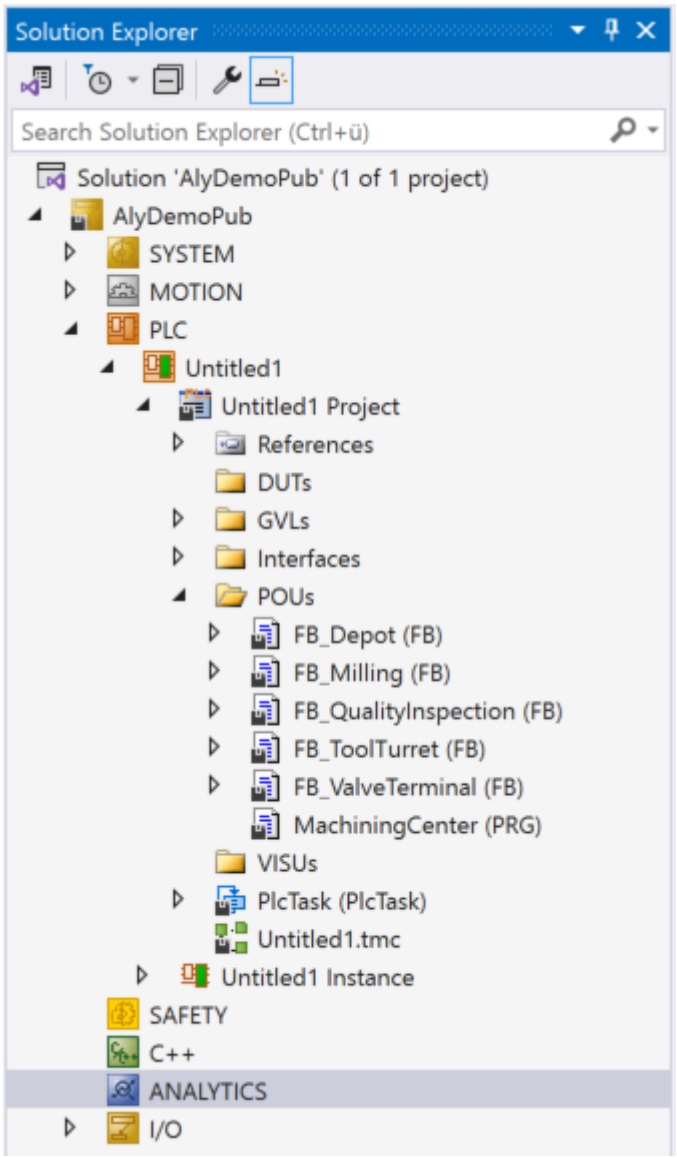
This step by step documentation presents the complete TwinCAT Analytics workflow. From the data acquisition over the communication and historizing up to the evaluation and analysis of the data and to the presentation of the data in web-based dashboard.

4.1 Recording data from the machine

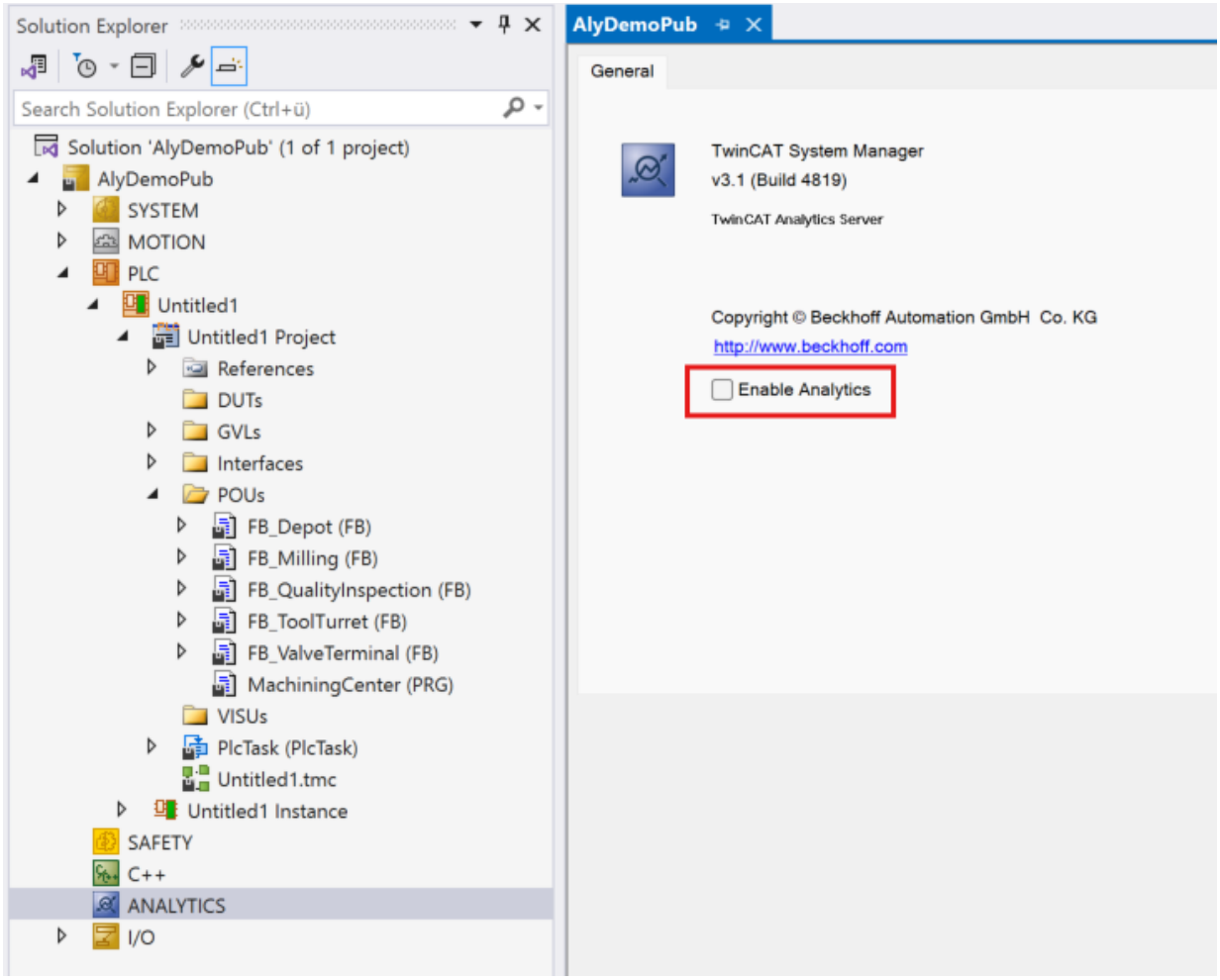
The Analytics Logger is the central unit for collecting machine data. It records process data from the I/Os, PLC, NC etc. on demand, cycle-synchronous. The Logger works deeply integrated in the real-time context of TwinCAT. However, it can also communicate specific data, such as IPC diagnostics.

The TwinCAT Analytics Logger is installed exclusively on the engineering system via its workload TF3500.AnalyticsLogger.XAE. This is followed by the configuration, which is downloaded to the target system with an Activate Configuration together with all the necessary drivers. Additional installations on the target system are therefore not necessary. 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 from the message broker with a ring buffer functionality. You can configure a ring buffer as well for the local binary file storage. The optional data compression ensures the lowest possible hard disk space requirement.

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

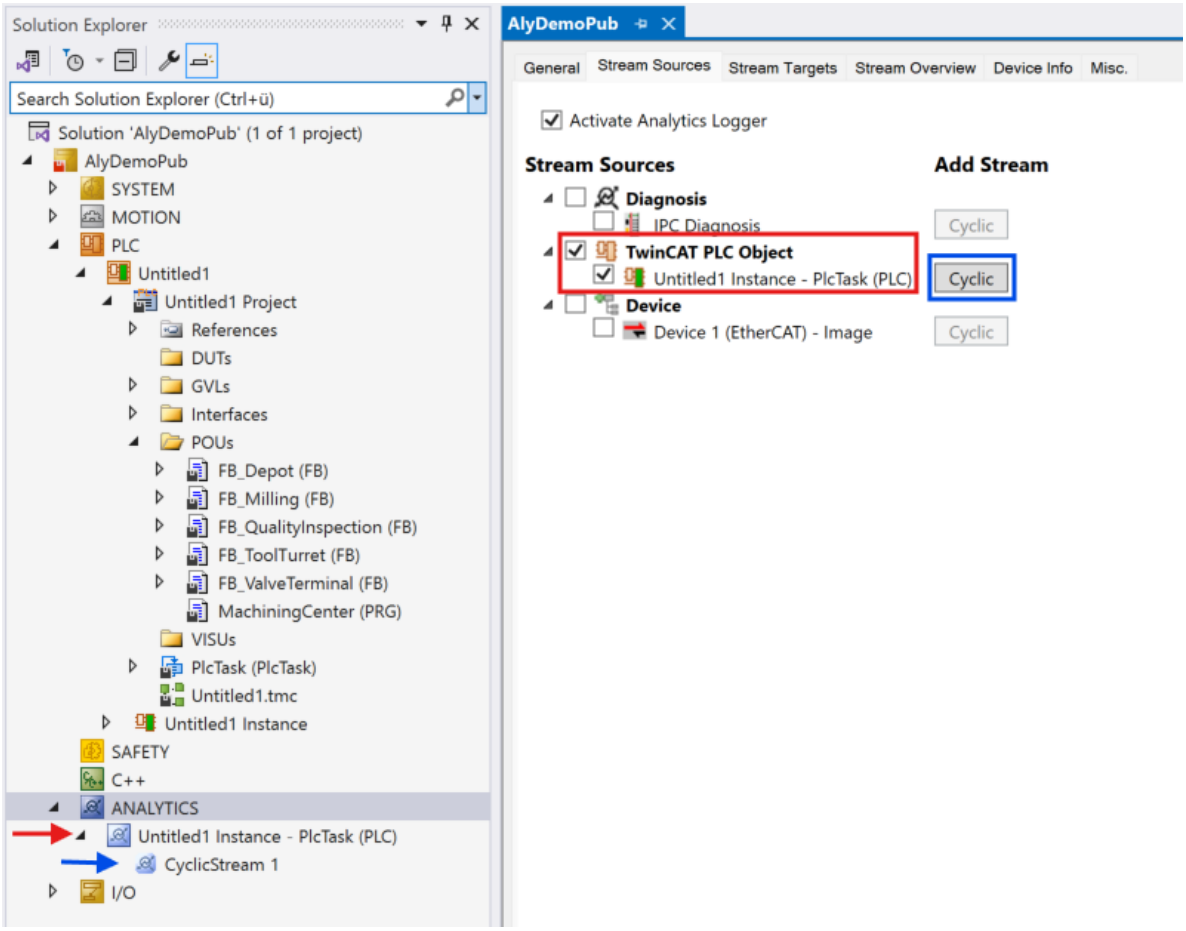


1. Double-click to open the General tab and activate the **Enable Analytics** checkbox.



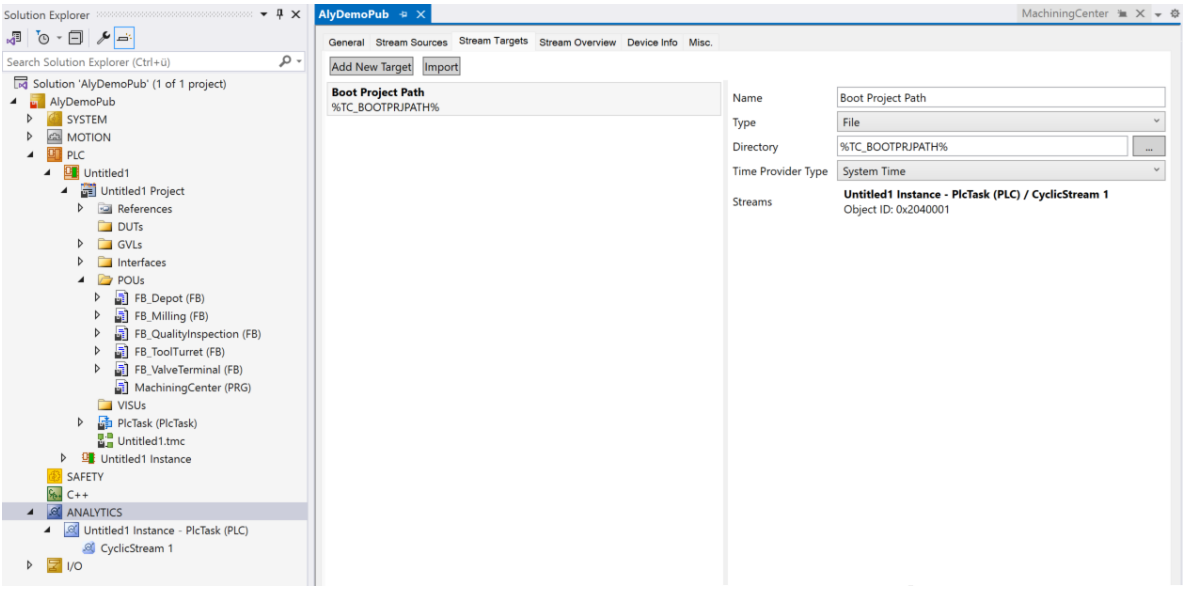
2. All necessary configuration tabs of the first level are then automatically activated. On the **Stream Sources** tab, choose the sources from which you want to record data.
3. To do this, activate the corresponding checkboxes (example red: PLC instance). Then add one or more streams for the corresponding source using the button on the right-hand side (blue), which is then enabled.

⇨ The streams are also visible in the Solution Explorer.

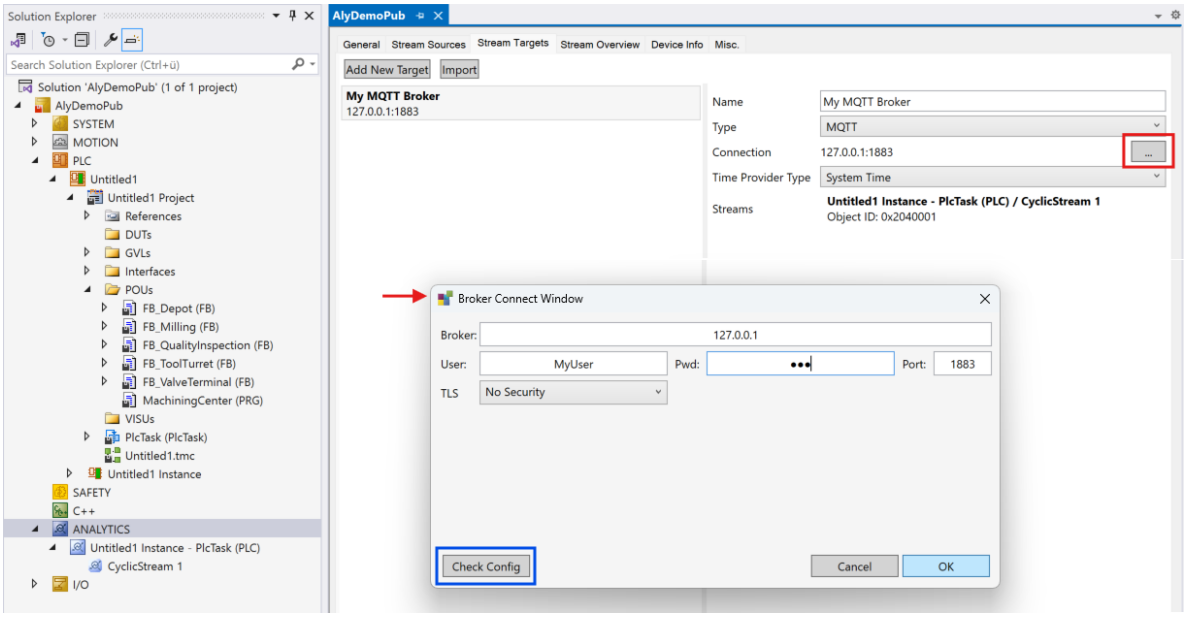


4. At the same level, switch to the **Stream Targets** tab to define data targets in the form of analytics files or MQTT message brokers.

   ⇨ An analytics file target is always set as the default. The binary data format files are stored in the TwinCAT boot directory. You can use this target as it is or customize it. The following settings can be made on the right-hand side of the window:
   Name: Name of the target
   Type: File
   Directory: Path to the Analytics file folder
   Time Provider Type: Choice of time provider for possible time synchronization



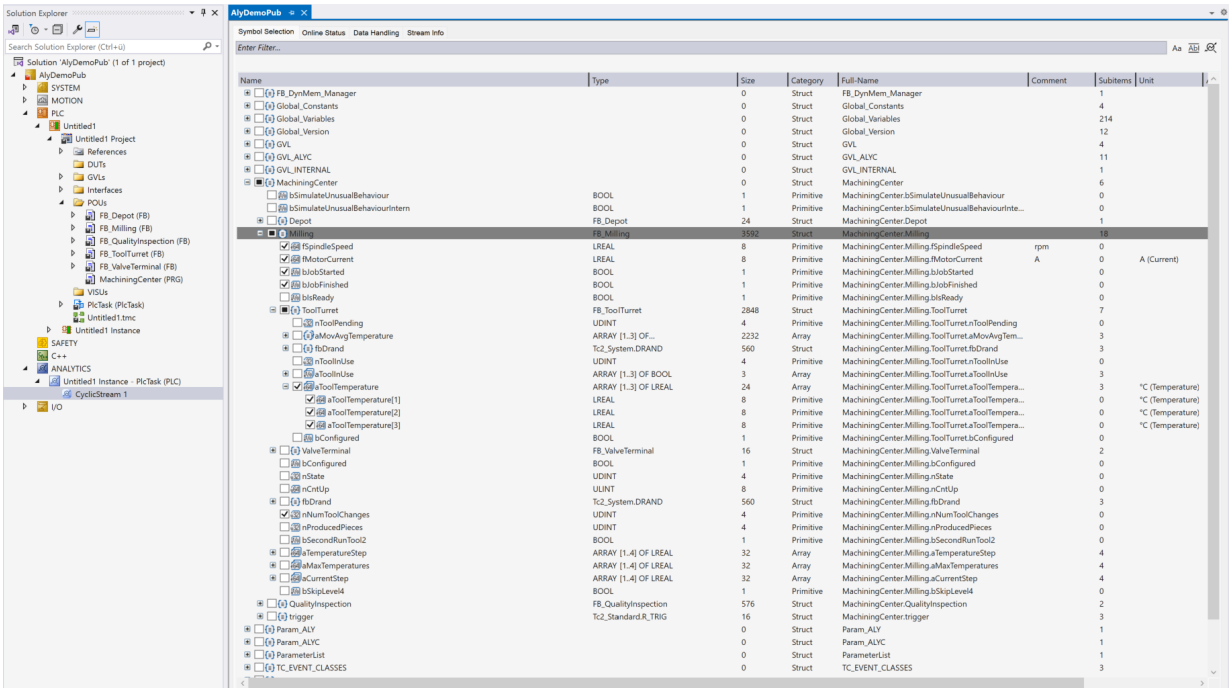
⇨ As an alternative to the previous step, you can change the type to MQTT. This adjusts the setting options:
Name: Name of the target
Type: MQTT
Connection: Connection data to a native MQTT message broker
Time Provider Type: Choice of time provider for a possible time synchronization



If you click on the button with the three dots (red) in the connection property, an additional **Broker Connect** Window opens. You can make the following detailed settings here:
Broker: IP address or host name of the broker
User: The user name for logging in to the broker
Pwd: The user's password
Port: The MQTT port can be entered here. The default value is 1883; as soon as you use certificates, the port automatically changes to 8883
TLS: Here you can choose between No Security, Use Certificates and Use Pre-Shared Key (PSK). Details on the certificate application can be found in the TF3500 Analytics Logger documentation

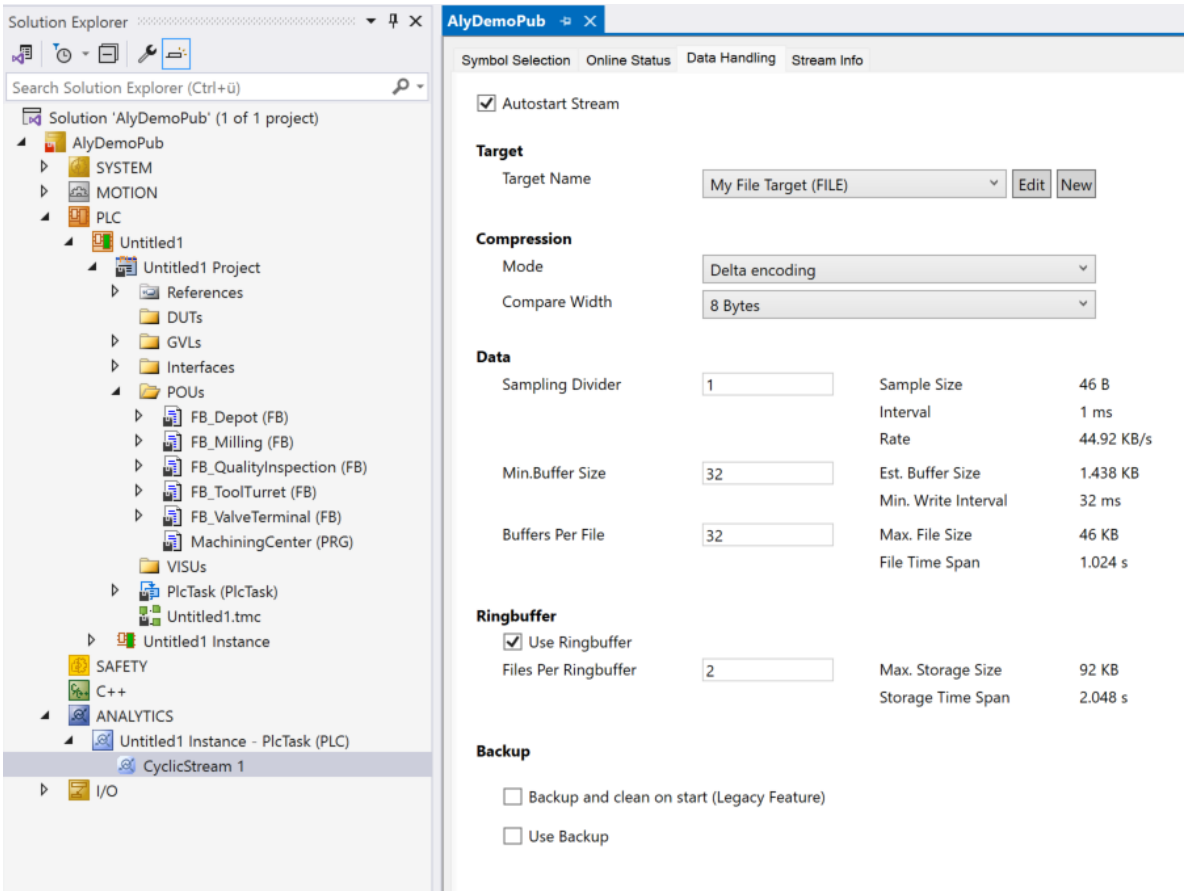
⇨ You can then switch directly to the third level, the stream level.

5. In the **Symbol Selection** tab, you will now see all the variables that you can choose for this stream. To do this, set the checkboxes on the variables accordingly. It is also possible to choose entire branches, function blocks or structures with the respective higher-level checkbox.

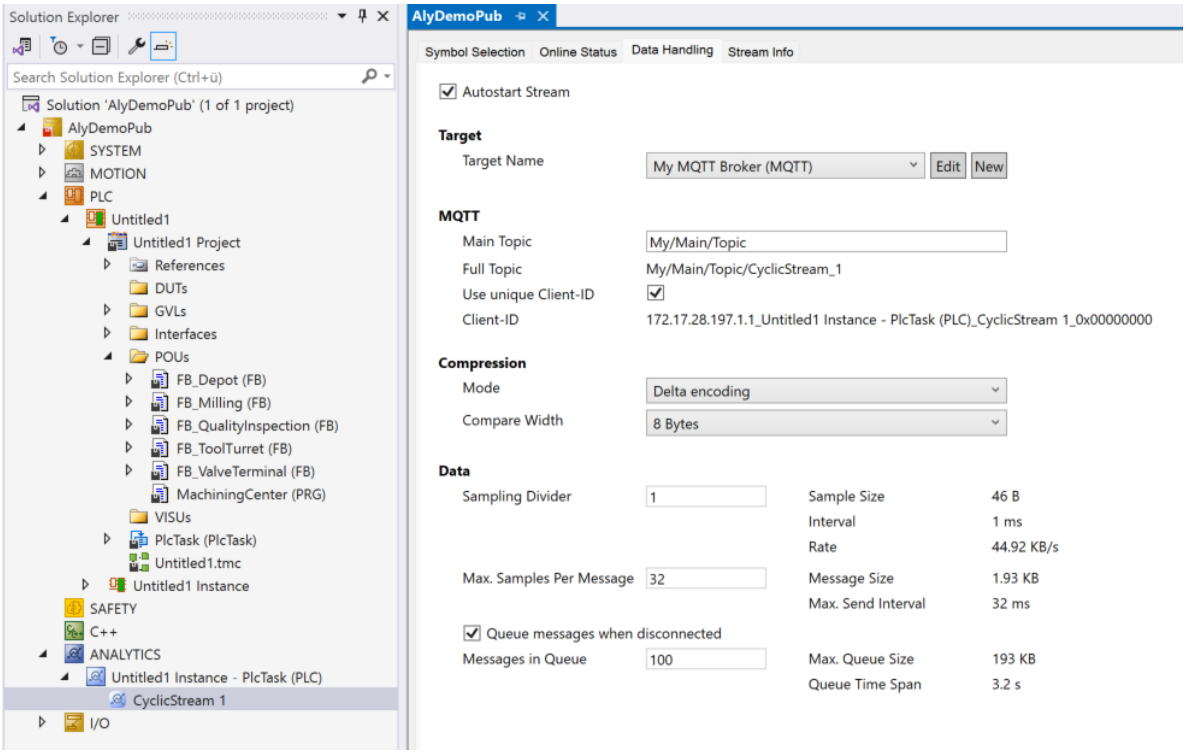


6. In the last step, switch to the **Data Handling** tab. The assignment of the respective stream to one of the created stream targets now takes place here. Here you can define which data is to be recorded and where. The masks of the setting options differ in that you have selected an analytics file or an MQTT broker as the target. Some settings are generally valid and do not depend on the target:
Compression: Here you can switch data compression on and off.
Sampling Divider: Here you can set a factor that ensures that the data is sampled at a rate that is a multiple slower than the task cycle time.

⇨ If your target is an analytics file, you also have the option of setting a ring buffer. In addition, the Use Backup option allows data to be saved in a separate analytics file in parallel to the actual recording of an event. A very useful function, especially when troubleshooting. Details can be found in the TF3500 Analytics Logger documentation.



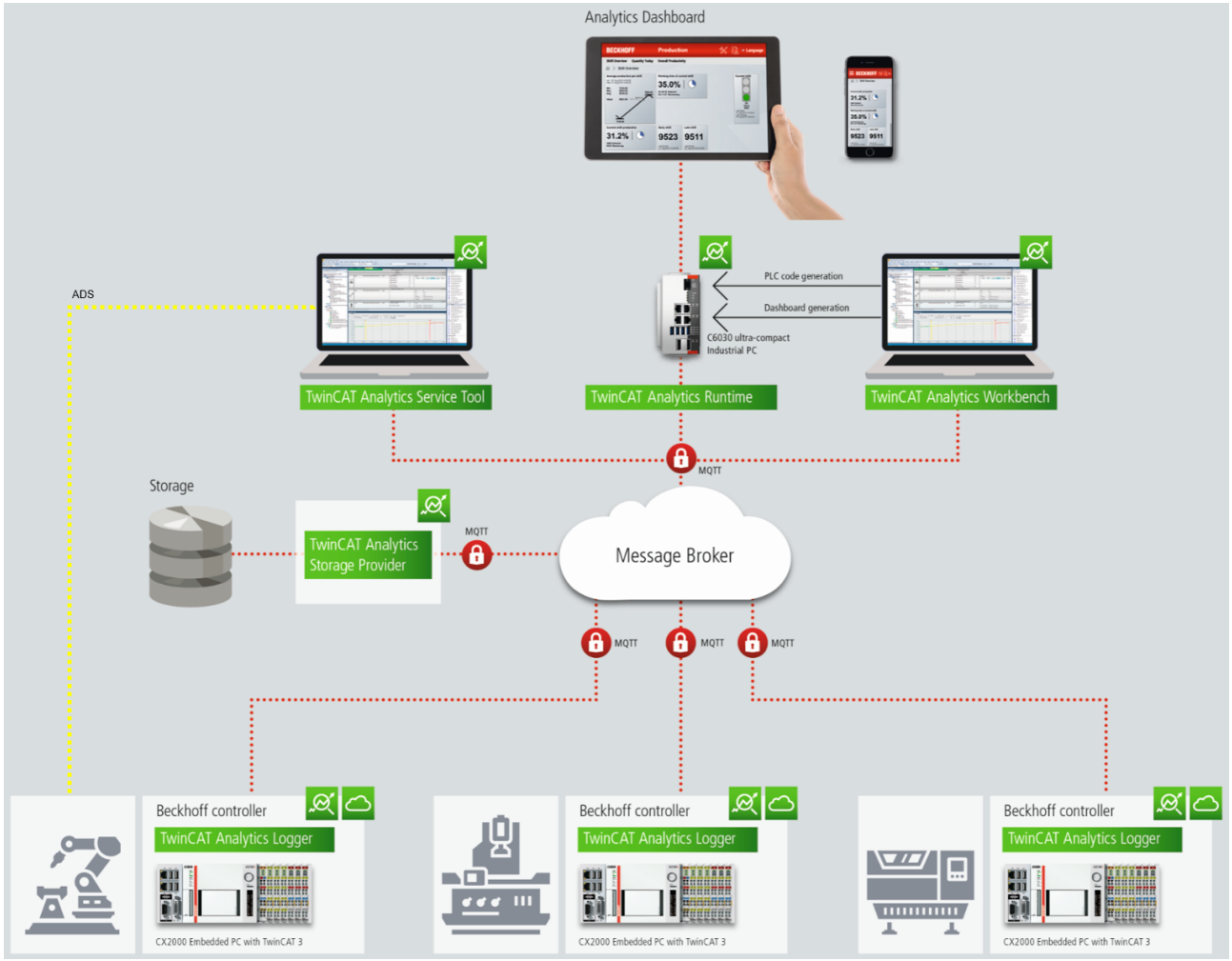
⇨ If your target is an MQTT broker, you also have the option of specifying a topic name for the organization on the message broker. It is also possible to switch on and set the buffering of data when disconnecting from the message broker.
Details can be found in the TF3500 Analytics Logger documentation.



⇨ The Analytics Logger configuration is now complete. As the Autostart Stream option is selected by default in the Data Handling tab, the Analytics Logger starts working directly when the TwinCAT configuration is activated on the corresponding target system.

4.2 Communication

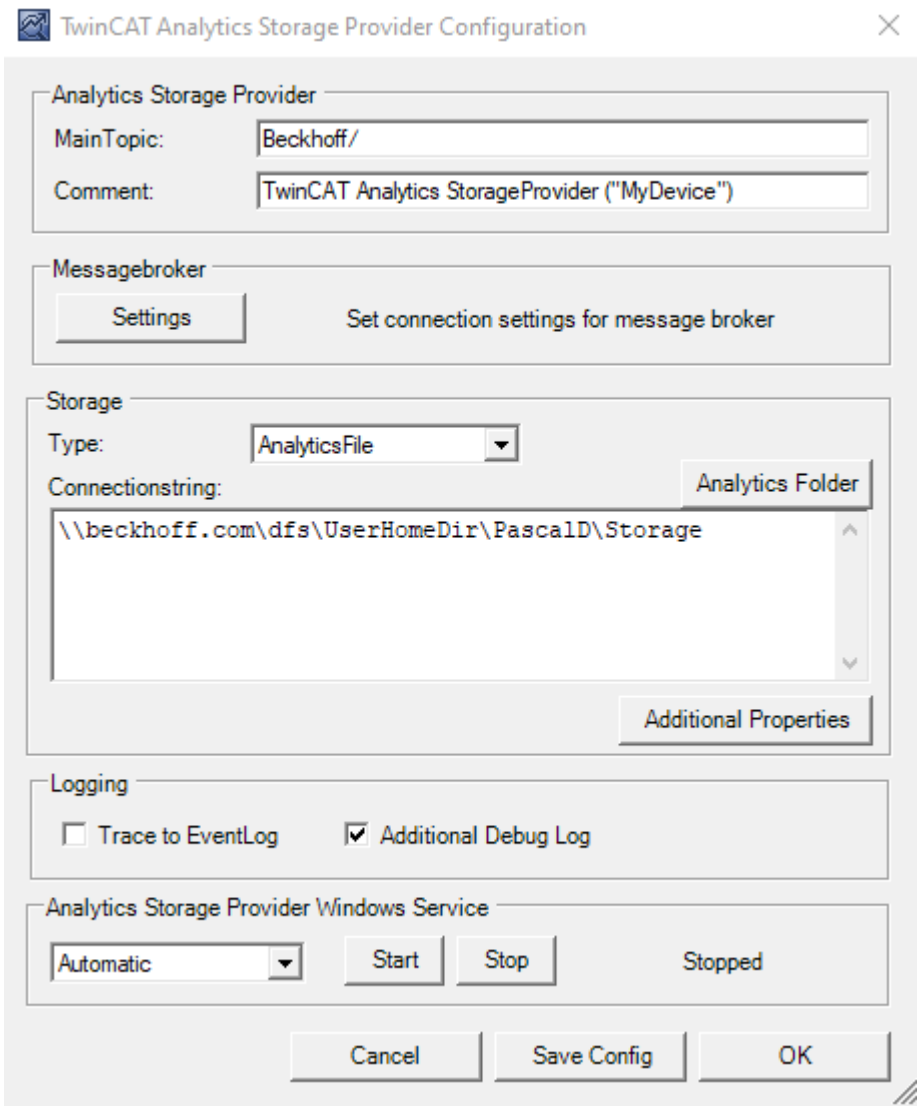
Currently, the Analytics workflow is fully mappable via MQTT. The engineering tools can also access the data of the machines via ADS and carry out analyzes.



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*.

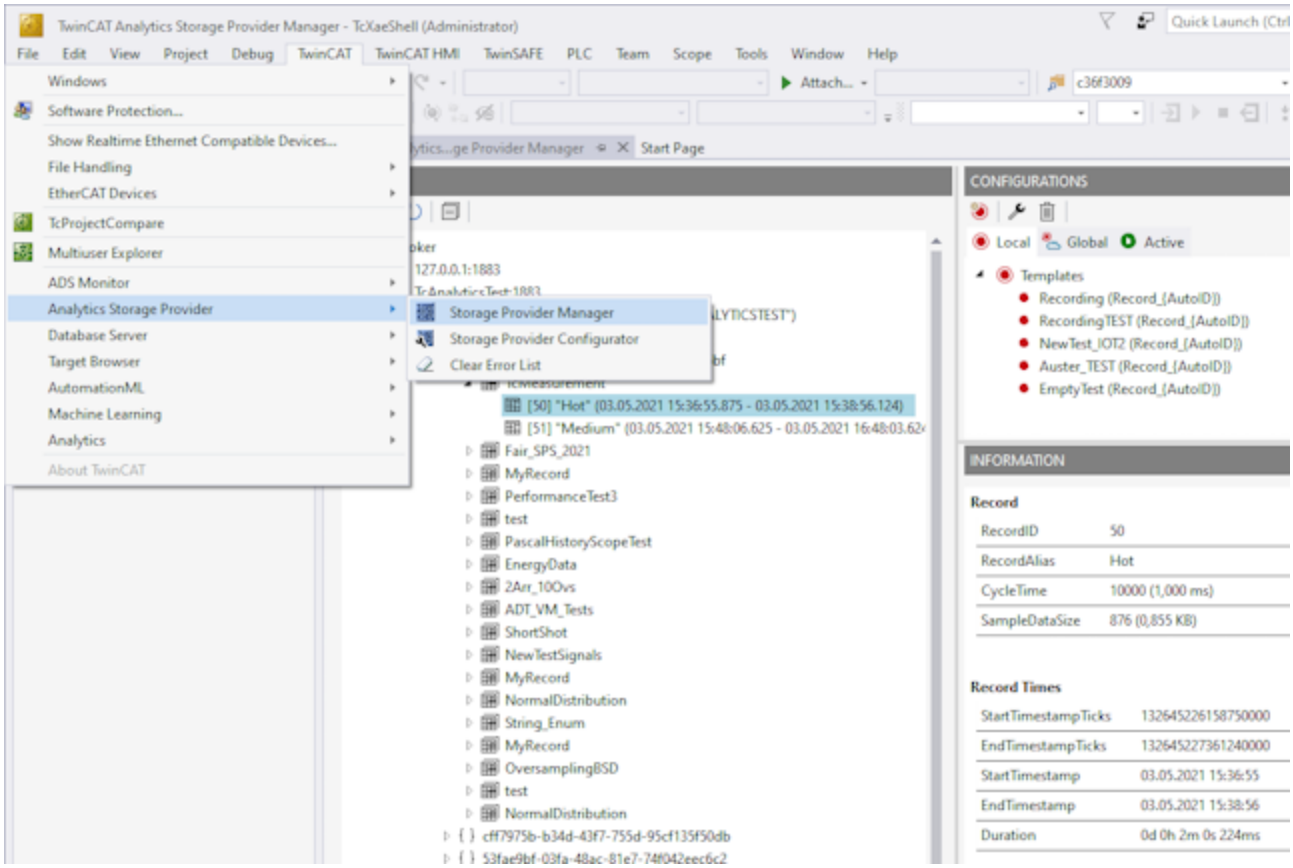


The main part of the topic can be defined in the configuration as well as the comment, which is used for identification if more than one Storage Provider is registered with the message broker.

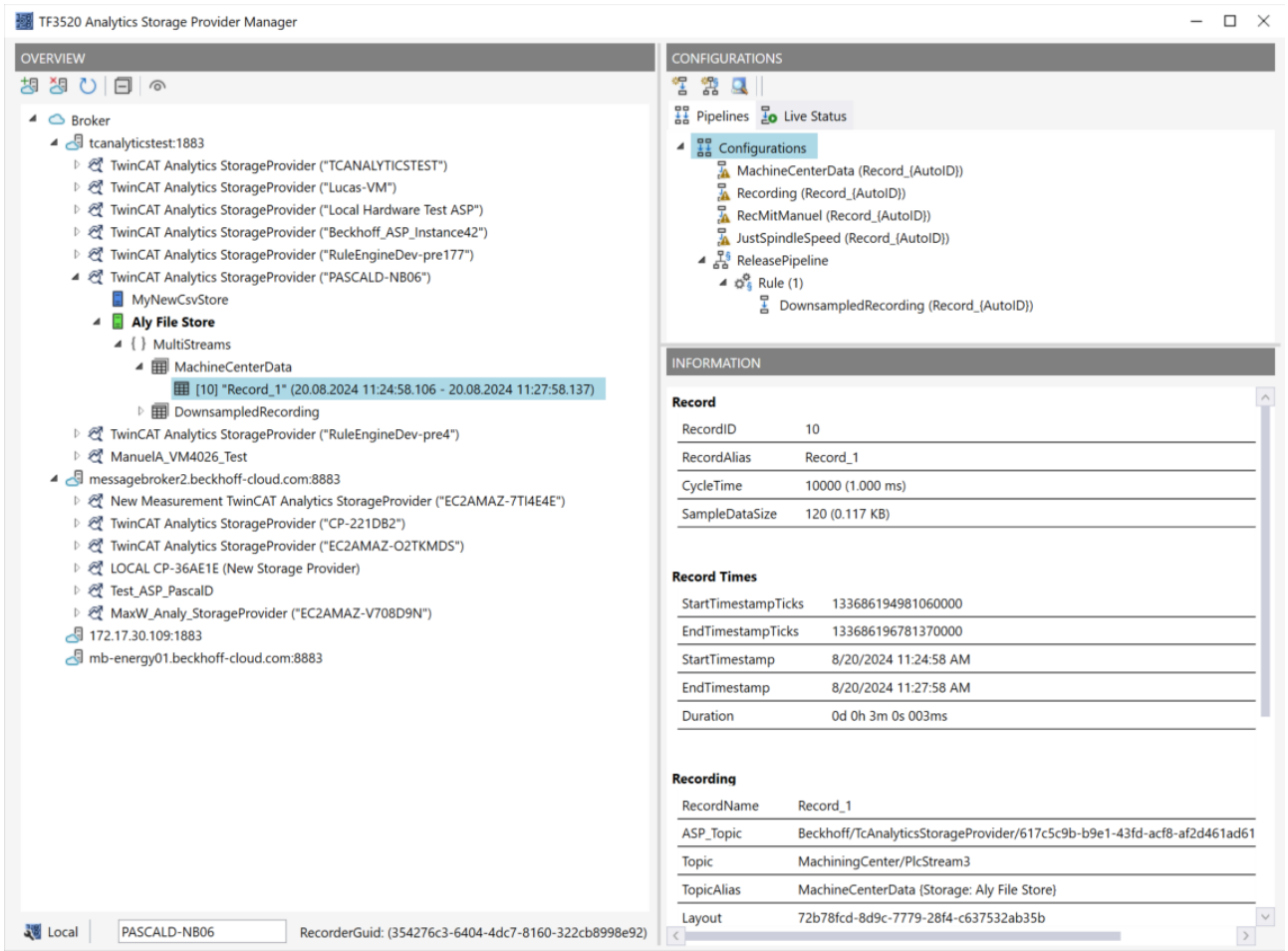
You can make the message broker settings and decide on a storage type:

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

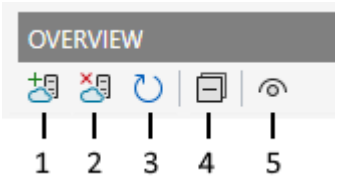
Finally you can save the configuration and start the service. The next step is to configure the specific recording. For this you should select the **Storage Provider Manager** in your development environment.



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



Toolbar Manager window (“OVERVIEW”)

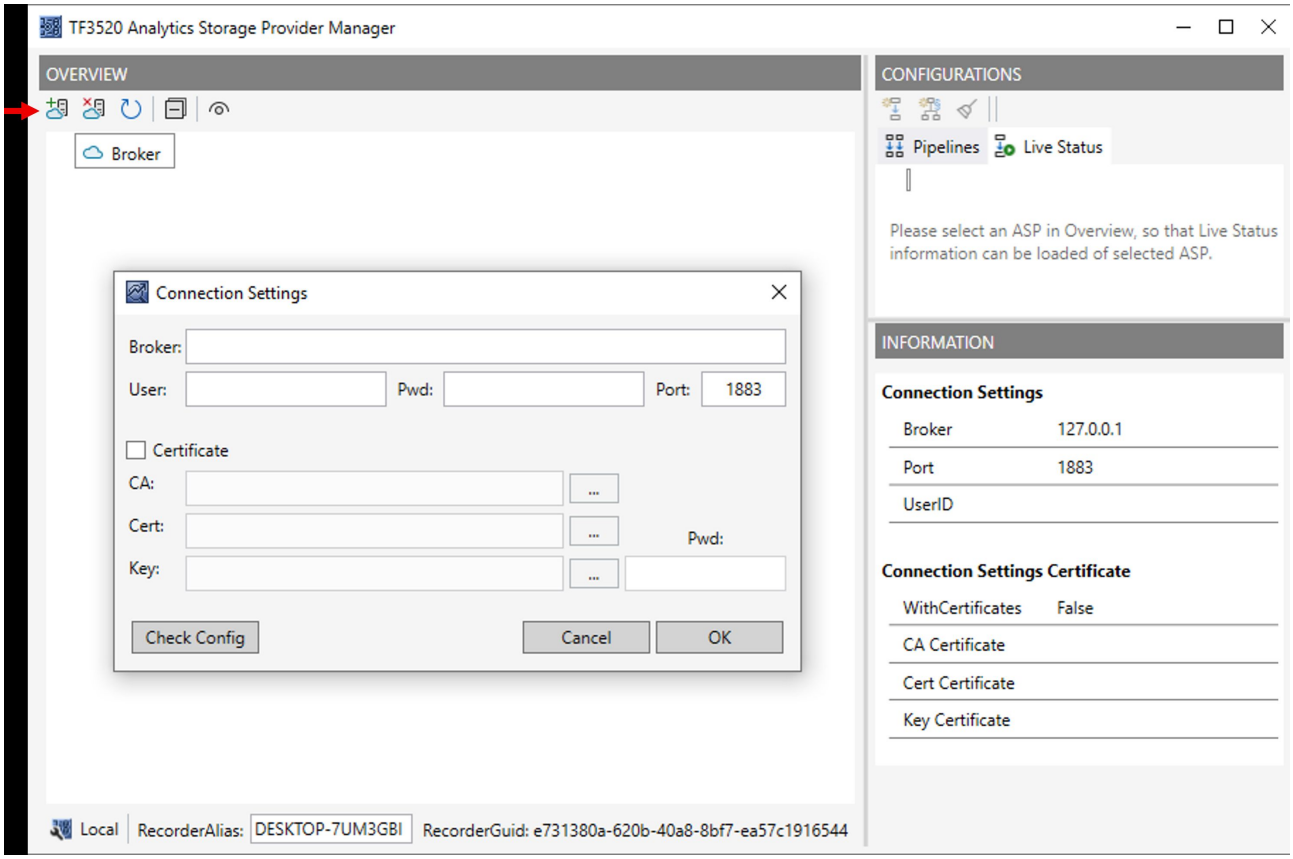


1	Add new broker
2	Remove selected broker
3	Refresh display
4	Collapse all nodes
5	Switching between dark and light modes

Function Manager window (“OVERVIEW”)

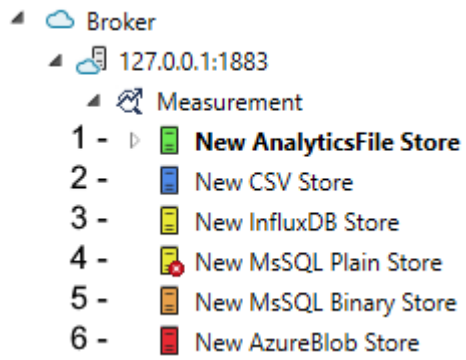
Assign a “RecorderAlias” first. This helps group the started recordings and find the recordings you started yourself again.

After that, one or more brokers can be set up. This is accomplished using the familiar input mask for MQTT connection properties.



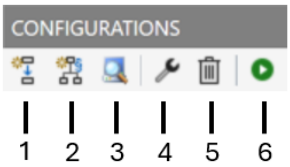
After establishing a connection to the broker, all Analytics Storage Providers connected to it will be listed.

“Storage” status



1	Storage online
2	Storage offline
3	Storage starting
4	Storage starts with an error. Still trying to start it.
5	Storage is shutting down
6	Storage is in the error state.

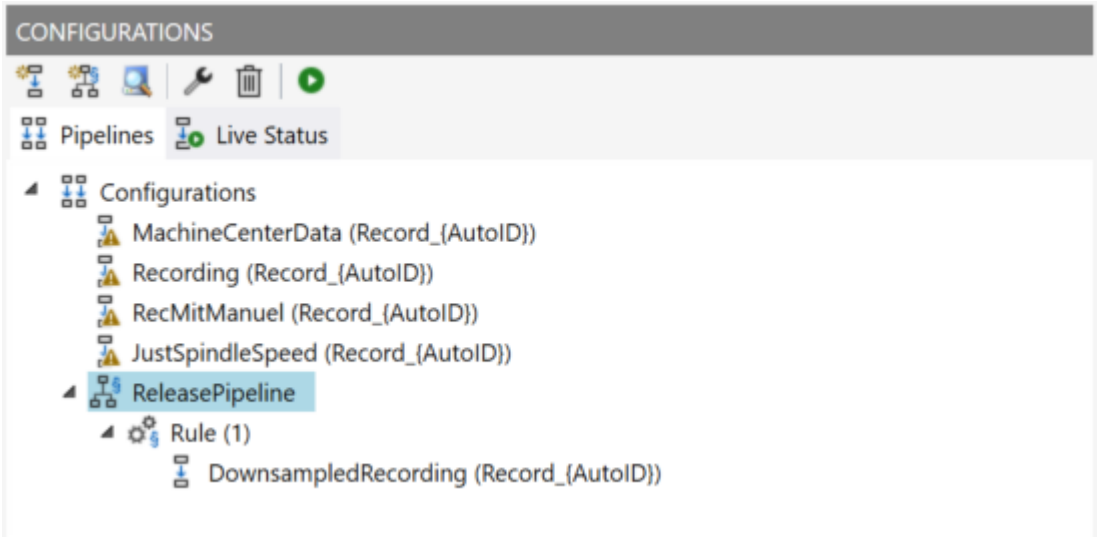
Toolbar Manager window (“CONFIGURATIONS”)



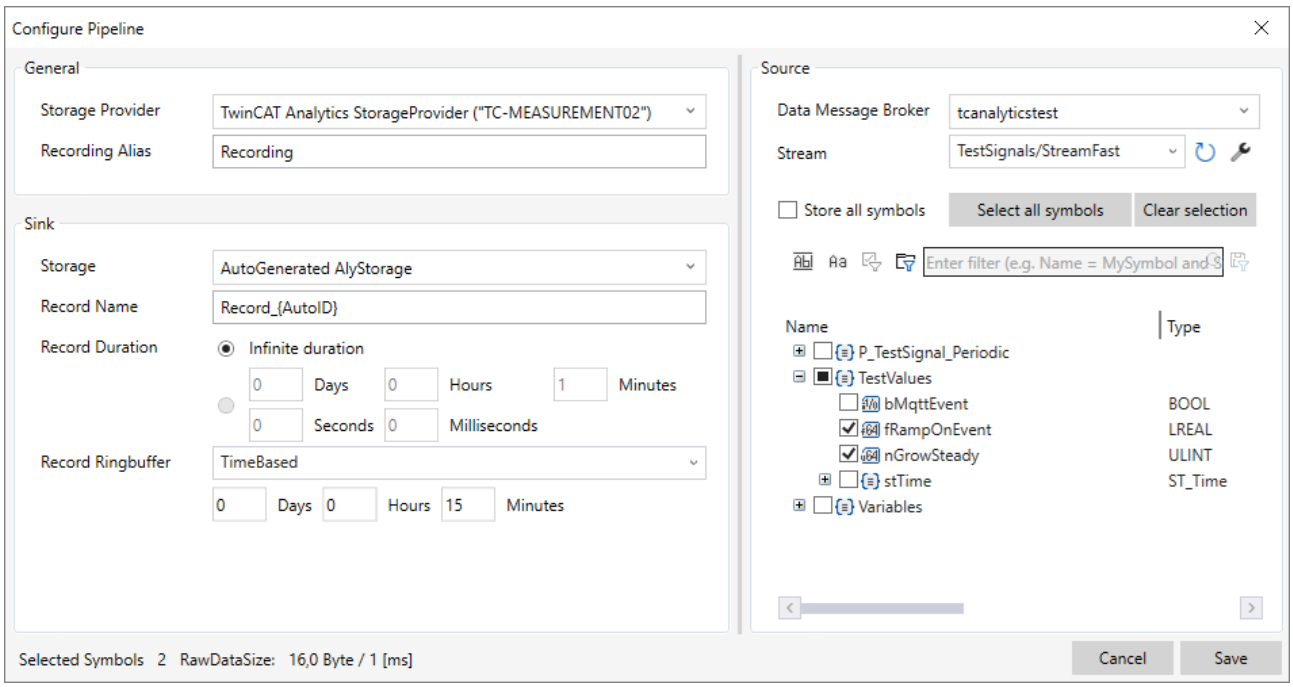
1	Create a new pipeline.
2	Create a new pipeline with Rule Engine.
3	Open Target Browser for connecting simple pipelines.
4	Edit a selected pipeline.
5	Delete a selected pipeline.
6	Start a selected pipeline.

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. From here, you can define new pipelines and edit, delete, or start existing ones.



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



First, select the desired **Storage Provider** and assign a meaningful **Recording Alias**.

You can select the variables to be recorded in the right-hand area under **Source**. Use the **Select all symbols** button to specifically add all available symbols, including all sub-symbols, to the configuration. If a filter is active, only the filtered symbols are applied. If the symbols change while a recording is in progress and one of the selected symbols is no longer included, the recording stops and no further data is recorded. Alternatively, you can use the **Store all symbols** option. This ensures that all symbols available on the stream are always recorded automatically, regardless of the currently available symbol configuration. In this case, manual selection is not required. If symbols are added to or removed from the stream while a recording is in progress, a new recording is created and the data continues to be recorded.

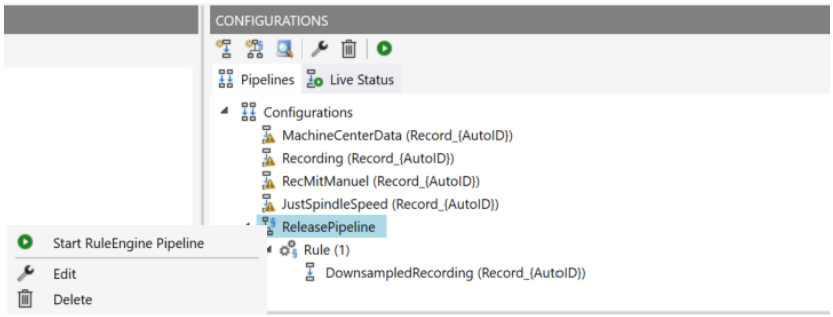
In the **Sink** area, you have the option of defining settings for data storage. Here you can select the desired **Storage** and the **Record Name**. Various placeholders are available for the Record Name, which you can use as required:

"{AutoID}"	Automatically incrementing index
"{Topic}"	Topic of the stream to be recorded.
"{SystemID}"	System ID of the stream to be recorded.
"{Layout}"	Layout of the stream to be recorded.
"{CycleTime}"	Cycle time of the stream to be recorded.
"{SampleSize}"	Data size of a single sample in bytes.
"{RecordStart}"	Start time of the recording

You can use **Record Duration** to set the recording duration. Otherwise, the recording will continue indefinitely until it is stopped manually. In addition, you can define a ring buffer for recording in the **Record Ring buffer** area. The buffer size can be specified based on a time interval or a memory size. The latter mode is not available for all storage types.

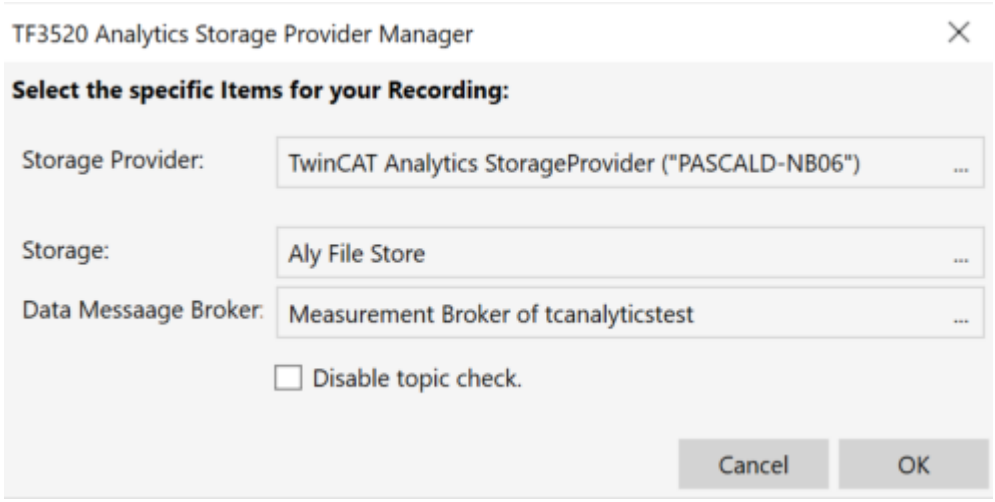
The entries are confirmed with **Save** and a new local recording definition is created. If you have already saved the configuration, you can save a copy using the context menu of the **Save** button **Save as Copy** .

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



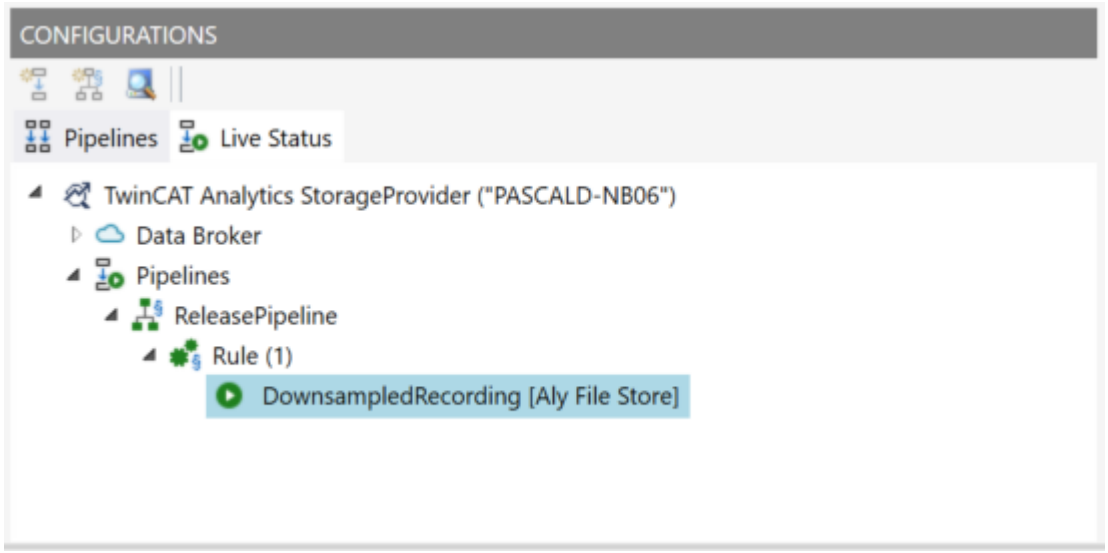
The definition can also be made globally accessible. This is done via the context menu using the **Publish Pipeline** option.

The following dialog will open:



You can now select the desired Analytics Storage Provider through which the definition will 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 way, any Storage Provider Manager connected to the MQTT Broker can find it.

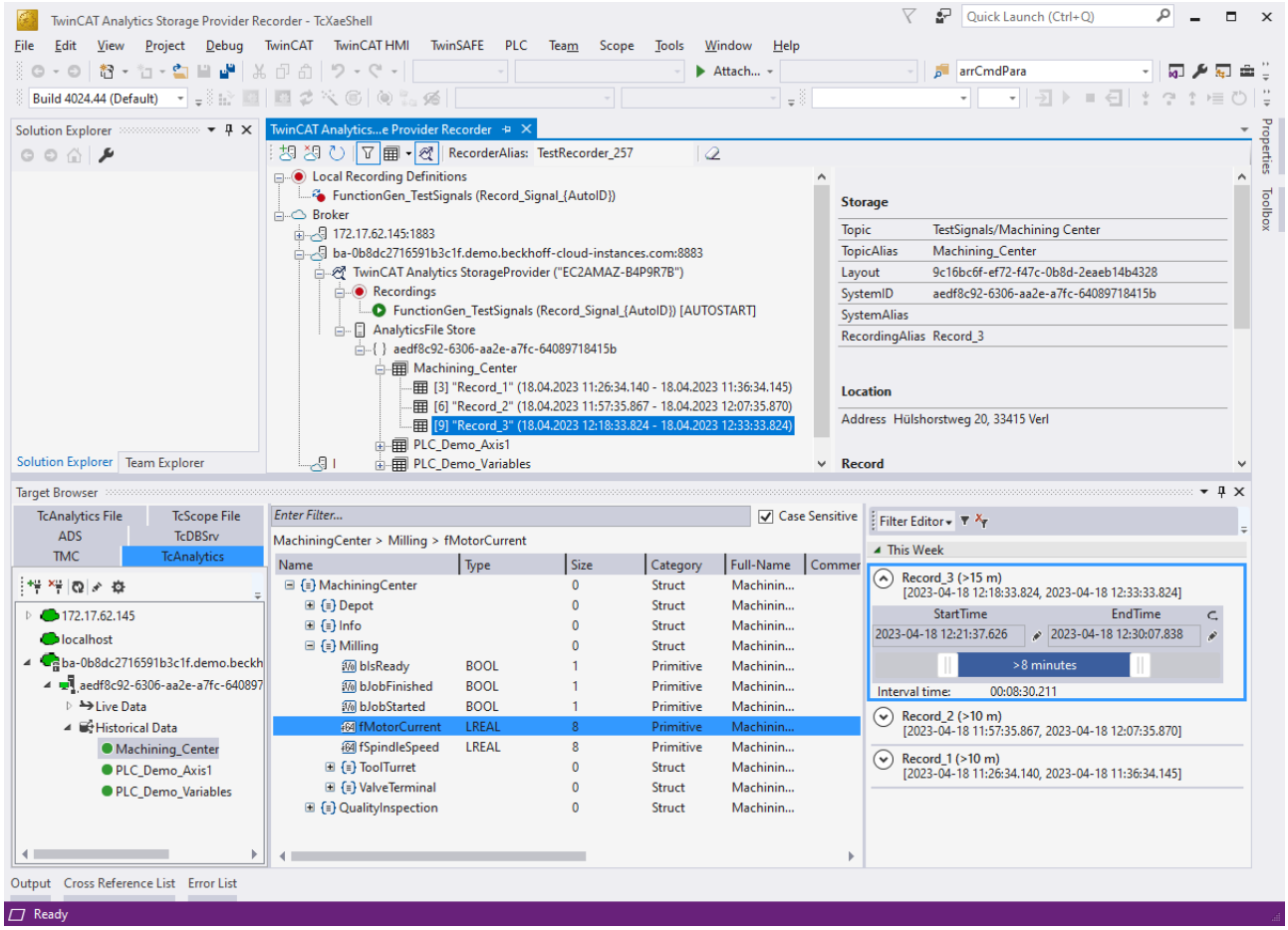
Once a pipeline has been started, the view automatically switches to the second tab, Live Status.



All active pipelines from all users are listed here. You can stop the pipelines in this tab, and you can also 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.

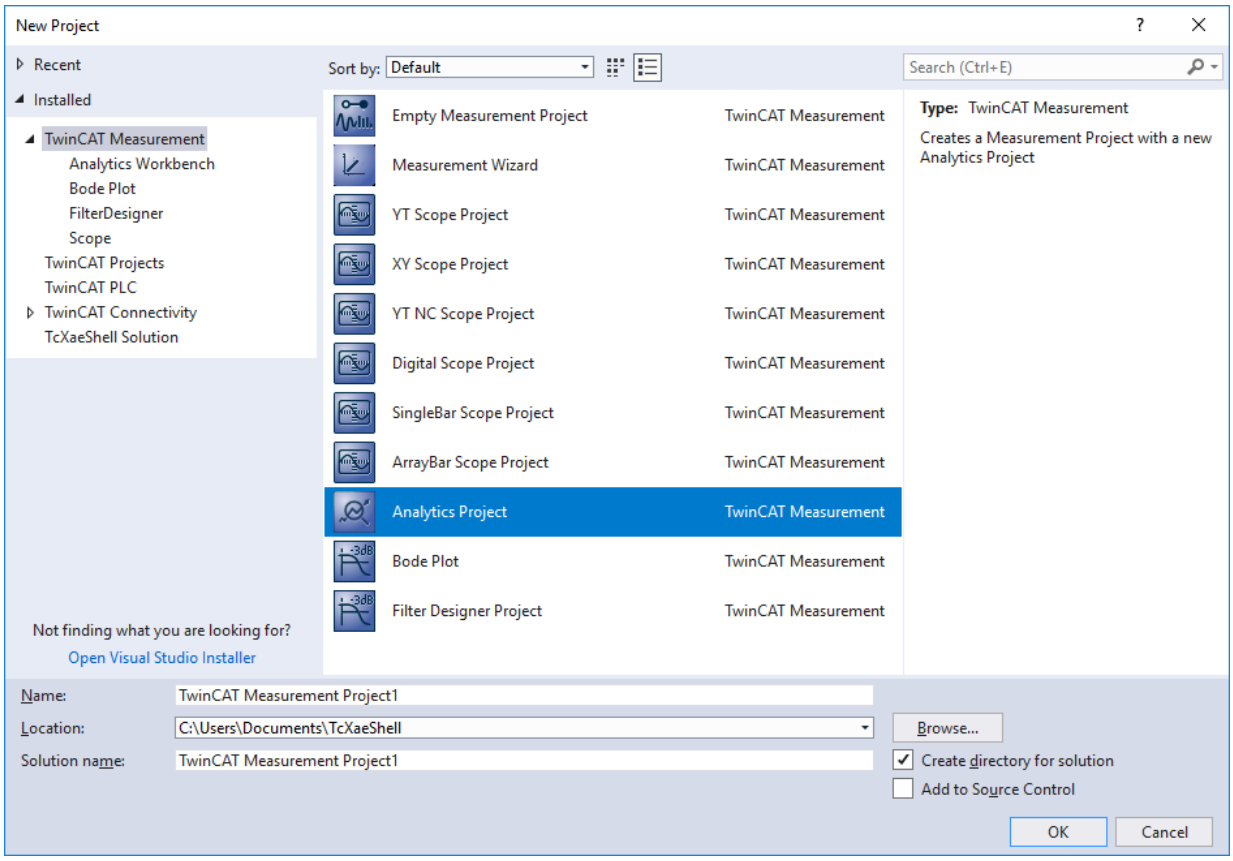


4.4 Analyse data

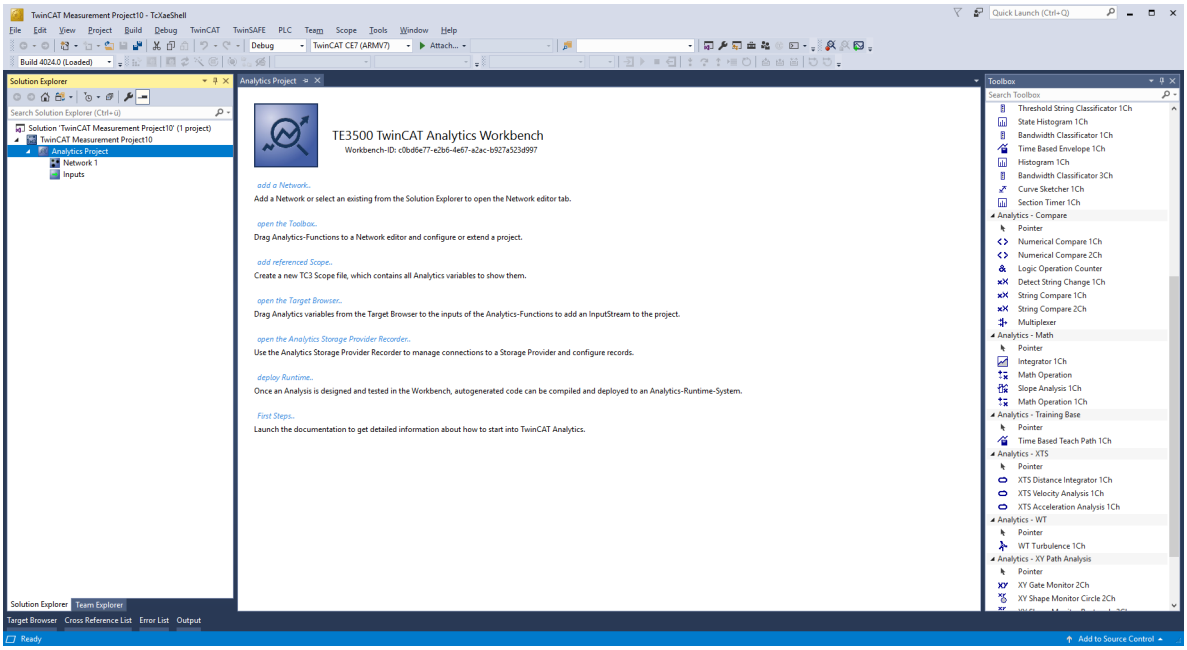
✓ Open your TwinCAT Engineering environment to start the data analysis.

1. Open **Visual Studio > File > New > Project…**

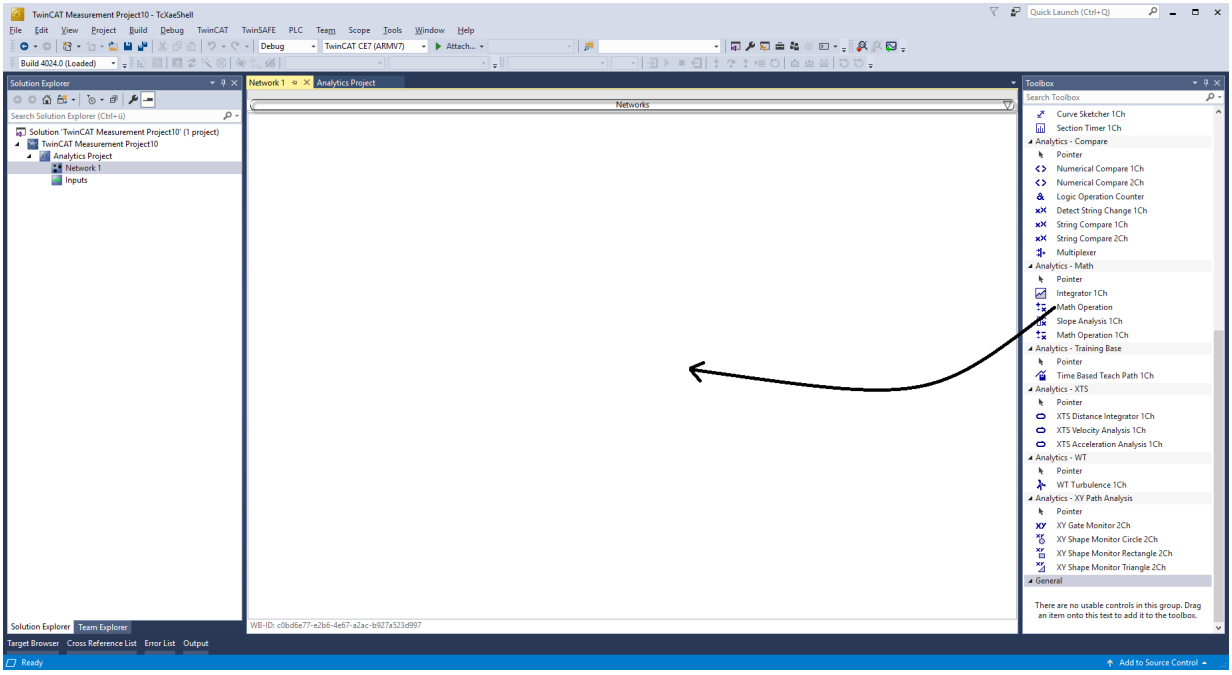
2. Select the **Analytics project template** from **TwinCAT Measurement**.



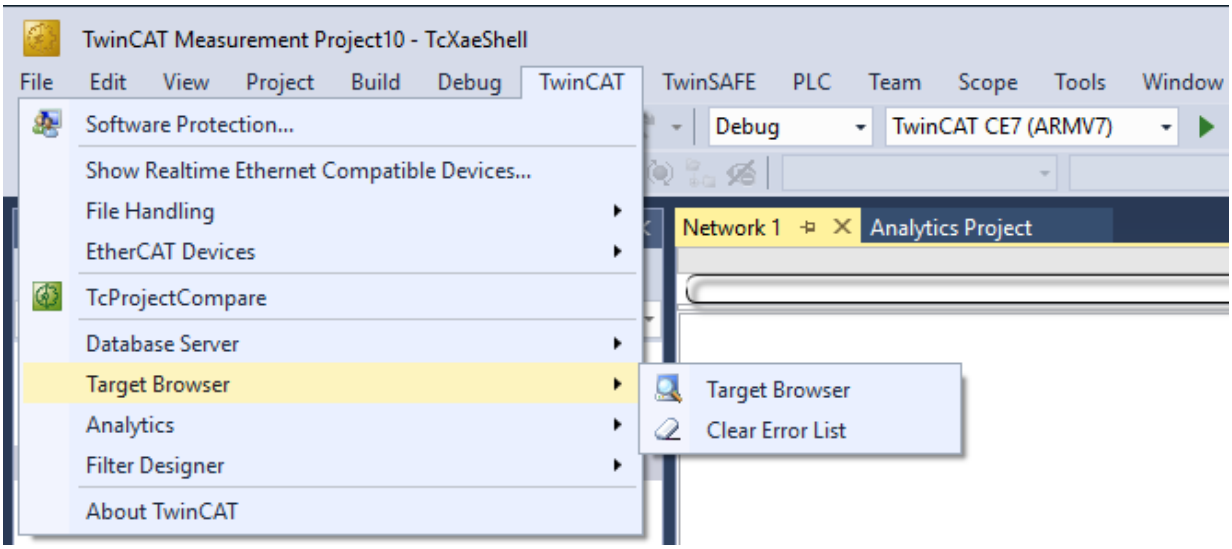
⇨ The new project is displayed in the Solution Explorer. After clicking the **Analytics Project** tree node element a start window opens where you can select your first action. From here you can add a network, open the **Toolbox**, open the **Target Browser** or open the **Analytics Storage Provider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

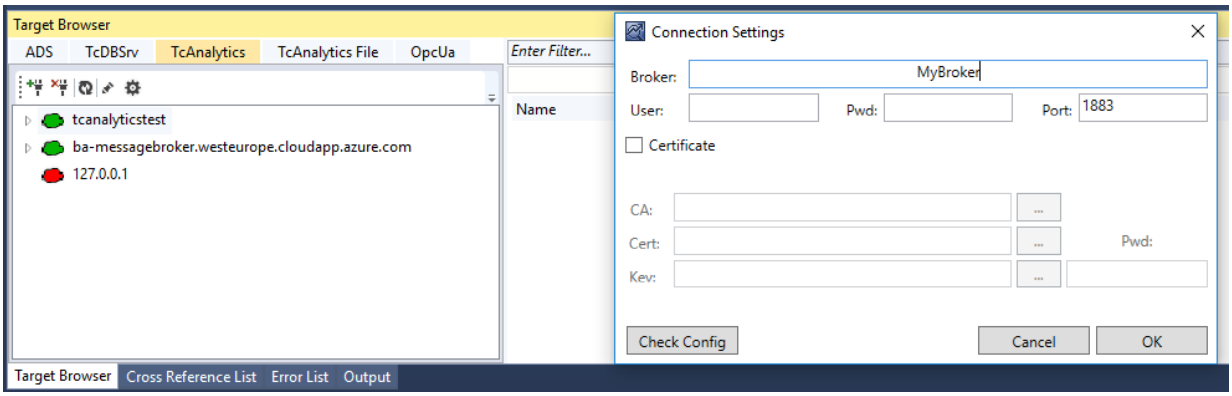


4. When you click on the network, an editor opens. Now you can drag and drop the desired algorithm into the editor interface.
5. After selecting the algorithm, you need to connect input variables to the modules (algorithm). To do this, open the **Target Browser**.
   TwinCAT > Target Browser > Target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

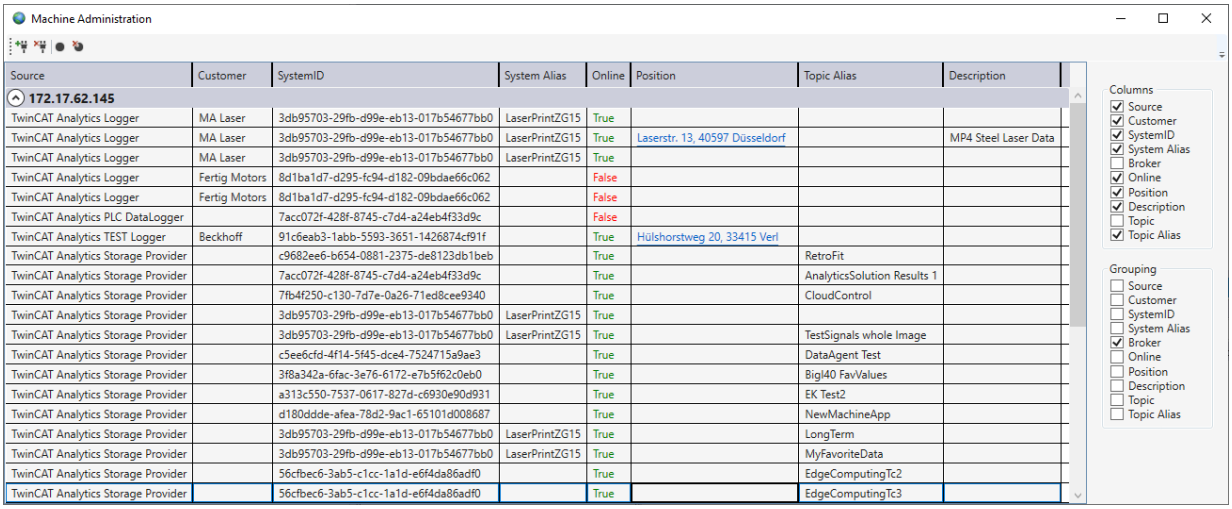


6. Now select the **TcAnalytics** or **TcAnalyticsFile** tab in the Target Browser. Continue with the tab **TcAnalytics** (MQTT).

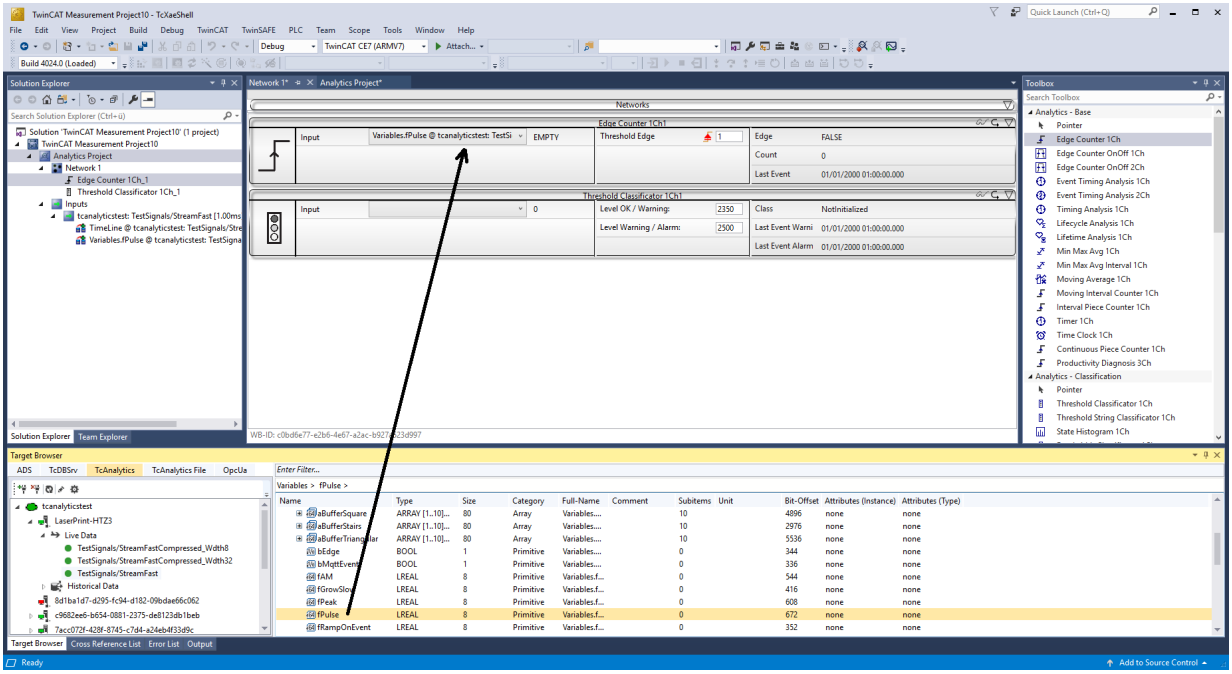
7. Click the icon highlighted in green in the toolbar of this Analytics extension. A window opens in which you can specify the connectivity data of your message broker.



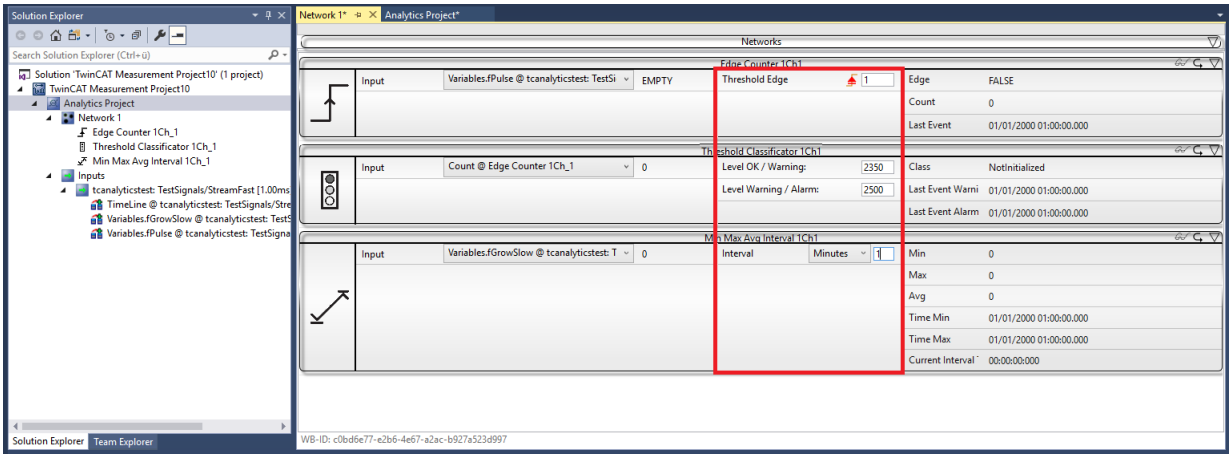
8. Select your MQTT Analytics client (TwinCAT Analytics Logger, TwinCAT IoT Data Agent or Beckhoff EK9160). There is a unique ID for each control. This ID is displayed in the Target Browser.
9. Clicking on the **gear icon**, you will get to the Machine Administration page. Here you can assign a system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



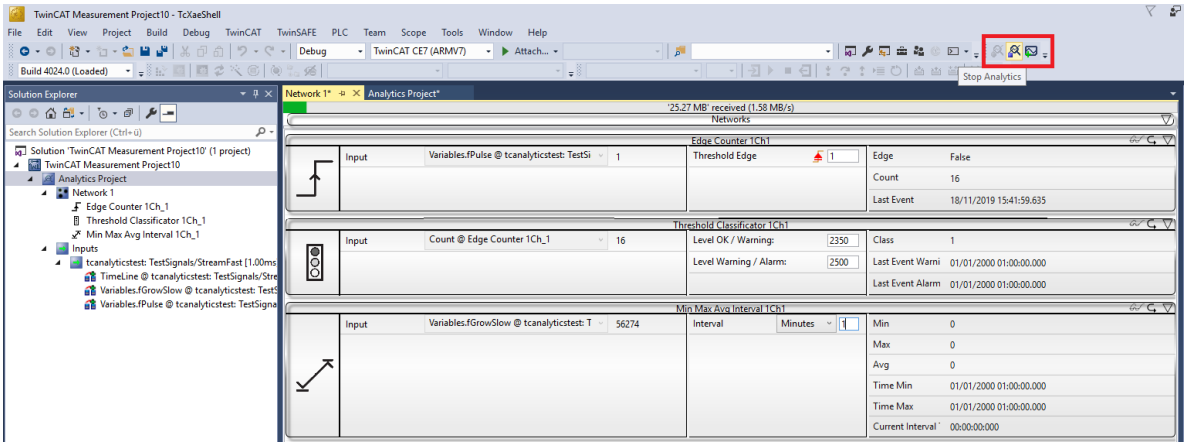
10. In the next step, you can choose between live data and historical data for each MQTT Analytics client. In this case, the historical data is provided by the TwinCAT Analytics Storage Provider.



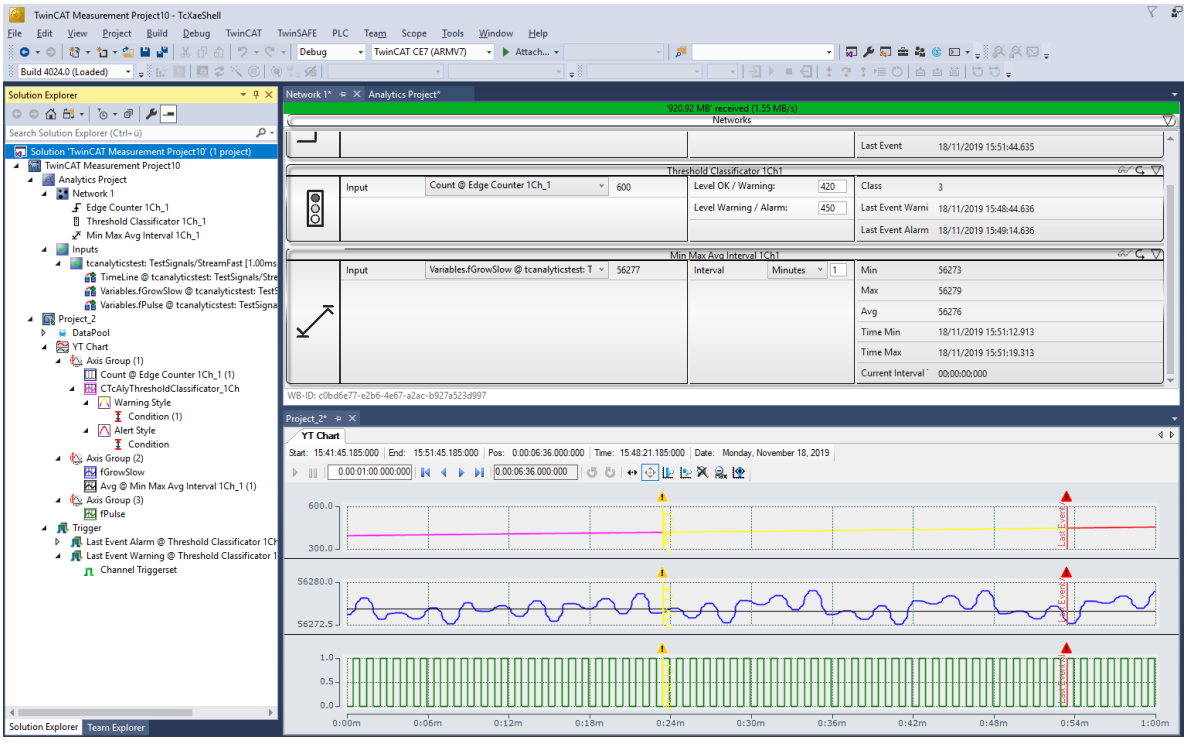
11. You can drag and drop the variables into the inputs of the specific algorithm. In most algorithms, conditions such as thresholds, time intervals, logical operators etc. can be specified. These settings are made in the middle of each module.



⇨ 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.

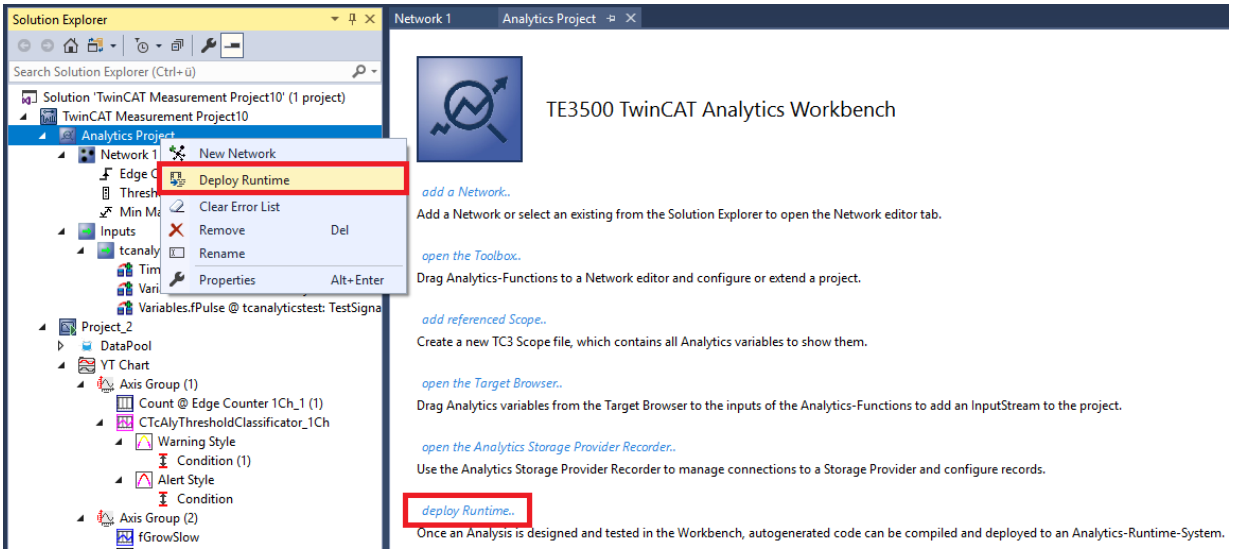


⇨ The analysis results can be displayed in the Scope View graphs using drag-and-drop. For example, a mean value can be displayed as a new channel in the view. Timestamps as markers on the X-axes show significant values.

4.5 24h Analytics application

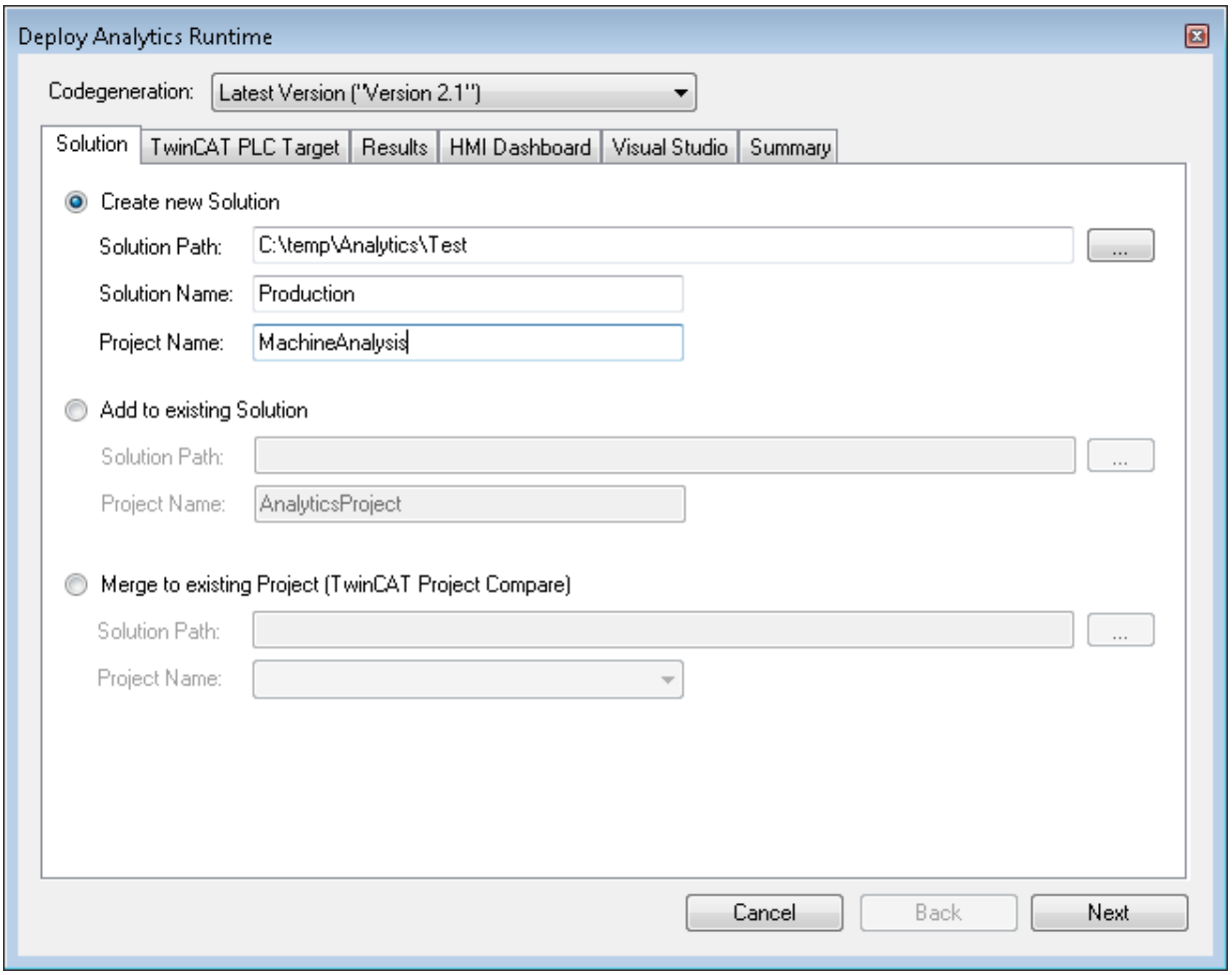
The last major step in the TwinCAT Analytics workflow is the continuous 24-hour machine analysis. It runs in parallel with the machine applications in the field. To make this very easy, the TwinCAT Analytics Workbench can automatically generate PLC code and an HTML5-based dashboard of your Analytics configuration. Both can be downloaded into a TwinCAT Analytics Runtime (TwinCAT3 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.

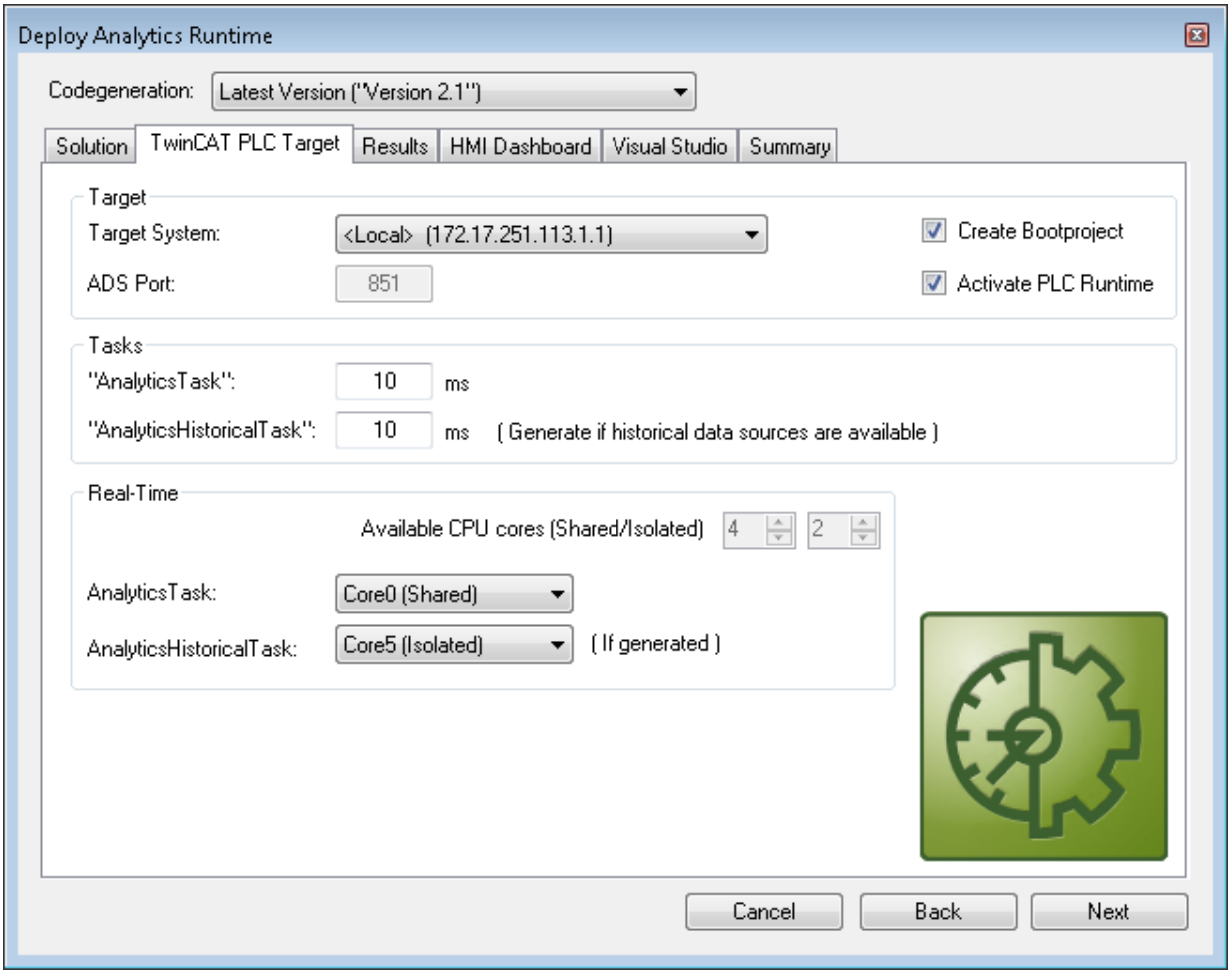


1. When the wizard is open, you can click through some tabs. The first one is called Solution. Here you can decide how your Analytics project should be used in the PLC code: As...
completely new solution.

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

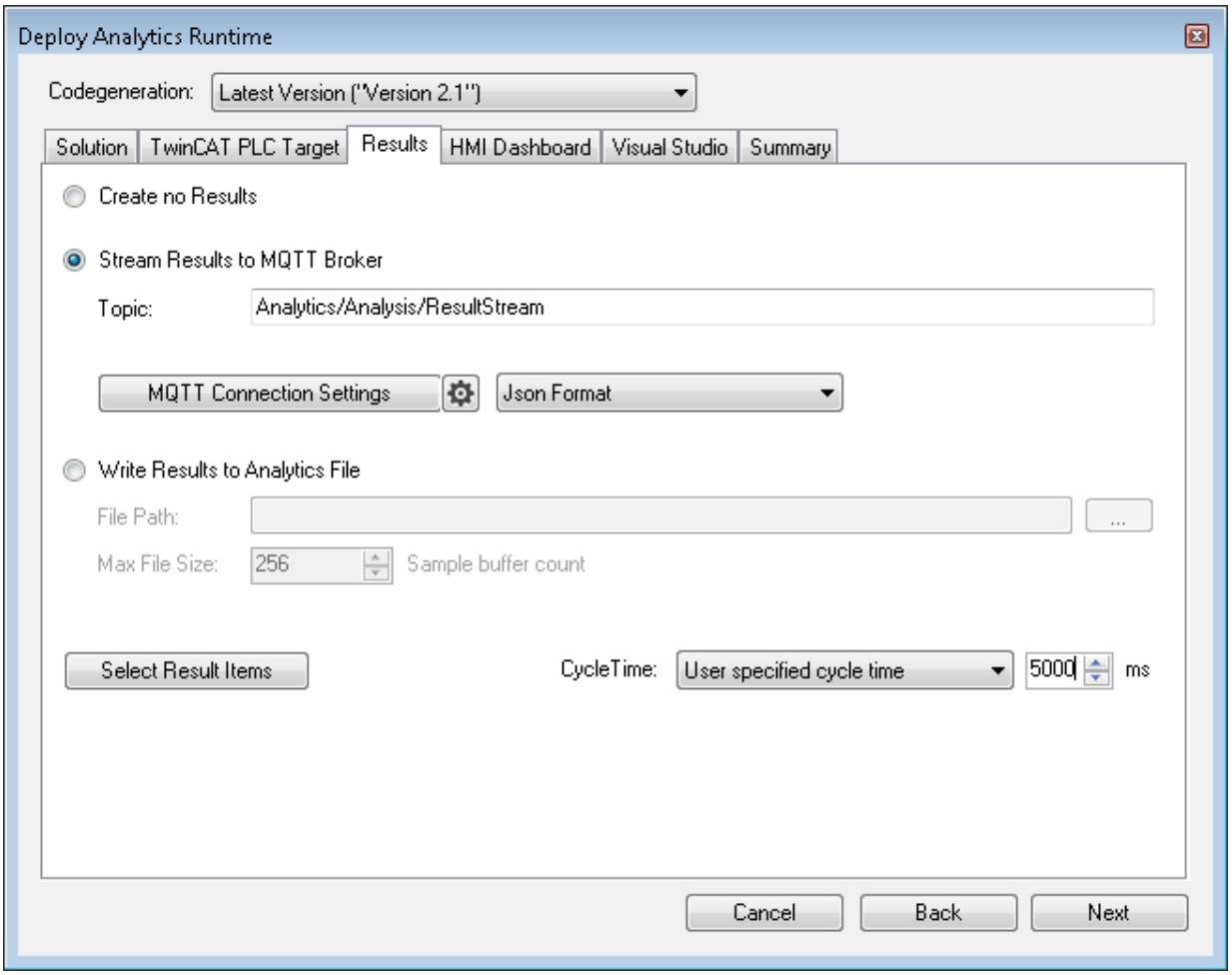


2. In the **TwinCAT PLC Target** tab you can select the ADS target system that runs the TwinCAT Analytics Runtime (TF3550). The created project is immediately executable. For this purpose you can set the Activate PLC Runtime option. In addition, it can be selected that a boot project is created directly.

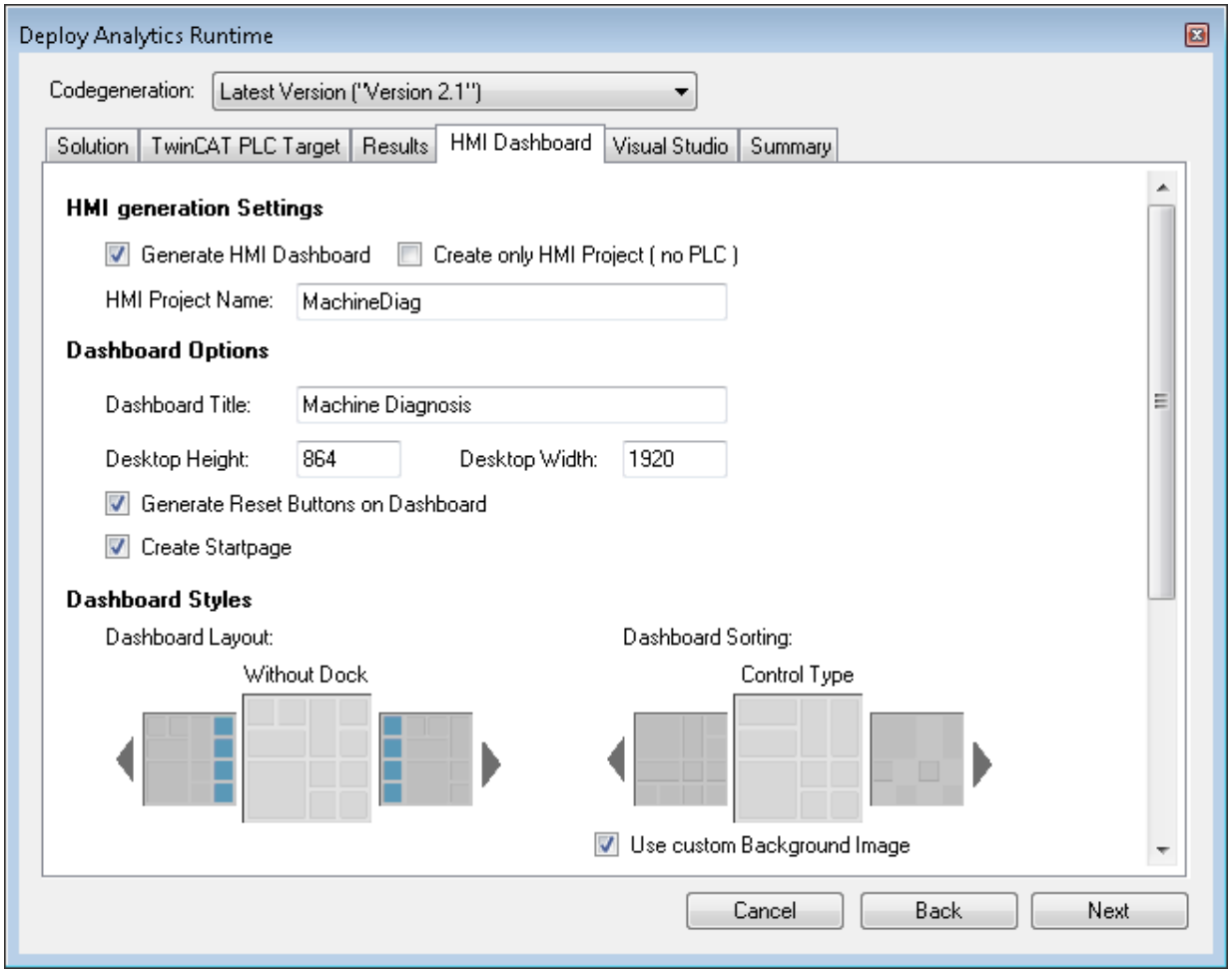


3. Especially for virtual machines, it is important to run the project on isolated cores, which is also an option in this tab. The next tab **Results** is needed only if you have selected the **Stream Results** option in the algorithm properties. If you want to send results, you can decide here in which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

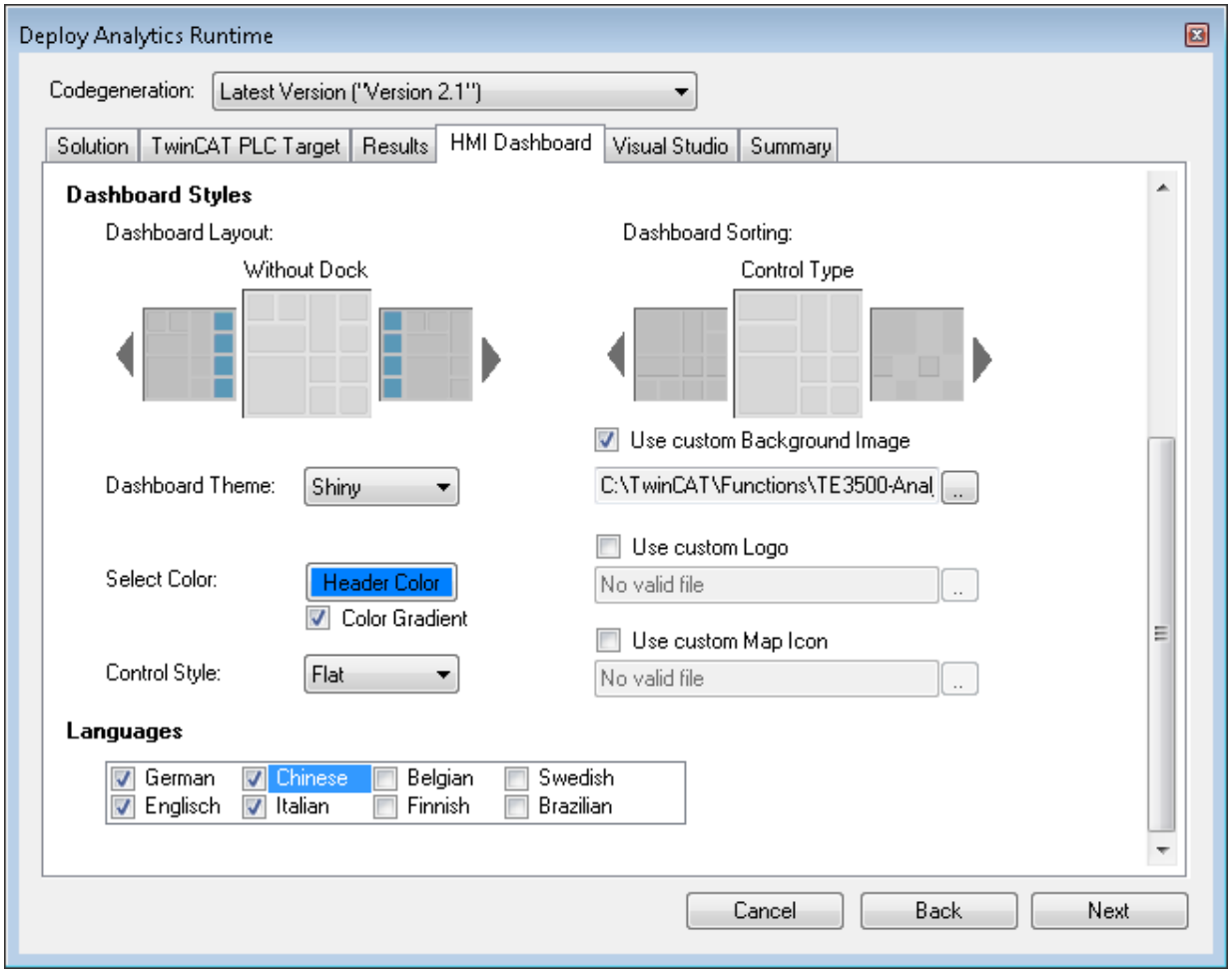
automatically and executed immediately after activation.
Downsampling of the results is possible by specifying a cycle time.



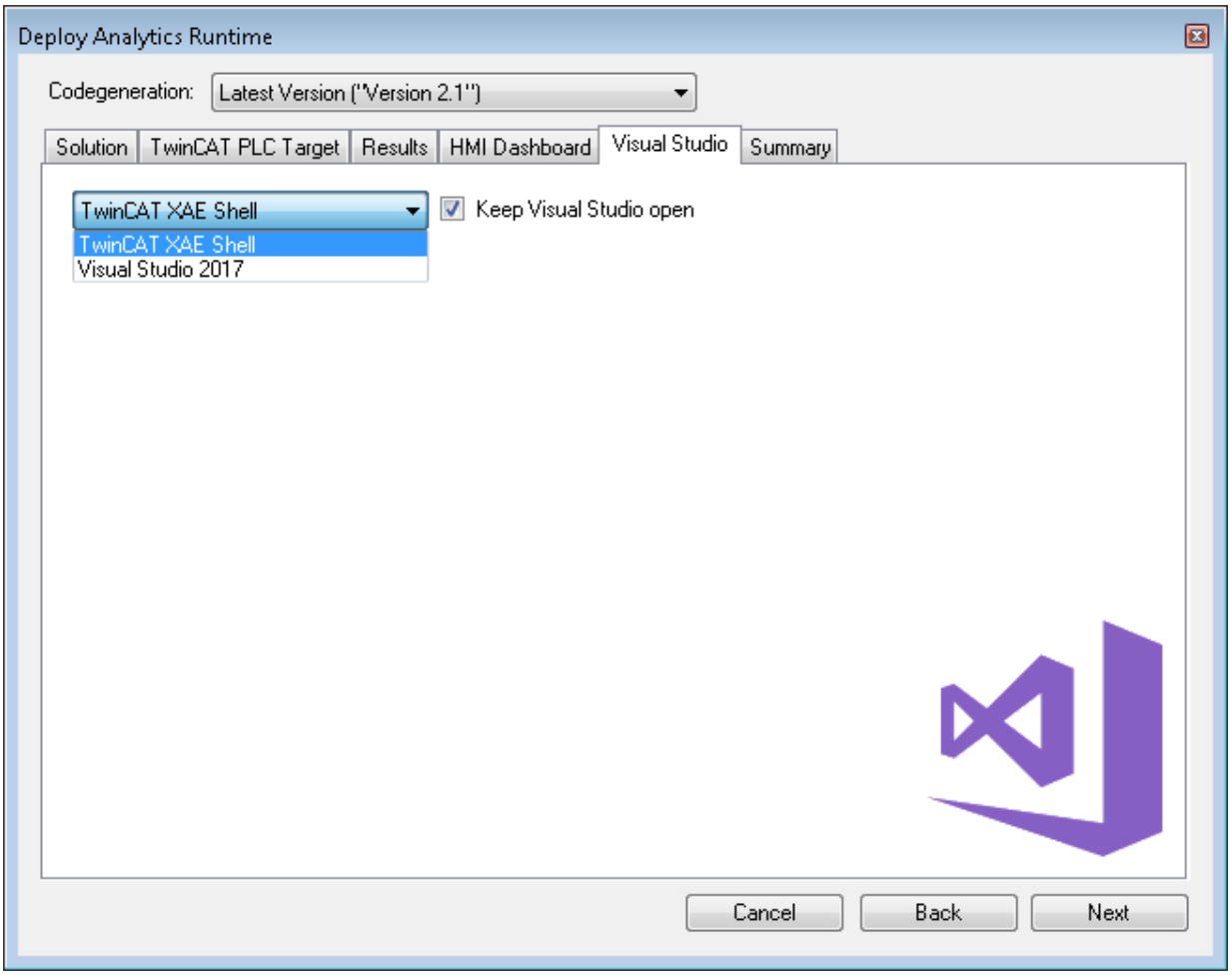
4. The next tab is for the **HMI Dashboard**. A prerequisite for the automatic generation of the dashboard is the selection of HMI Controls for the corresponding algorithms whose results are to be displayed in the dashboard.



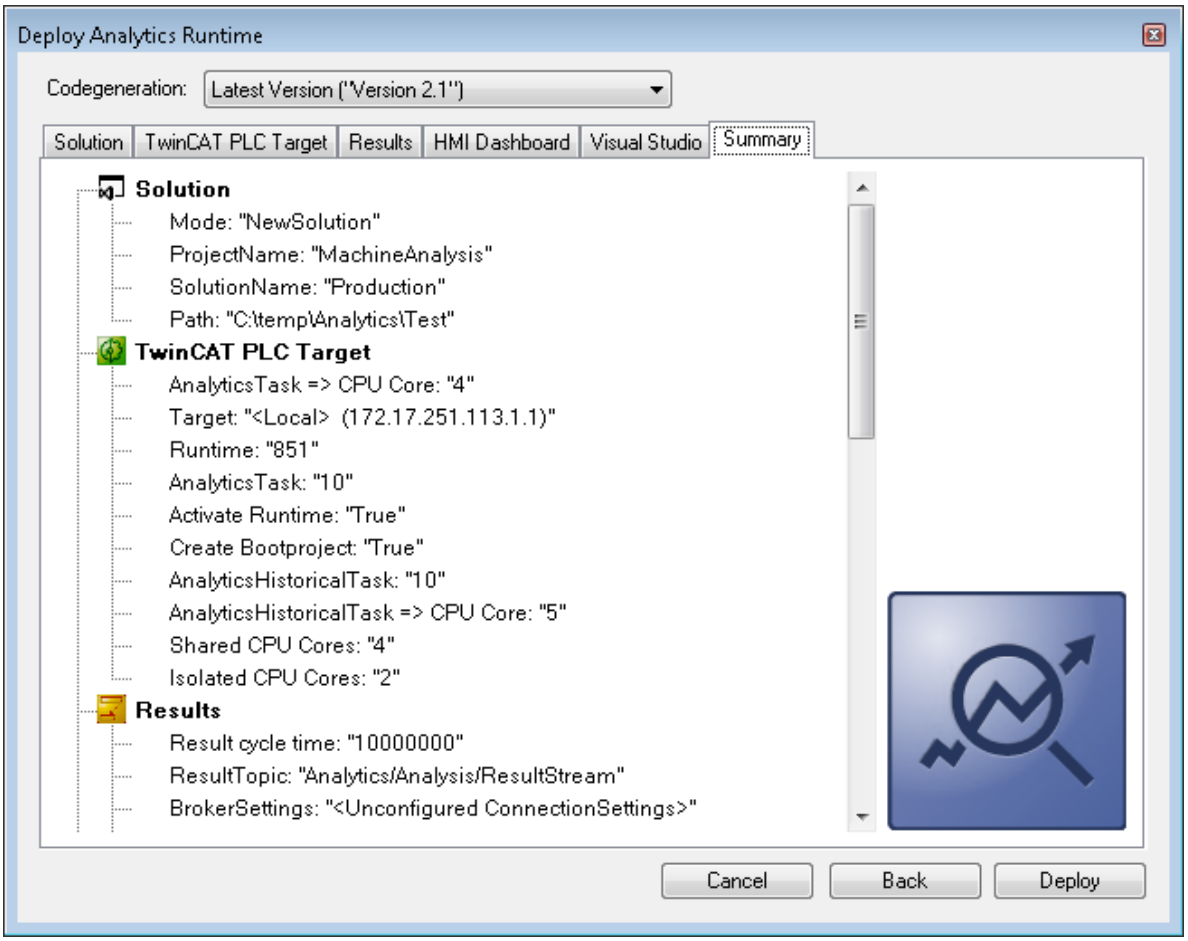
5. You can choose different options for your Analytics Dashboard, such as a start page with a map, layouts, sorting algorithms, custom colors and logos. If you select multiple languages for the Analytics Controls, a language switching menu will also be generated.



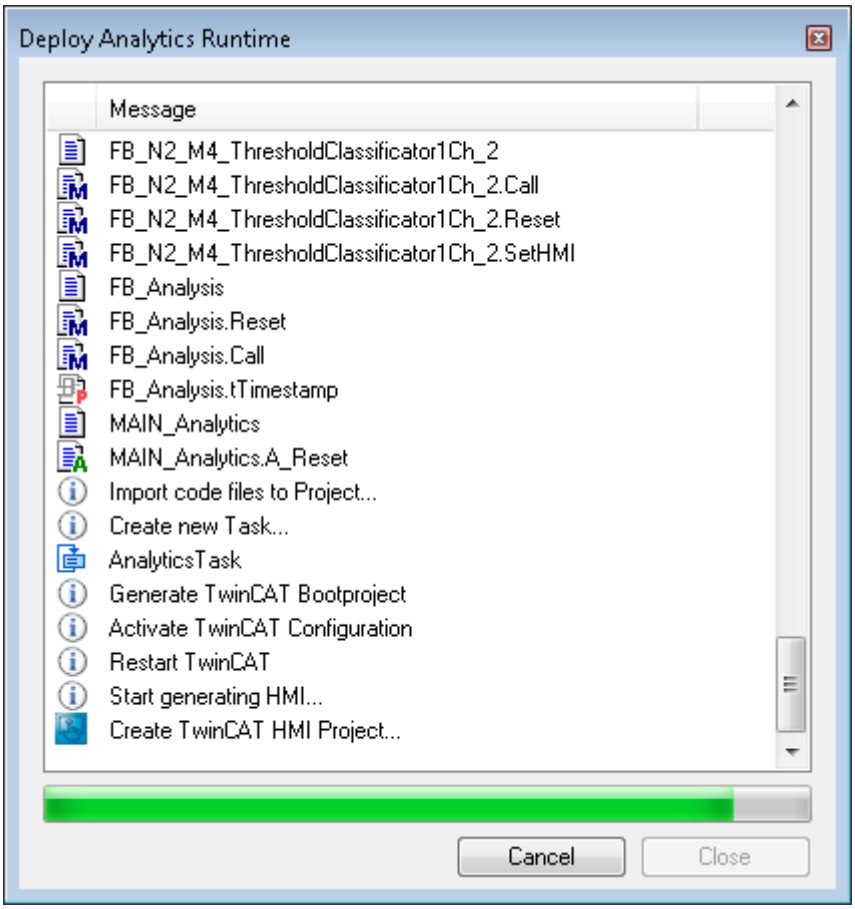
6. Select one of the installed versions of Visual Studio and, whether the instance should start visibly or just be set up and activated in the background.

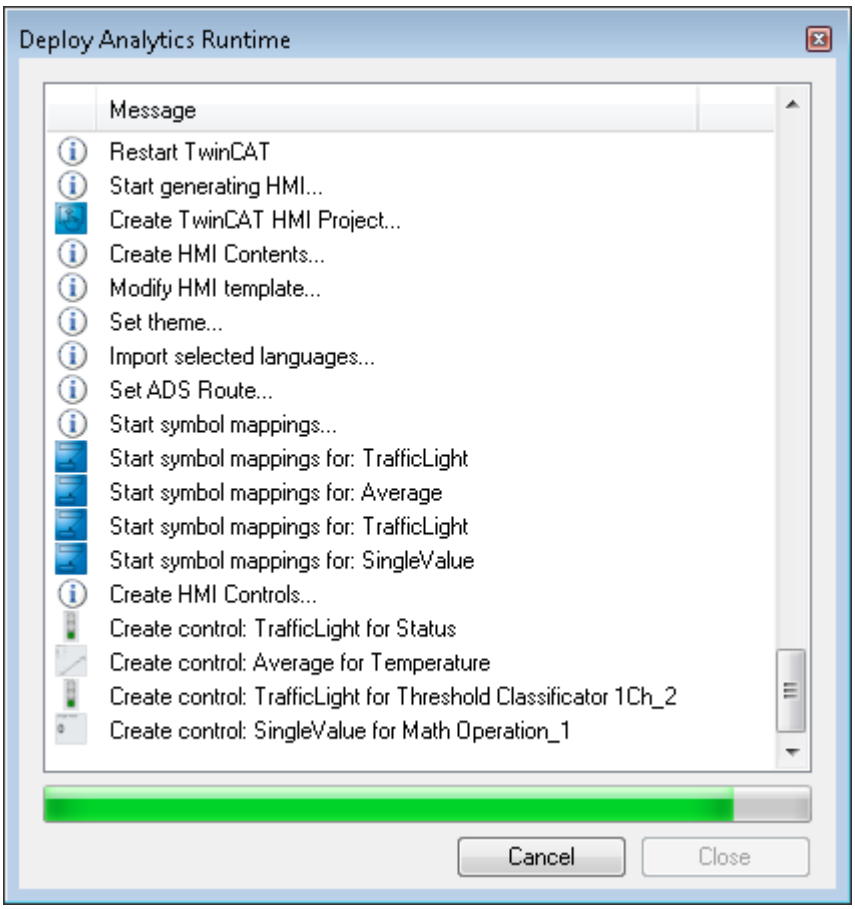


⇨ At last you can find an overview.

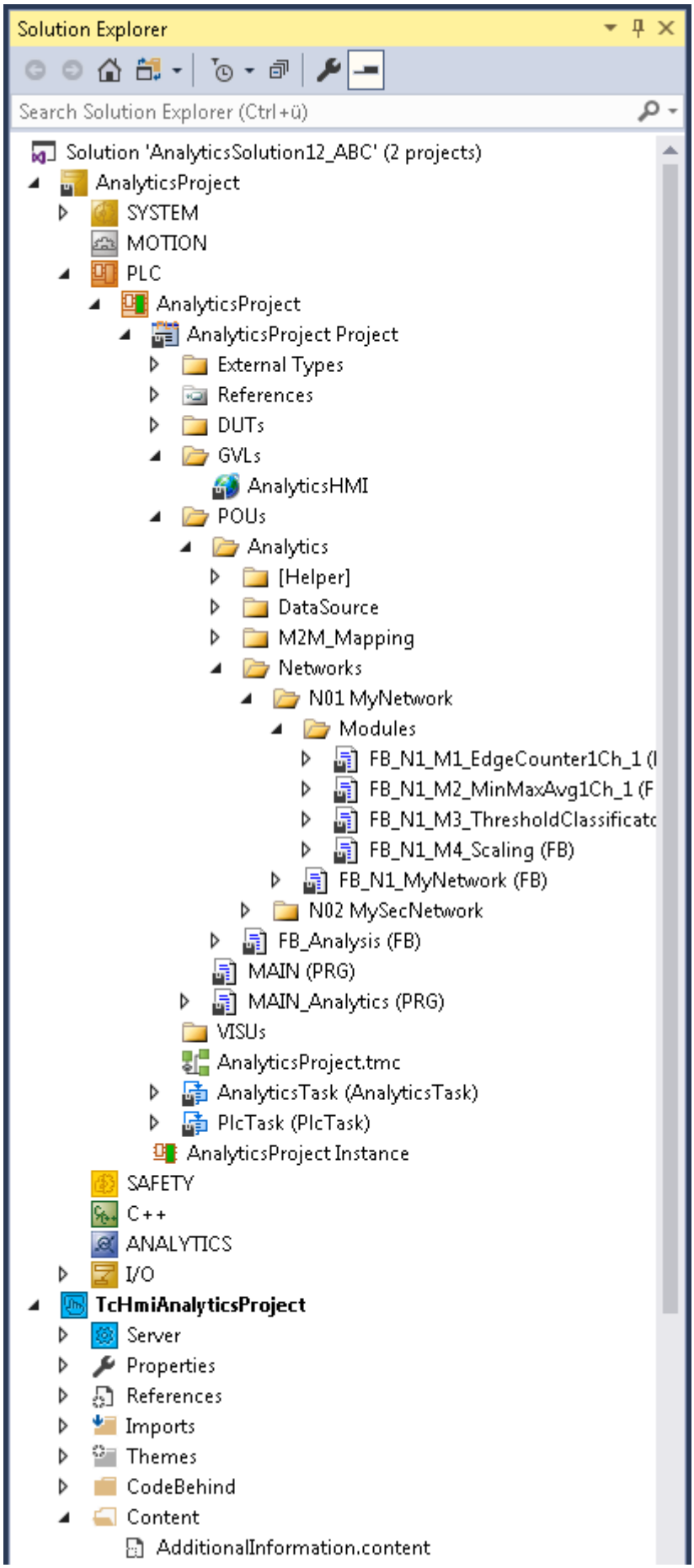


7. Now you can click the **Deploy** button to start the generation process. The PLC project and the HMI dashboard are now generated.





⇨ 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

5.1 Basic Concepts

Variables and Datatypes

There are several types of variables that can be logged. Variables that are part of:

- PLC or NC process images
- PLC programs
- process images of devices e.g. an EtherCAT master and
- data areas of generic TcCom objects.

Moreover, they may be of any datatype defined in IEC 61131 or the C++ standard in case of generic TcCom-objects.

Structured datatypes may recursively contain other structured datatypes and may be logged as a whole or partially. For more information on this topic, refer to the Configuration section.

Operation modes

This documentation makes use of the terminology and concepts described in the MQTT section. Please refer to that section for general information about the protocol.

A key concept of TwinCAT Analytics are streams, which form the basic unit of transaction between a data source and a destination. One Data Logger can control several streams.

A stream may comprise four components:

- stream description
- stream Tx-description
- symbol info
- and stream data.

In **MQTT-mode**, Analytics participants coordinate by means of the first three components using an MQTT broker and a specific MQTT topic for each component. The **topics** are generally structured as illustrated in the following table, where <> brackets indicate variables as opposed to the other parts that are fixed.

Component	Topic structure	Format	Purpose
Description	<MainTopic>/<StreamTopic>/Descr	JSON	Provides information on whether a stream source is online or offline. Includes a timestamp for the information.
Tx description	<MainTopic>/<StreamTopic>/Bin/Tx/Descr	JSON	Provides information about the transmission parameters when a stream source is actively transmitting data.
Symbol info	<MainTopic>/<StreamTopic>/Bin/Tx/Symbols	Binary	Contains meta information about the variables i.e. excluding the actual value.
Data	<MainTopic>/<StreamTopic>/Bin/Tx/Data	Binary	Contains the plain variable values.

The main topic is the only subtopic that can freely be set by the user.

A stream can be **started** and **stopped**. A stream can be started by default via the System Manager configuration as soon as TwinCAT starts in Run Mode. Additionally, streams can be started and stopped from PLC code.

Once connected to a broker, the Logger first sends the stream description. Then, as soon as the stream starts, it sends the Tx description and symbol info. This enables recipients to take all measures that are needed before data arrives. Finally, data is sent cyclically.

In **File Mode**, the TwinCAT Boot directory on the target device is used as a base for a dedicated Analytics directory, which in turn contains one subdirectory per stream as soon as the respective TwinCAT project has been activated. A specific .tas file in the stream directory contains the symbol information, while .tay files, which are created cyclically, contain the stream data.

Relation of Logger and Streams

One data logger can control several streams. As will be described more detailed in the Configuration section a user can add one or more data loggers to the Analytics configuration. Streams then are added automatically, depending on what variables are available to be logged. To understand how streams are assigned to a logger it is useful to understand that every stream has specific characteristics. One characteristic, the **cycle time**, comes from the fact that every variable that is acted upon cyclically is subject to a cyclic task; therefore, cycle times that underlie different variables can vary. Since a stream has a fixed cycle time by definition and to decouple tasks with the same cycle time, one stream is created for every task that drives relevant variables. Additionally, it is purposeful to further divide streams depending on the **stream source**, meaning the origin PLC instance or TcCom object. This enables users to send variables of different sources to different MQTT topics and start/stop the transmission independently. Eventually it all comes down to the following scheme:



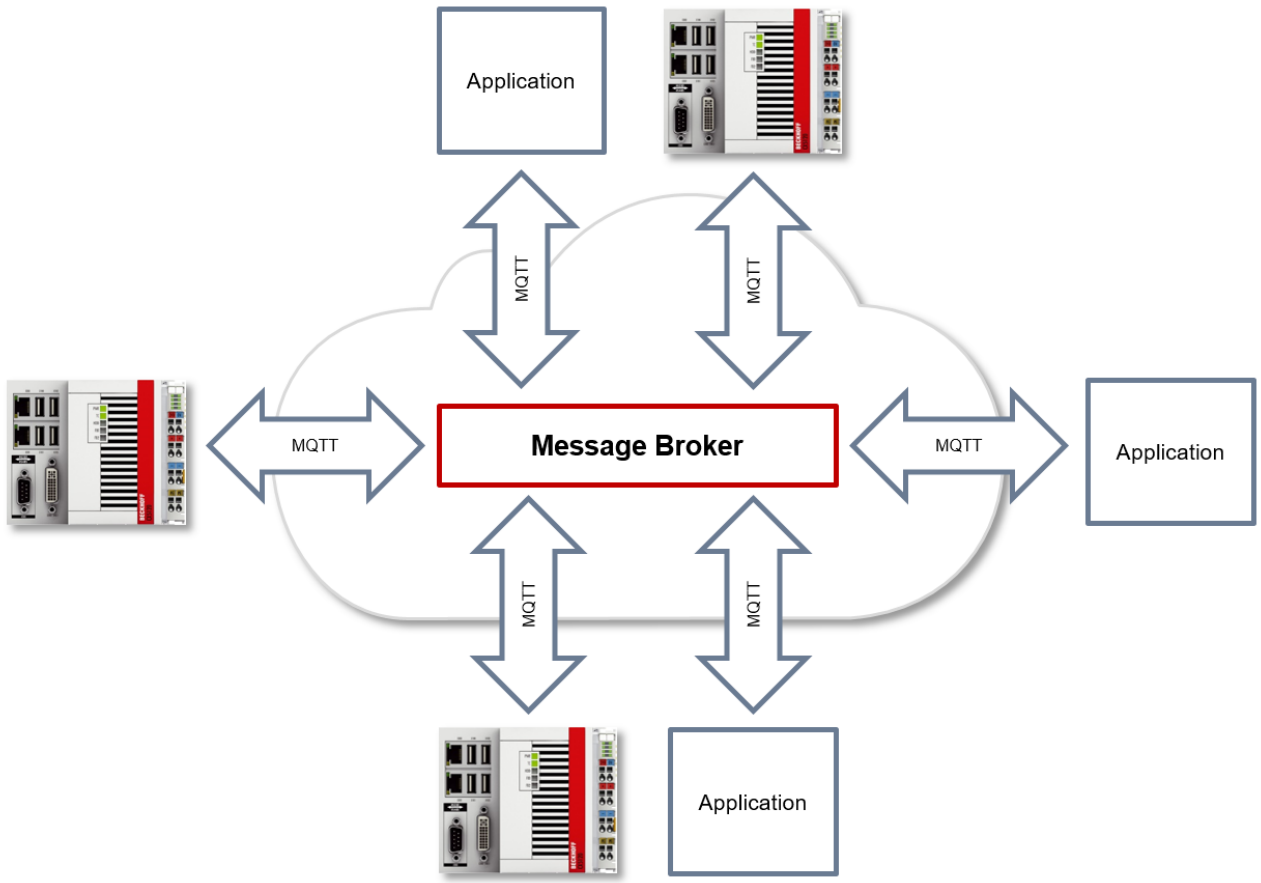
For every stream source and every task that drives variables of that source, a stream is created.

When configuring the data logger, there are configuration parameters that all streams have in common and those that are stream specific. The logger specifies the destination, the compression method, MQTT credentials etc. whereas e.g. the data size and start/stop-functionality are stream specific.

In most cases it might not be necessary to send or write data every task cycle, so the data of a stream recorded in a cycle, a so-called **sample**, can be buffered before sending it to the broker or writing it to file. The number of samples in a buffer and thus the cycle time of a **buffer** being sent/written can be configured. Moreover, the number of buffers in a file and thus the **file size** can be configured.

5.2 MQTT basics

MQTT (Message Queueing Telemetry Transport) is a publisher/subscriber-based communication protocol which enables message-based transfer between applications. The message broker is a central component of this transfer type. It distributes messages between the individual applications or the sender and receiver of a message. The message broker decouples the sender and receiver, so that it is not necessary for the sender and receiver to know and exchange each other's address information. During sending and receiving, all communication devices contact the message broker, which handles the distribution of the messages.



ClientID

When establishing a connection with the message broker, the client transmits a ClientID, which is used to uniquely identify the client on the message broker. The MQTT communication driver from TwinCAT 3 automatically generates its own ClientID, which is based on the following naming scheme:

PlcProjectName-TcMqttClient%n

%n is an incremental counter for the number of the respective MQTT client instance. Each instance of the FB_IotMqttClient function block increments this counter. In most cases, using this ClientID format is sufficient. In special cases, e.g. depending on the message broker or also due to the own MQTT application, an application-specific ClientID must be assigned. This can be done via a corresponding input at the FB_IotMqttClient and FB_IotMqtt5Client function blocks.

If a unique ClientID is to be generated automatically at the start of the PLC project, the use of a GUID is recommended, which can be generated via the FB_CreateGuid function block from the Tc2_System library. The following sample code illustrates the use of this function block.

```
PROGRAM MAIN
VAR
  fbGuid : FB_CreateGUID;
  objGuid : GUID;
  sGuid : STRING;
  nState : UINT;
  bStart : BOOL; // set to TRUE to start this sample
END_VAR

CASE nState OF
  0 :
    IF bStart THEN
      bStart := FALSE;
      nState := nState + 1;
    END_IF

  1 : // create GUID using FB_CreateGuid from Tc2_System library
    fbGuid(bExecute := TRUE, pGuidBuffer := ADR(objGuid), nGuidBufferSize := SIZEOF(objGuid));
    IF NOT fbGuid.bBusy THEN
      fbGuid(bExecute := FALSE);
```

```
      IF NOT fbGuid.bError THEN
        nState := nState + 1;
      ELSE
        nState := 255; // go to error state
      END_IF
    END_IF

  2: // GUID has been created, now convert to STRING
    sGuid := GUID_TO_STRING(objGuid);
    nState := nState + 1;

  3: // done

255: // error state

END_CASE
```

After execution of this State Machine, the variable sGuid contains the generated GUID as STRING. This can then be used at the FB_IotMqttClient and FB_IotMqtt5Client function blocks as ClientID.

Payload

The content of an MQTT message is referred to as payload. Data of any type can be transferred, e.g. text, individual numerical values or a whole information structure.

Message payload formatting

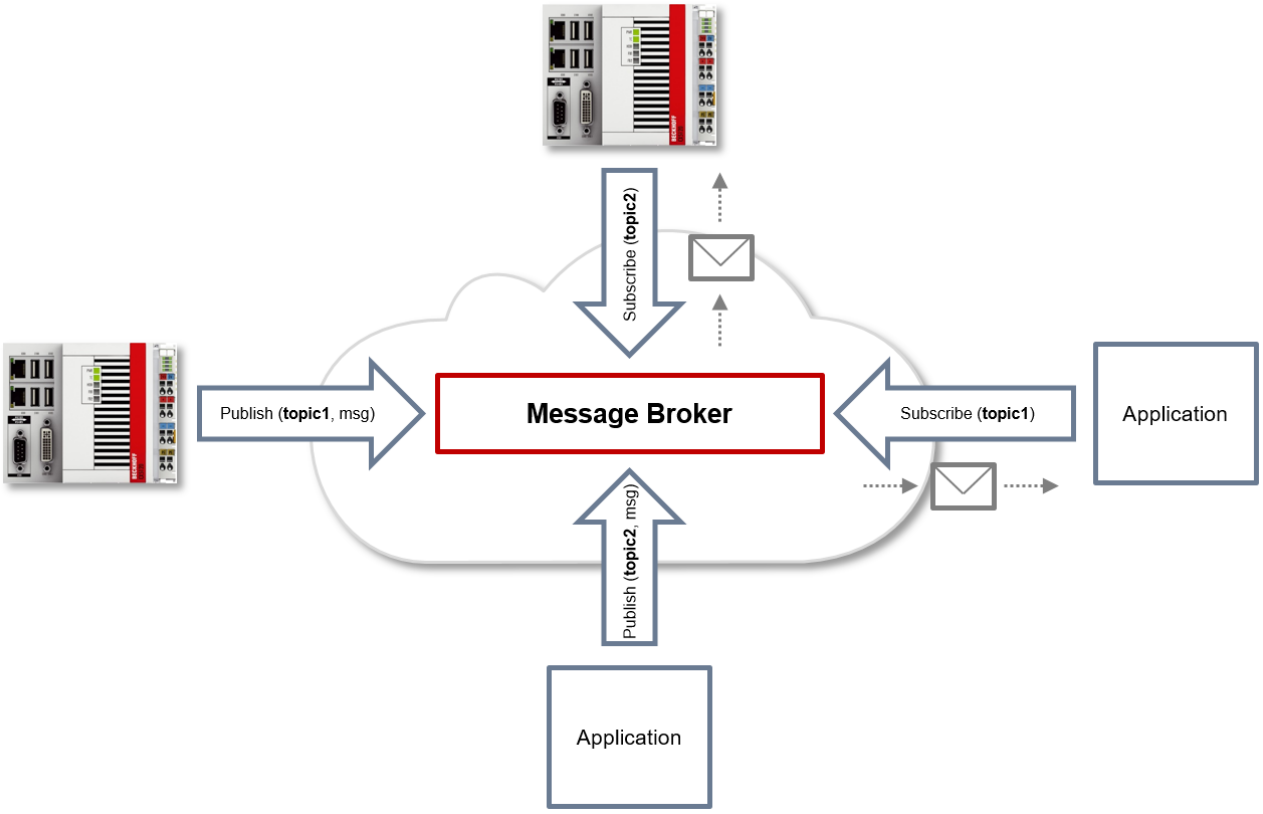
Note that the data type and the formatting of the content must be known to the sender and receiver side, particularly when binary information (alignment) or strings (with or without zero termination) are sent.

Topics

If a message broker is used that is based on the MQTT protocol, sending (publish mode) and subscribing (subscribe mode) of messages is organized with the aid of so-called topics. The message broker filters incoming messages based on these topics for each connected client. A topic may consist of several levels; the individual levels are separated by “/”.

Example: Campus / Building1 / Floor2 / Room3 / Temperature

When a publisher sends a message, it always specifies for which topic it is intended. A subscriber indicates which topic it is interested in. The message broker forwards the message accordingly.



Communication example 1 from the diagram above:

- An application subscribes to “topic1”.
- A controller publishes a message to “topic1”.
- The message broker forwards the message to the application accordingly.

Communication example 2 from the diagram above:

- A controller subscribes to “topic2”.
- An application publishes a message to “topic2”.
- The message broker forwards the message to the controller accordingly.

Wildcards

It is possible to use wildcards in conjunction with topics. A wildcard is used to represent part of the topic. In this case a subscriber may receive messages from several topics. A distinction is made between two types of wildcards:

- Single-level wildcards
- Multi-level wildcards

Example for single-level wildcard:

The + symbol describes a single-level wildcard. If it is used by the subscriber as described below, for example, corresponding messages to the topics are either received by the subscriber or not.

- The receiver subscribes to Campus/Building1/Floor2/**+**/Temperature
- The publisher sends to Campus/Building1/Floor2/Room1/Temperature - OK
- The publisher sends to Campus/Building1/Floor2/Room2/Temperature - OK
- The publisher sends to Campus/Building42/Floor1/Room1/Temperature - NOK
- The publisher sends to Campus/Building1/Floor2/Room1/Fridge/Temperature - NOK

Example for multi-level wildcard:

The # symbol describes a multi-level wildcard. If it is used by the subscriber as described below, for example, corresponding messages to the topics are either received by the subscriber or not. The # symbol must always be the last symbol in a topic string.

- The receiver subscribes to Campus/Building1/Floor2/**#**
- The publisher sends to Campus/Building1/Floor2/Room1/Temperature - OK
- The publisher sends to Campus/Building1/Floor2/Room2/Temperature - OK
- The publisher sends to Campus/Building42/Floor1/Room1/Temperature - NOK
- The publisher sends to Campus/Building1/Floor2/Room1/Fridge/Temperature - OK
- The publisher sends to Campus/Building1/Floor2/Room1/Humidity - OK

QoS (Quality of Service)

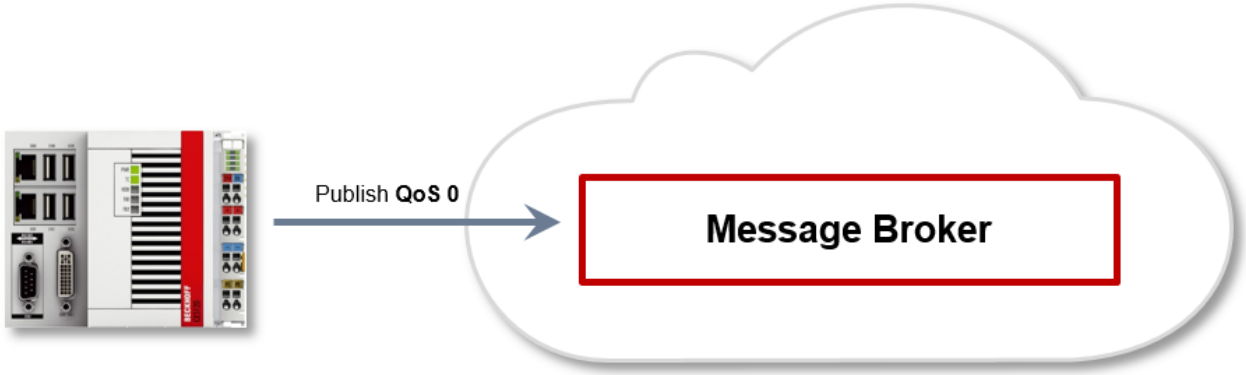
QoS is an arrangement between the sender and receiver of a message with regard to guaranteeing of the message transfer. MQTT features three different levels:

- 0 – not more than once
- 1 – at least once
- 2 – exactly once

Both types of communication (publish/subscribe) with the message broker must be taken into account and considered separately. The QoS level that a client uses for publishing a message is set by the respective client. When the broker forwards the message to client that has subscribed to the topic, the subscriber uses the QoS level that was specified when the subscription was established. This means that a QoS level that may have been specified as 2 by the publisher can be “overwritten” with 0 by the subscriber.

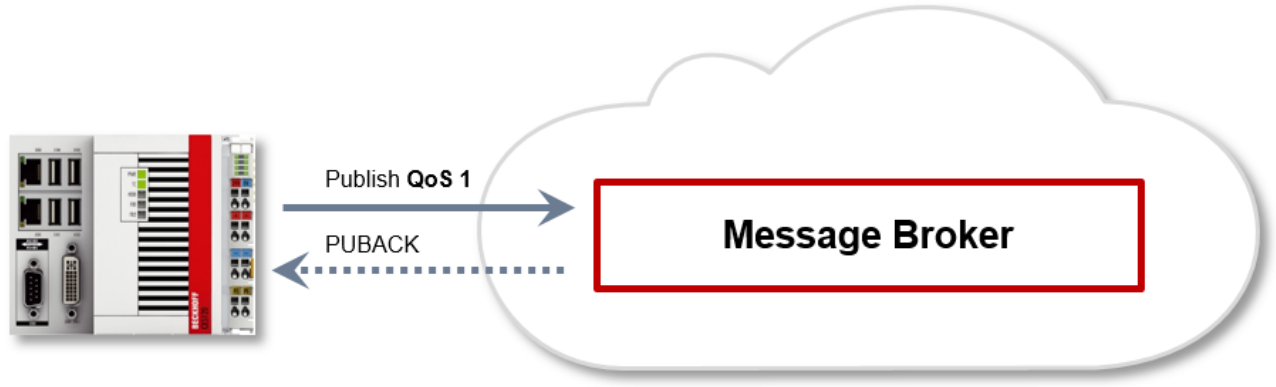
QoS-Level 0

At this QoS level the receiver does not acknowledge receipt. The message is not sent a second time.



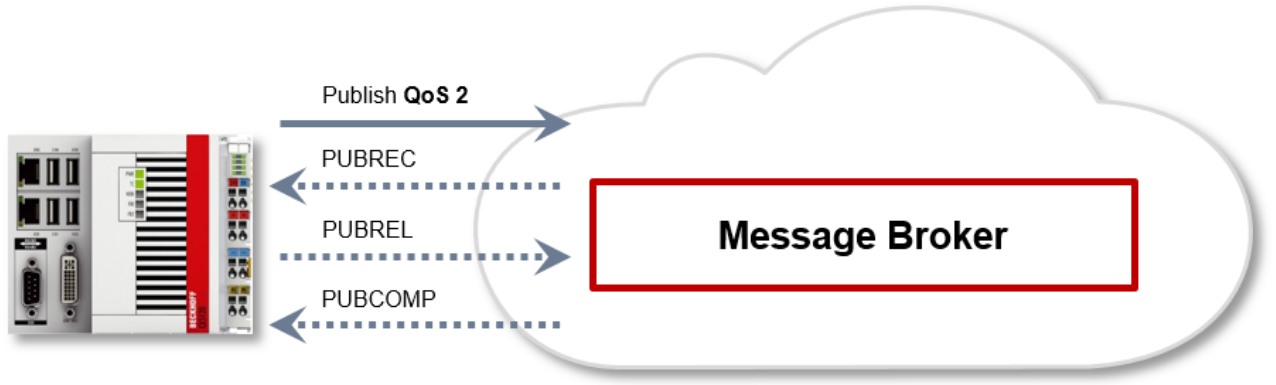
QoS-Level 1

At this QoS level the system guarantees that the message arrives at the receiver at least once, although the message may arrive more than once. The sender stores the message internally until it has received an acknowledgement from the receiver in the form of a PUBACK message. If the PUBACK message fails to arrive within a certain time, the message is resent.



QoS-Level 2

At this QoS level the system guarantees that the message arrives at the receiver no more than once. On the MQTT side this is realized through a handshake mechanism. QoS level 2 is the safest level (from a message transfer perspective), but also the slowest. When a receiver receives a message with QoS level 2, it acknowledges the message with a PUBREC. The sender of the message remembers it internally until it has received a PUBCOMP. This additional handshake (compared with QoS 1) is important for avoiding duplicate transfer of the message. Once the sender of the message receives a PUBREC, it can discard the initial publish information, since it knows that the message was received once by the receiver. In other words, it remembers the PUBREC internally and sends a PUBREL. Once the receiver has received a PUBREL, it can discard the previously remembered states and respond with a PUBCOMP, and vice versa. Whenever a package is lost, the respective communication device is responsible for resending the last message after a certain time.



The LastWill is a message sent by the broker to all clients subscribed to the matching topic in the event of an abnormal connection failure. If the MQTT client in the PLC loses the connection to the broker and a LastWill was stored when the connection was established, this LastWill is communicated by the broker without the client having to do it.

In the event of a planned disconnect, the LastWill is not necessarily transmitted according to the specification. From the PLC programmer's point of view, he can decide whether he wants to publish the LastWill before calling the disconnect. To this end, the LastWill message is published again on the LastWill topic. This is necessary because the broker would not publish the LastWill message due to the regular disconnection.

In the event of a TwinCAT context change and a resulting restart of the MQTT communication, the IoT driver sends the previously specified LastWill to the broker, because at this point, doing this from the PLC is not an option. If no LastWill was defined when the connection was established, no message will be transmitted before the disconnect.

Safety

When a connection to the message broker is established, security mechanisms such as TLS can be used to encrypt the communication connection or to execute authentication between client and message broker.

Sources

For further and more detailed information about MQTT, we recommend the following sites:

HiveMq Blog: http://www.hivemq.com/blog/mqtt-essentials/ (the main basis for this article)

5.3 Data Compression

In order to reduce the amount of data to be sent without reducing the amount of information and thus enhance the performance, a compression method derived from the Run Length Encoding method can be used. It utilizes the fact, that data of two consecutive samples in a buffer may not vary in parts. Thus knowing what parts of a previously sent buffer are equal in conjunction with the varying data, a recipient can reconstruct the next buffer without the need to receive the whole buffer. After sending the first buffer uncompressed, the logger constructs the compressed buffer by comparing user-specified units of data (e.g. every byte, every 8 byte etc.) one after another. The logger counts the amount of data units (called Compression Compare Width) that are equal and places this information in the buffer instead of the data. Dependent on the kind of data this can lead to a data saving or overhead. To decide whether a compression is purposeful or not the user is provided with a data compression saving value, which can be found on the Online-tab of every stream dialog window. A positive value indicates saving and a negative value indicates overhead.

6 Configuration

Configuring the TwinCAT Analytics Logger is very simple and can be done with just a few clicks. The configuration is divided into three main parts.

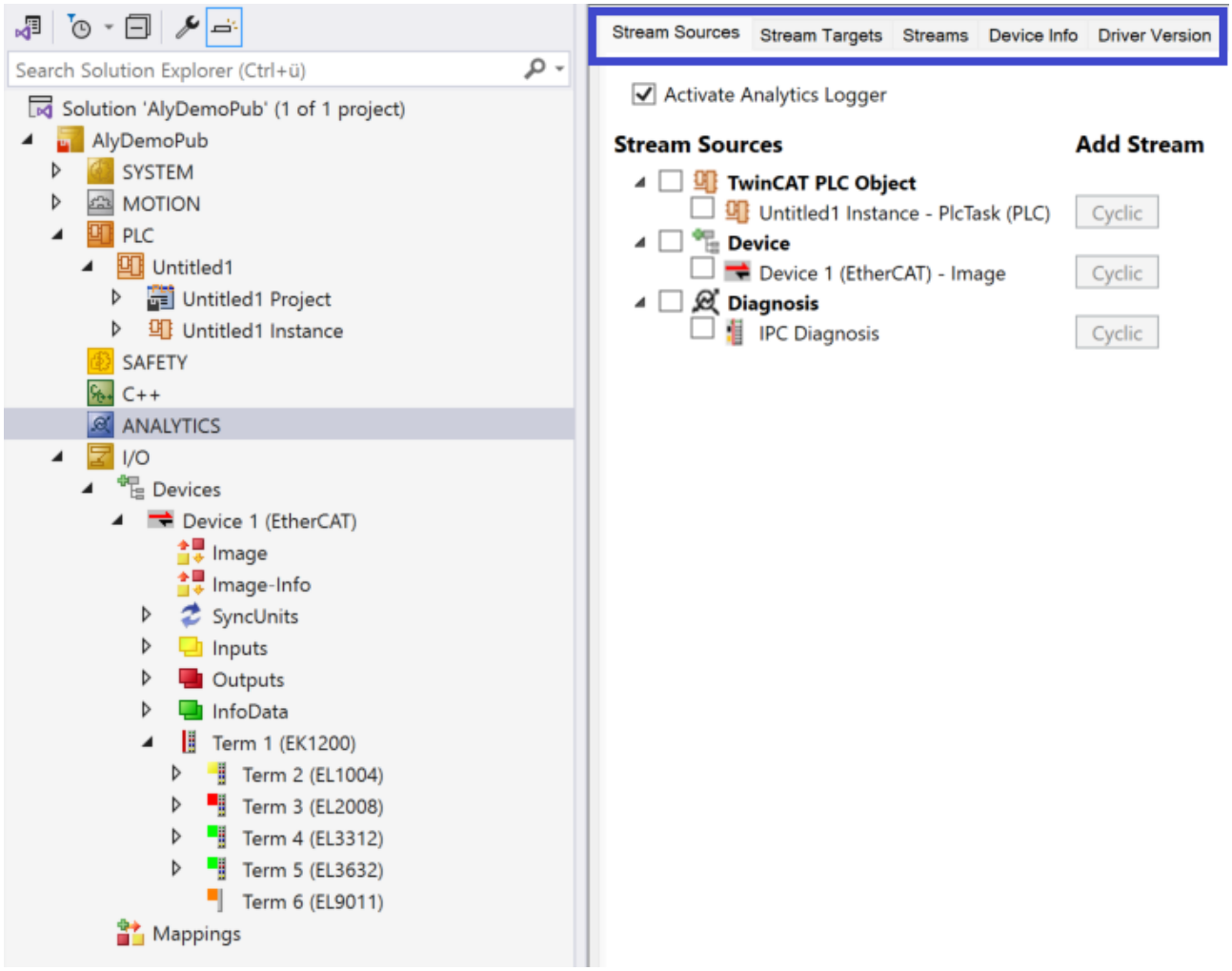
1. Basic settings
2. Context settings
3. Stream settings

The following chapters describe all setting options for the three areas.

6.1 Basic settings

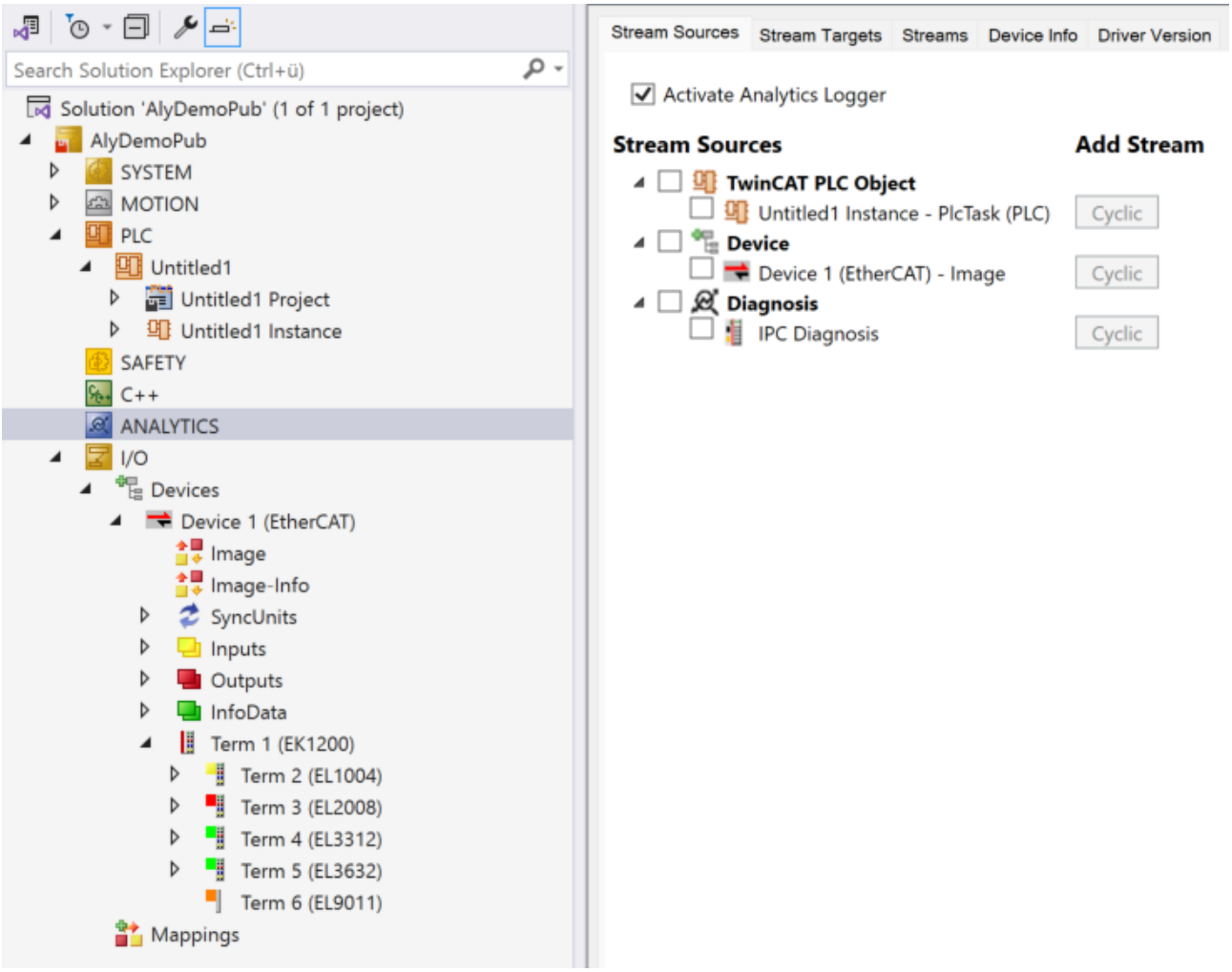
If you want to configure a TwinCAT Analytics Logger, start with the basic settings. You can access these by double-clicking the **ANALYTICS** tree node in the TwinCAT System Manager. The basic settings are divided into various tabs (marked in blue) as follows.

1. Stream Sources
2. Stream Targets
3. Stream Overview
4. Device Info
5. Driver Version

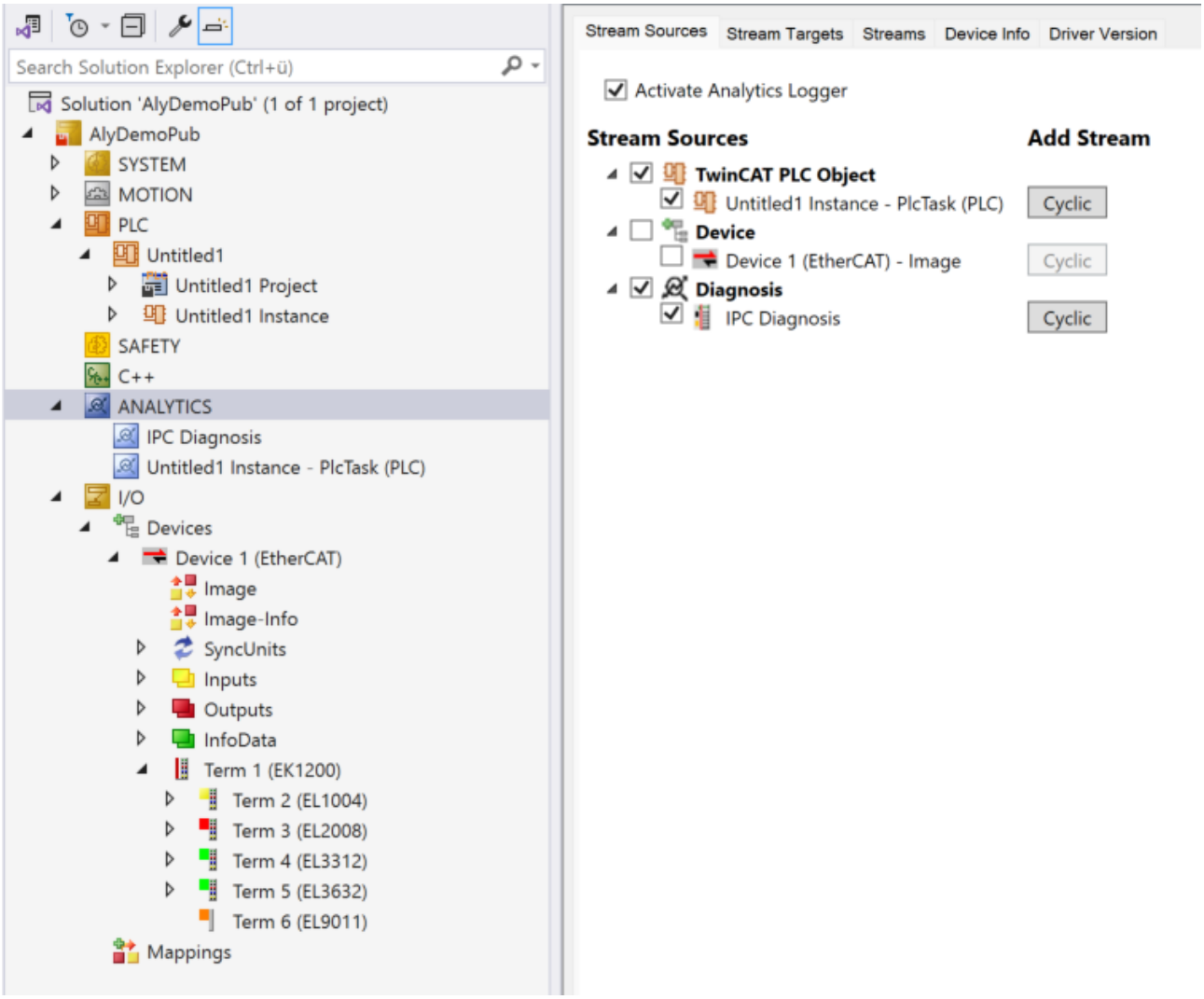


6.1.1 Stream Sources

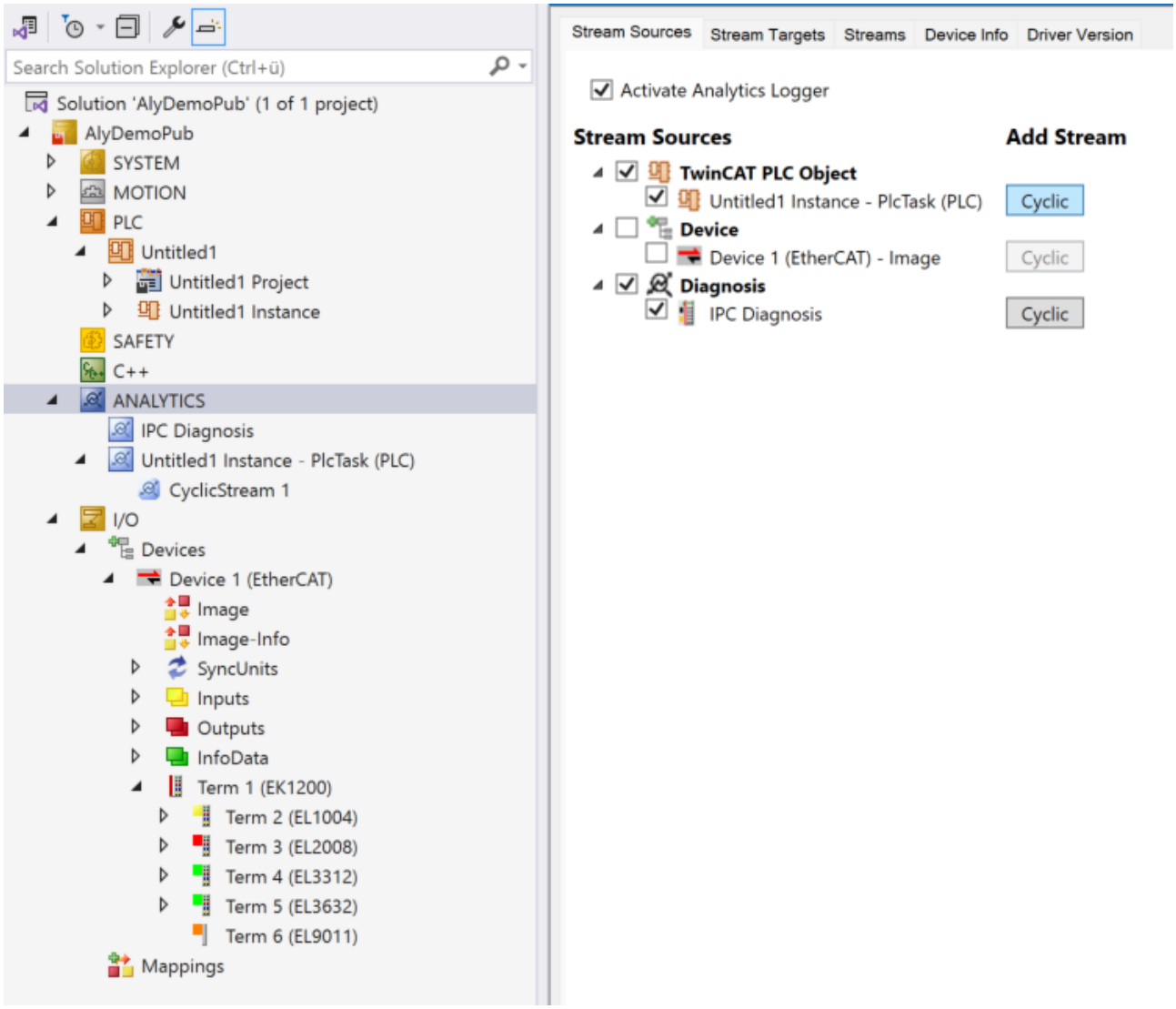
In the **Stream Sources** tab of the basic settings, you can select from which controller context the data is to be recorded. All available contexts are displayed automatically. For example, one or more PLC objects, EtherCAT and other I/O devices, as well as specific streams such as IPC diagnostics.



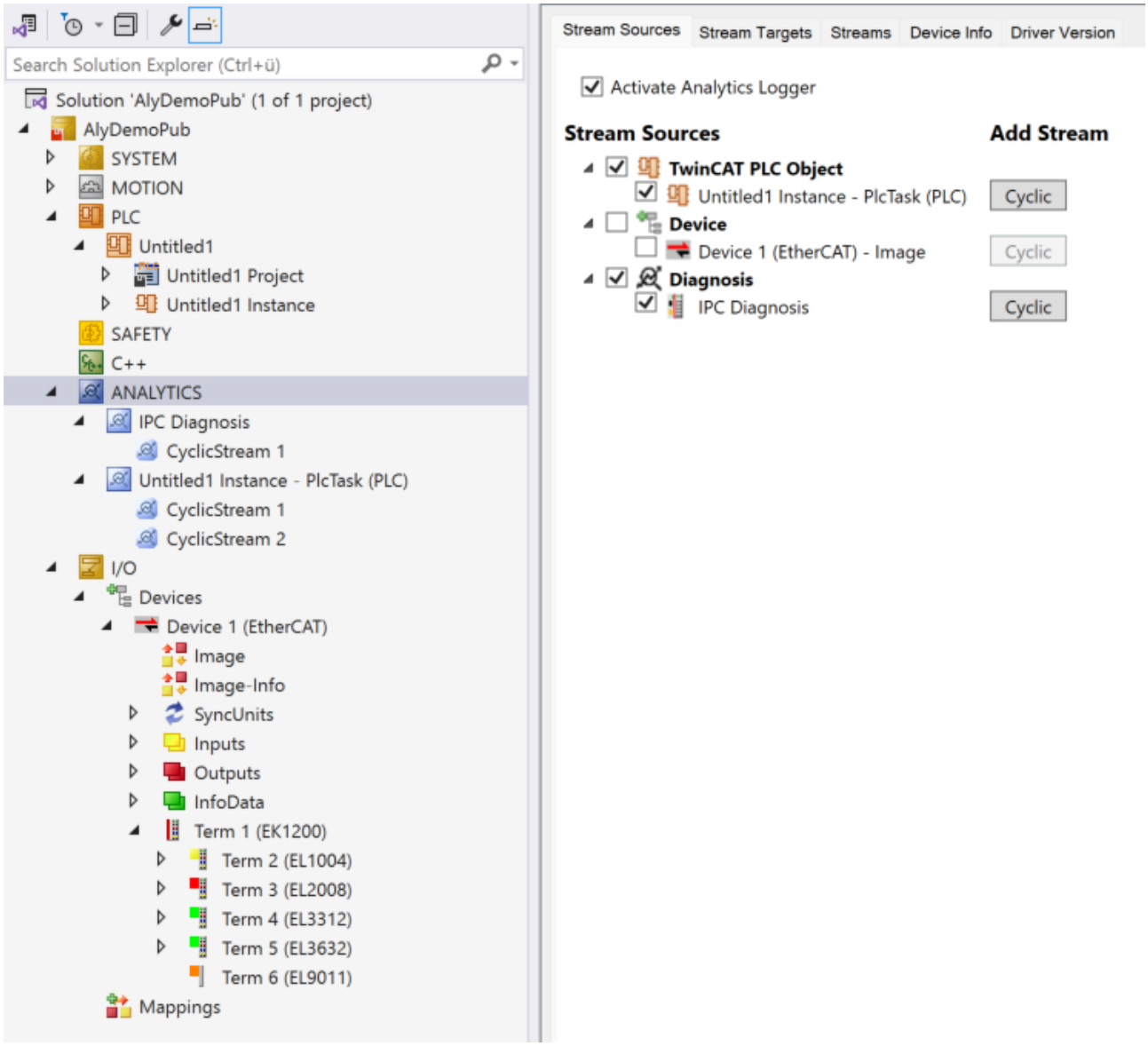
Once you have made your choice using the checkbox, the selected contexts will be visible in the tree view under the Analytics main node. In addition, the buttons under Add Stream are activated in the tab.



You can add one or more streams to a context using the buttons.



The buttons can simply be pressed several times. In the Solution Explorer, the streams are added to the respective contexts and displayed.



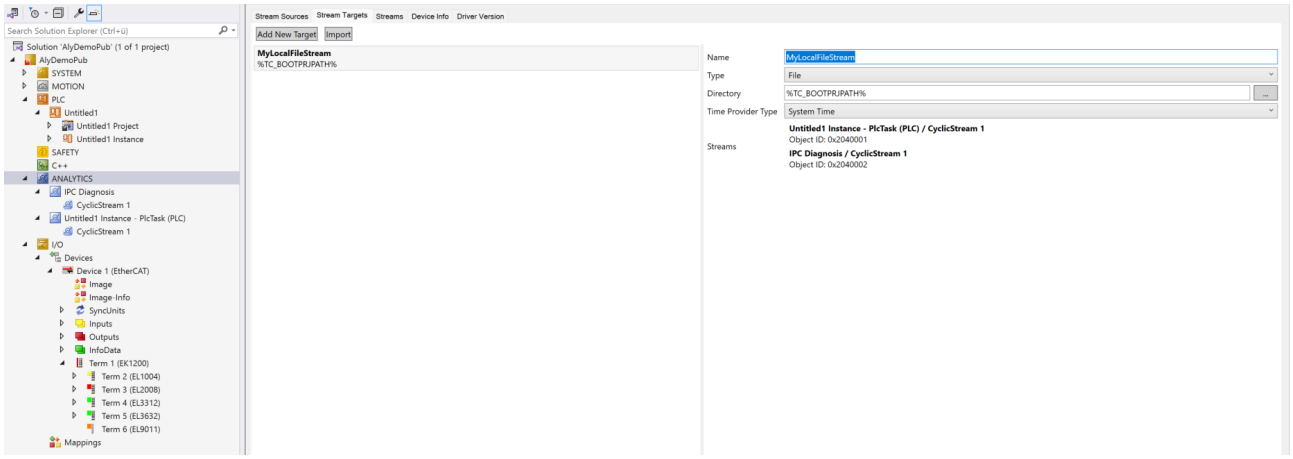
If you want to delete a stream again, simply right-click on the context menu in the tree and select **Delete**. Stream can also be temporarily switched to **Disable** via the same context menu.

6.1.2 Stream Targets

One or more targets for the acquired data can be defined in the **Stream Targets** tab. The TwinCAT Analytics Logger supports two target options. It can write data in Analytics binary format to the local file system. Or send the data in the same format to an MQTT message broker. It does not matter whether the message broker runs on the local or a remote system.

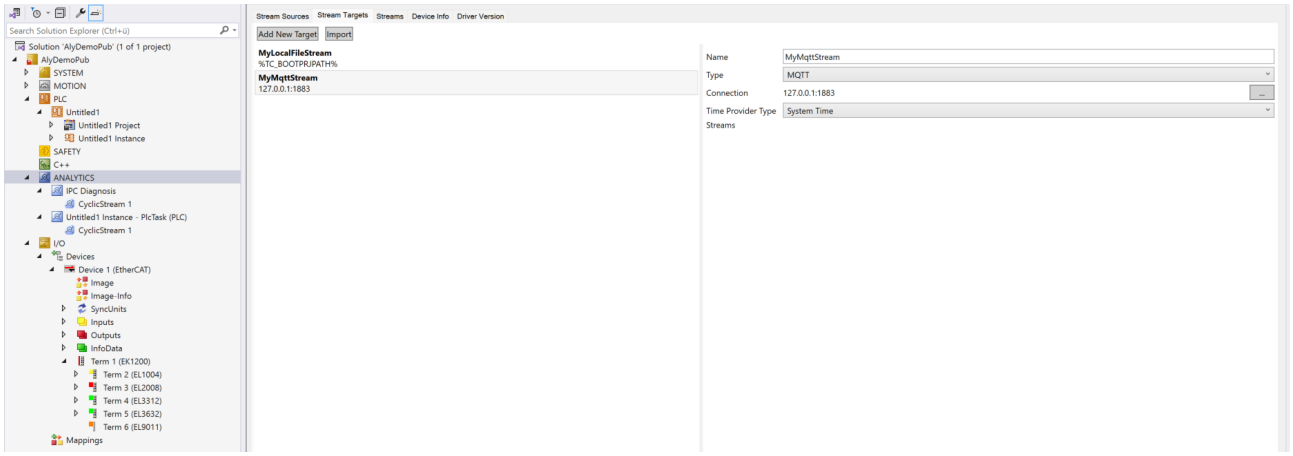
Click on **Add New Target** to create a new entry in the list. All entries in the list are later available for the stream definition and can be individually assigned to the streams. A target can also be used by different streams at the same time.

Once you have added a target, you can assign a name that describes the target. For example, MyLocalFileTarget. Then set the type of target in the drop-down box, in this case set to File.

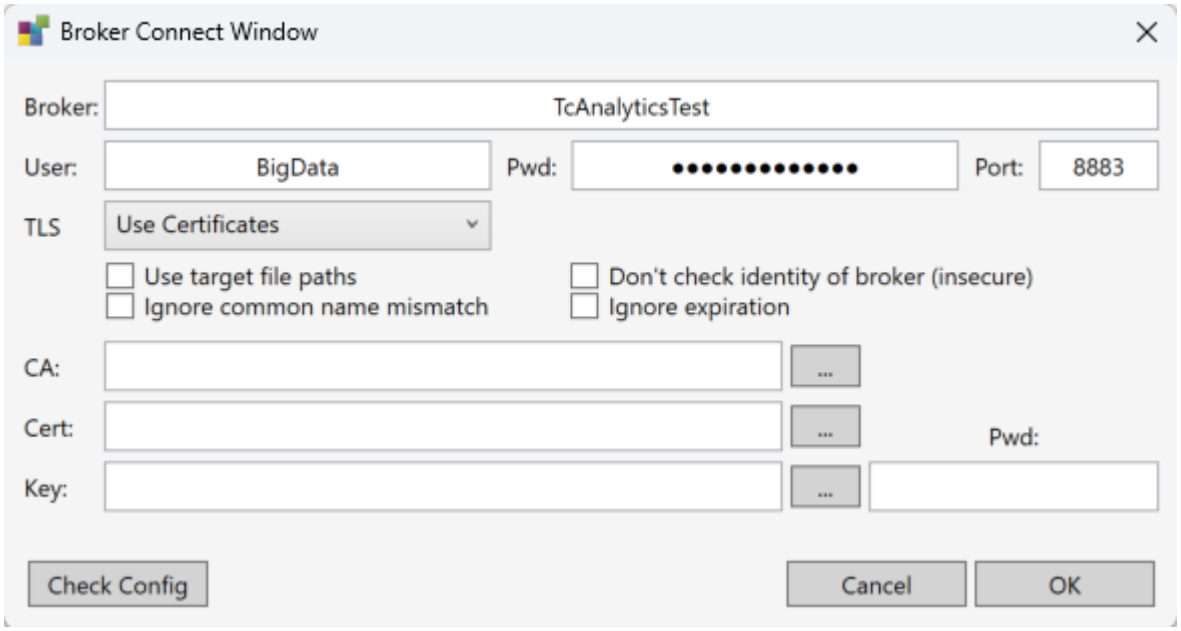


This gives you the option of choosing a directory for the data on the local hard disk. The default is the TwinCAT boot directory. A time provider type is available as a choice for any desired synchronization of timestamps with other systems. This is explained in more detail in the Timestamp correction [▶ 55] chapter, as it is also available for both target types File and MQTT.

If you choose MQTT as the target type, you have the option of specifying connection data to the message broker.

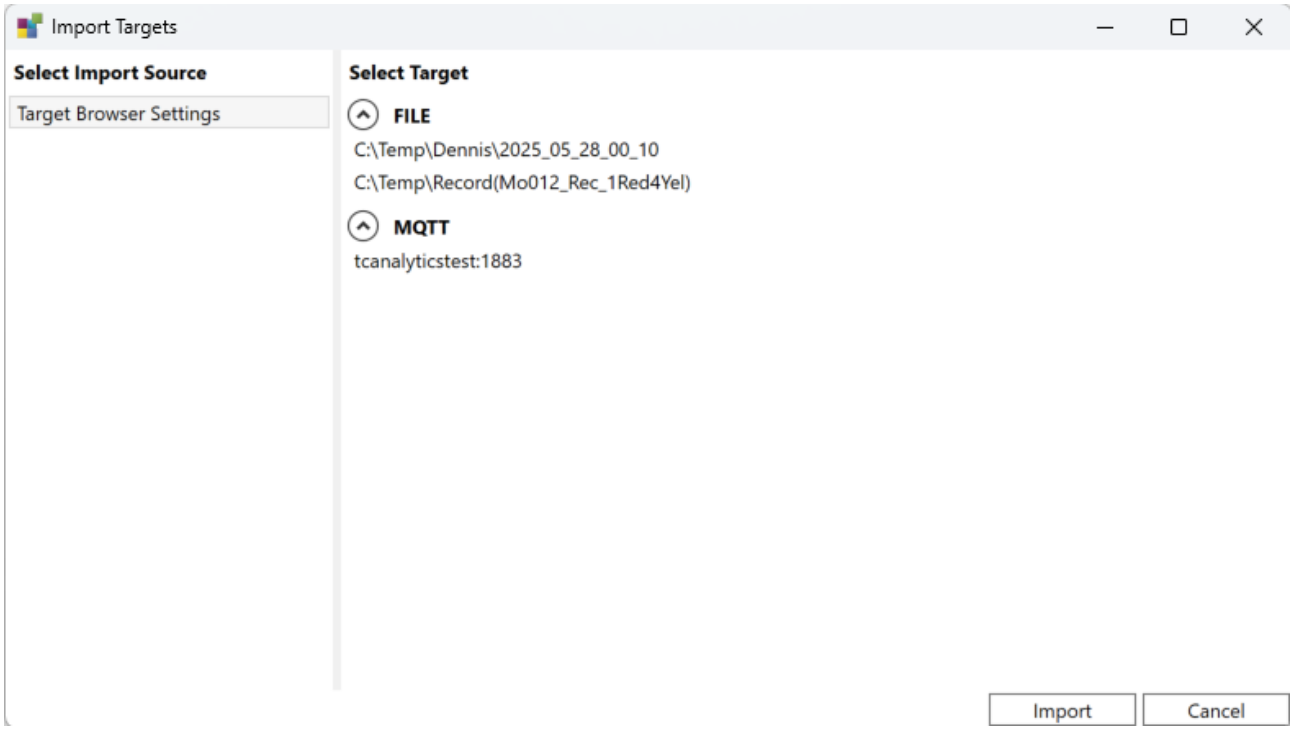


To do this, click the button with the three dots. The following window then opens.



Here you essentially enter the host name or the IP address of the MQTT message broker. It should be noted that this is a native MQTT message broker that supports retain topics. It is recommended to use the broker's authentication options. The required user and password entries can be made directly here. You must also decide whether communication via TLS should be secured with certificates. The setting options can be found in the Security TLS [▶ 57] chapter.

If you have already defined targets in the TwinCAT Target Browser, you can also import them here. To do this, please click the **Import** button. The following window opens.

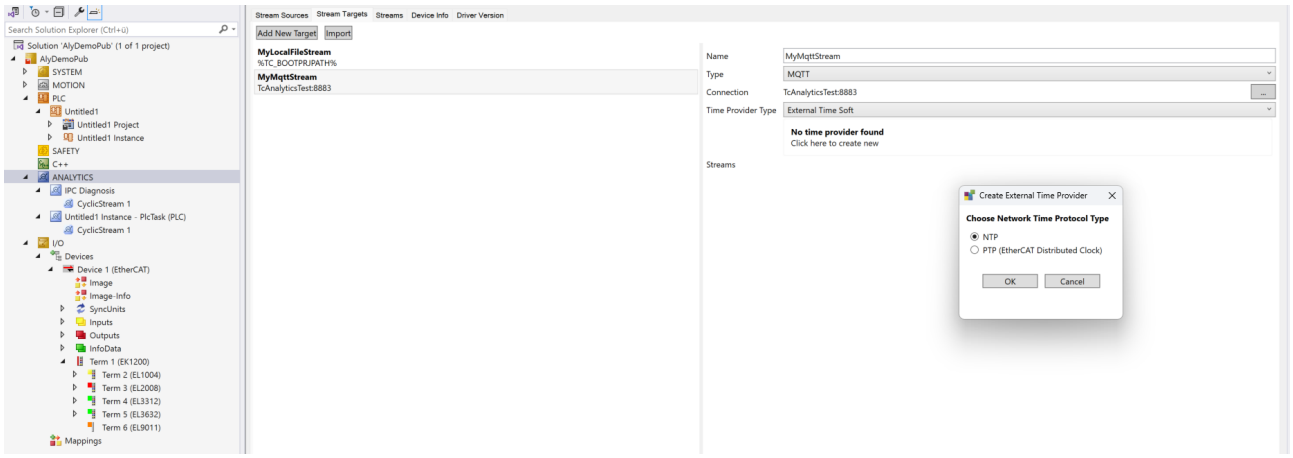


Choose a file or MQTT target and press Import. All settings, including security, are automatically applied to the Analytics Logger.

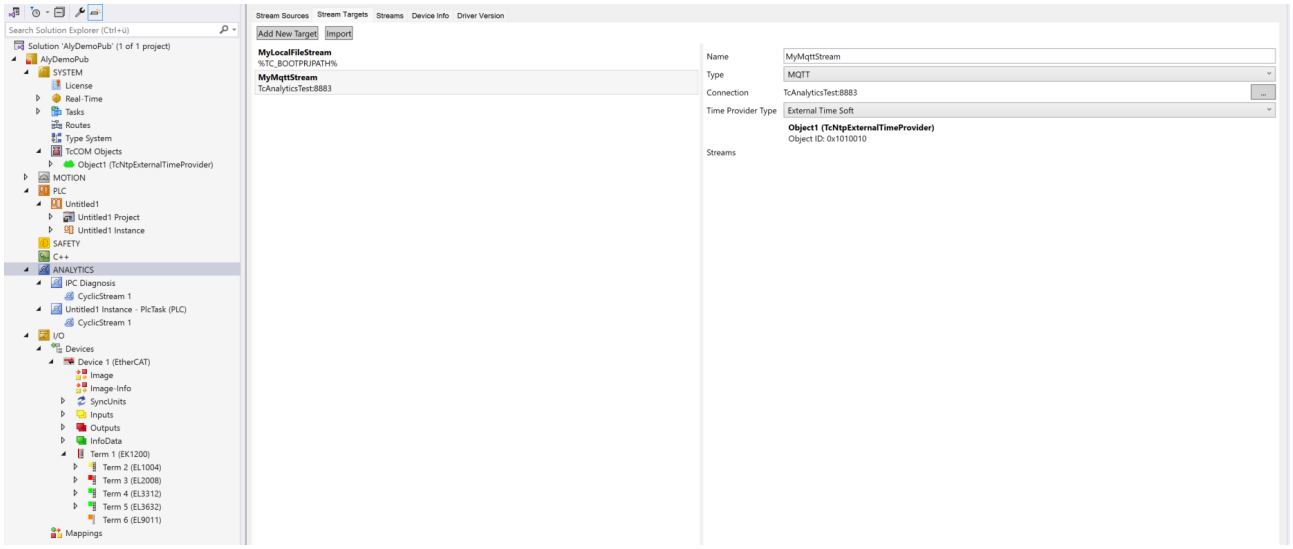
6.1.2.1 Timestamp correction

The TwinCAT time, which is used by default for timestamps in the TwinCAT Analytics Logger, may deviate from the actual system time after a longer period of use of the controller. This is due to the fact that different hardware counters are used as clock generators. With the concept of timestamp correction, it is possible to add an external offset to the TwinCAT timestamp, which corrects the timestamp. The correction using the External Time Provider can be made both in relation to an external time source via NTP (Network Time Protocol) and in relation to the EtherCAT Distributed Clock via PTP (Precision Time Protocol).

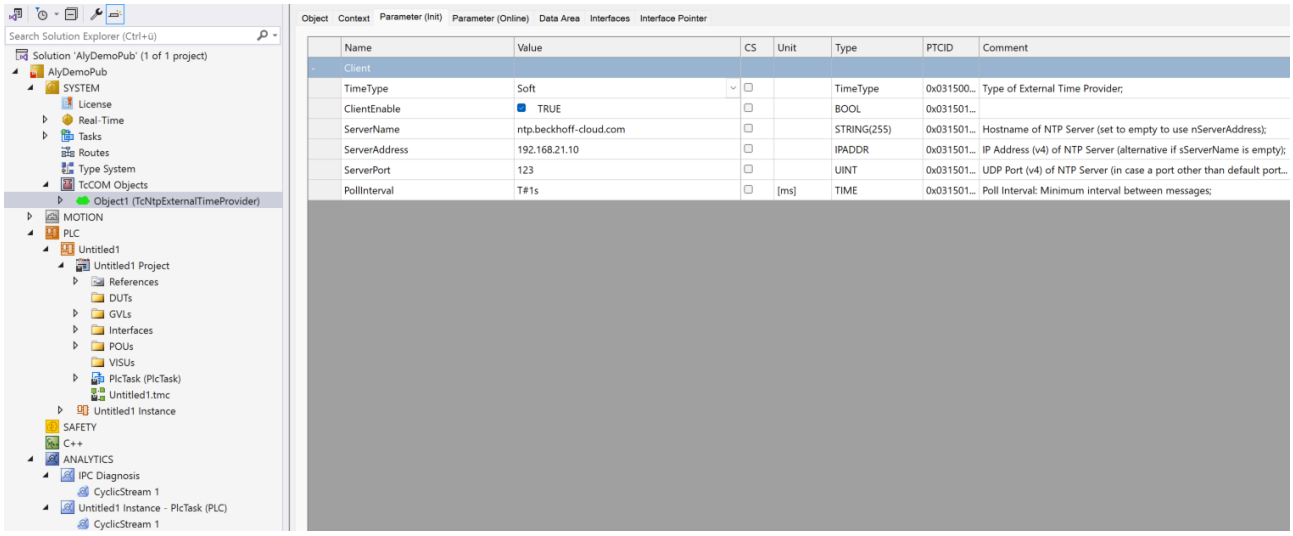
You can specify the time provider type individually for each defined target. The default value is always your own system time.



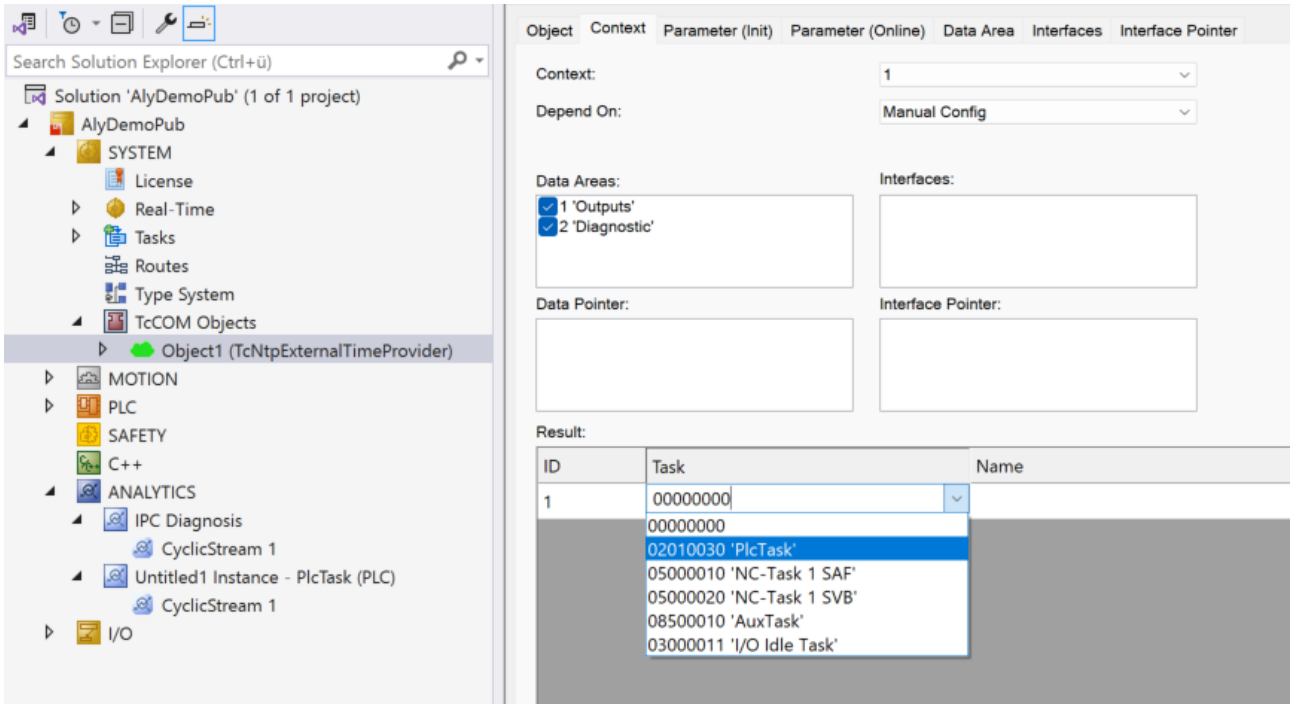
You can use the drop-down box to select a time provider in the desired accuracy class. For example, External Time Soft. If no time provider has yet been created in the TwinCAT project in the accuracy class, the Analytics Logger will output **No time provider found** at this point. You can then click on this area and a window will ask you whether you want to choose NTP or PTP (EtherCAT DC). A corresponding object is then created automatically.



Under the system node in the TwinCAT project tree, you will also immediately see a new object under TcCOM Objects. You can open this from there or double-click on the object name in the Analytics Logger tab and the same page will be opened automatically.



Here you should ensure that the **Task** that controls the Time Provider object is selected in the **Context** tab.



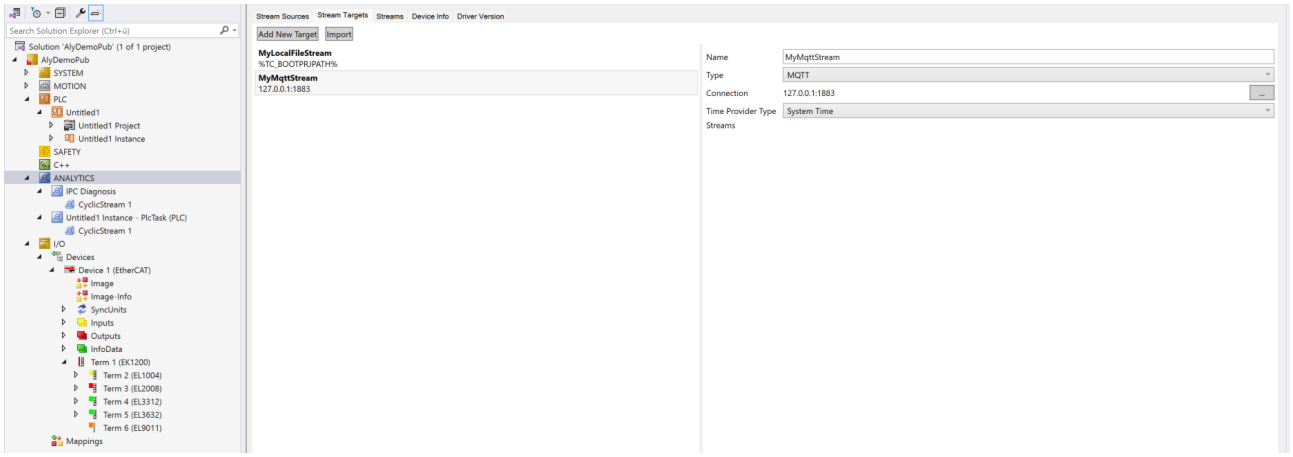
In addition, the cycle time of the provider's synchronization with the NTP server and the server host name can be set via the **Init parameter**. The Analytics Logger uses this to create timestamp corrections to compensate for the difference between the TwinCAT system time and the NTP-synchronized time.

6.1.2.2 Security - TLS

TLS (Transport Layer Security) provides a secure communication channel between a client and a server. At its core, TLS consists of cryptographic protocols that use a handshake mechanism to negotiate various parameters to establish a secure connection between the client and the server. The TwinCAT Analytics Logger supports TLS versions 1.2 and 1.3, as well as the modes CA Certificates, CA Certificates & Client Certificate, and Preshared Key (PSK).

MQTT communication with TLS

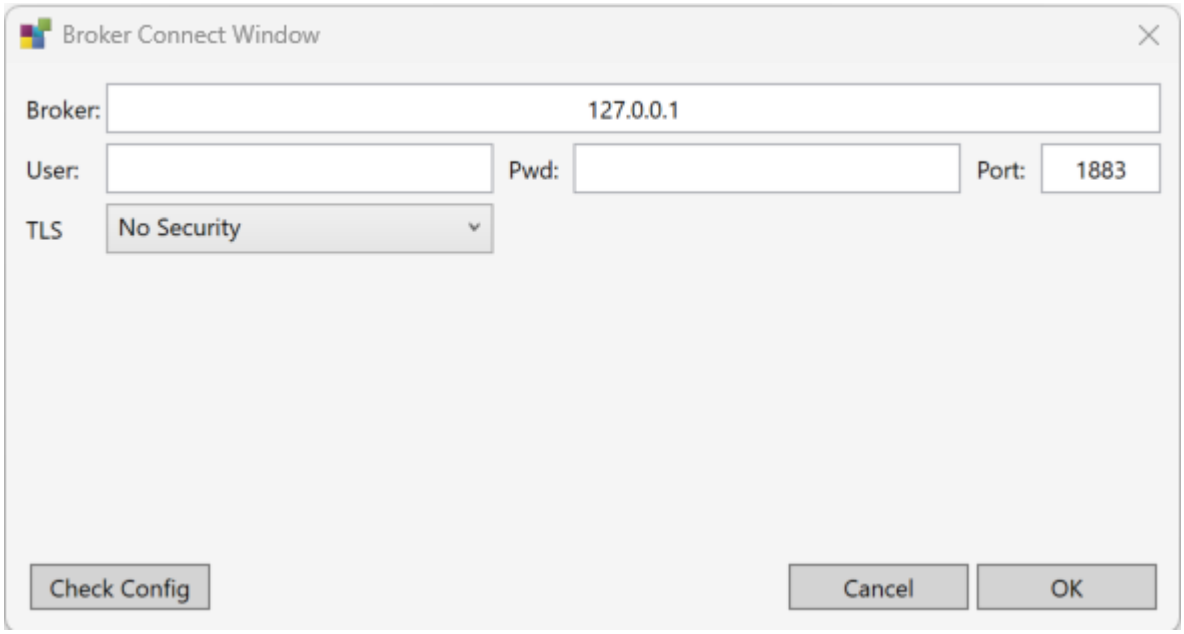
When certificates are used, TCP port 8883 is exclusively reserved for MQTT over TLS!



If you have chosen an MQTT target, you can click on the button with the three dots under Connection and the following window for the MQTT connection data will open. After configuring the broker and entering the user name and password, you can open the TLS drop-down list and select Use Certificates or Use Pre-Shared Key (PSK).

Once TLS is enabled, another drop-down list appears to the right of it, allowing you to select the TLS version:

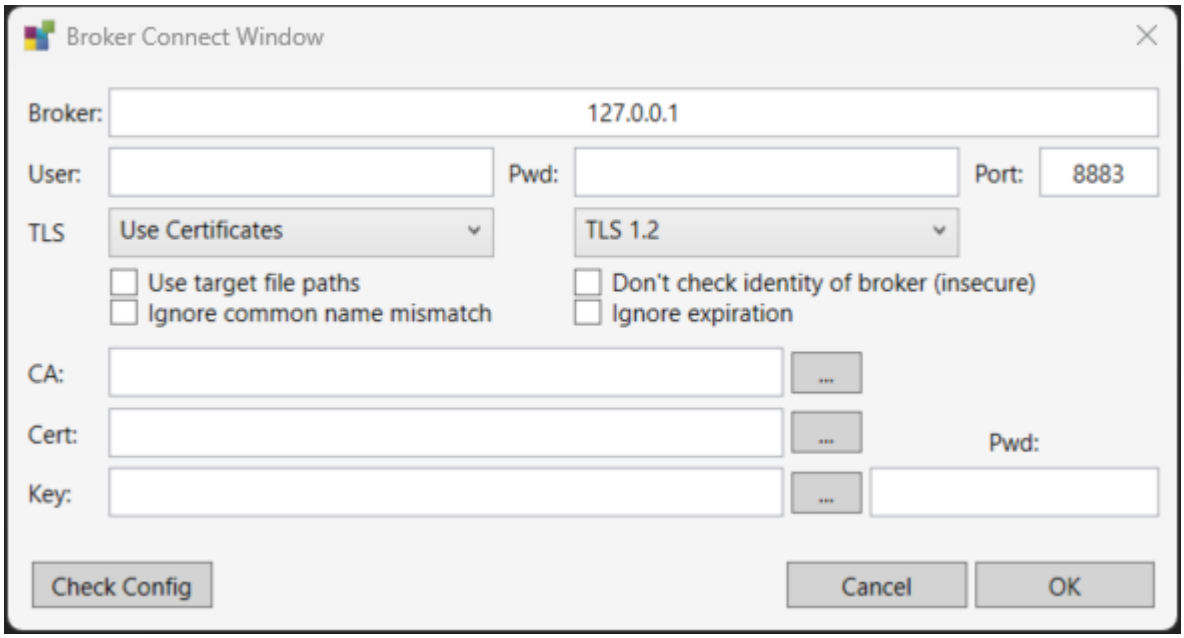
TLS version	Description
TLS 1.2	A proven standard, widely used and compatible.
TLS 1.2	Current version with improved security features and a faster handshake.



CA certificate

Encryption and authentication via TLS can also be accomplished through a certificate authority (CA). The CA provides a signature covering the public key for all communication clients. In this case an MQTT client connects to a message broker without a dedicated client certificate.

The Cert and Key fields do not need to be filled in.

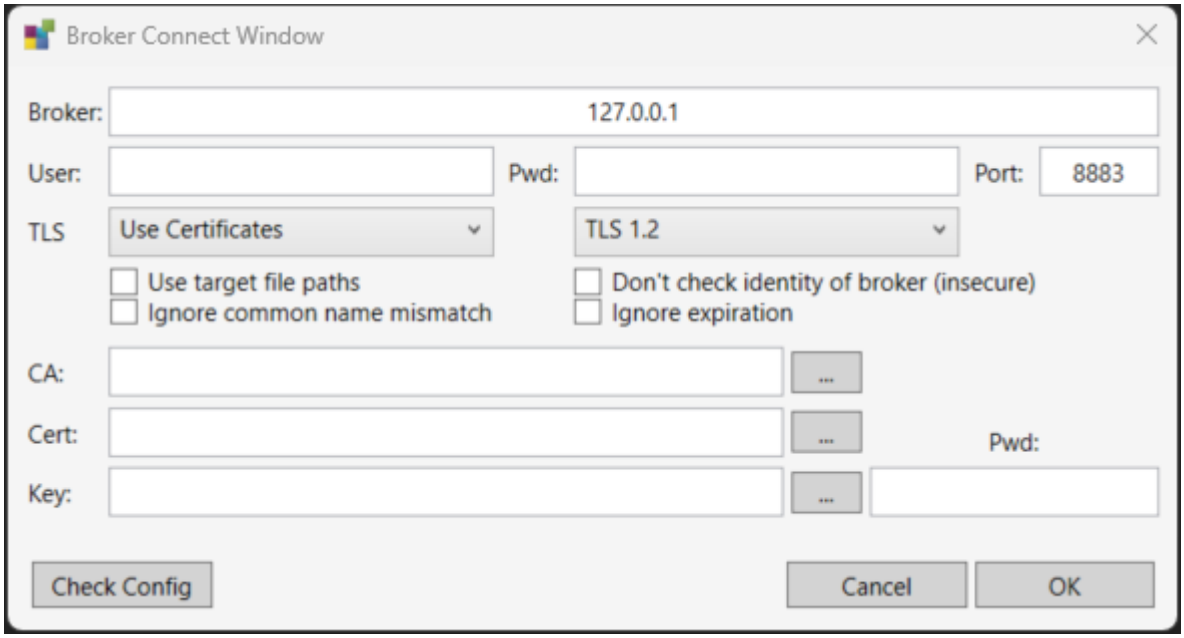


- **Use target file paths:** If this option is activated, the target searches for the certificates on its own system under the specified path. If this option is not active, the specified paths are checked on the engineering system and the certificates are downloaded from there to the target.
- **Ignore common name mismatch:** If this option is activated, the common name (CN) of the server certificate is not checked.
- **Don't check identity of broker (insecure):** This option ignores the result of the server certificate check.

- **Ignore expiration:** If this checkbox is selected, the expiration of the server certificate is ignored.

CA Certificate & Client Certificate

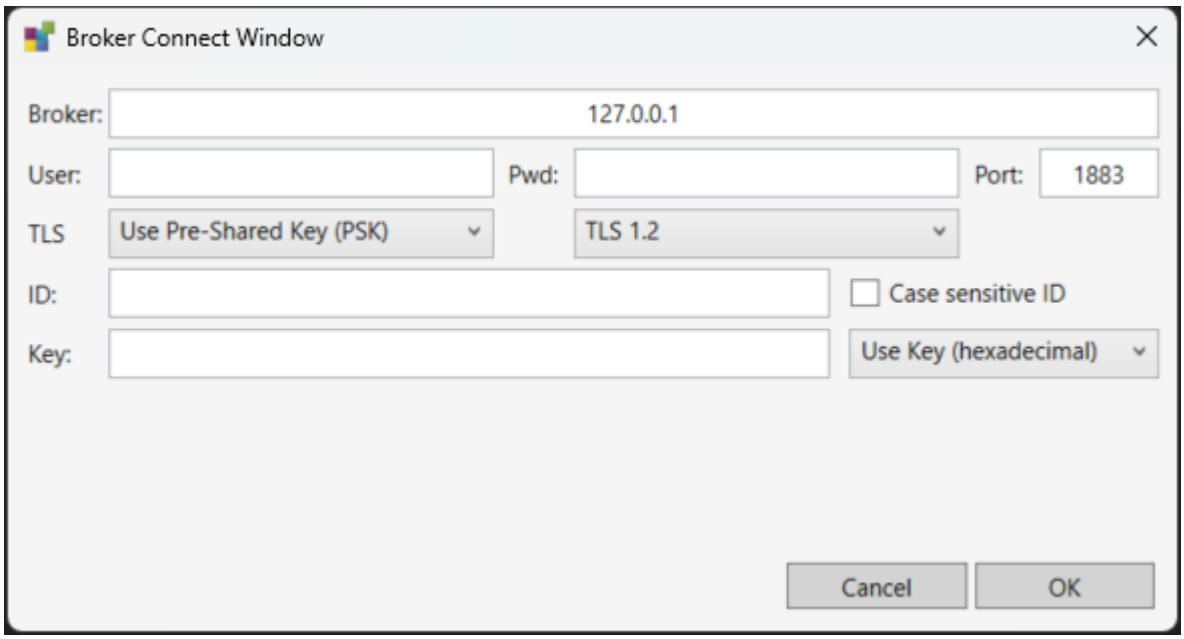
Encryption and authentication via TLS can also be accomplished through a certificate authority (CA). The CA provides a signature covering the public key for the message broker (the so-called server key) and usually also for all connecting clients. All communication devices can then trust each other, because the issuing certificate authority is trusted.



- **Use target file paths:** If this option is activated, the target searches for the certificates on its own system under the specified path. If this option is not active, the specified paths are checked on the engineering system and the certificates are downloaded from there to the target.
- **Ignore common name mismatch:** If this option is activated, the common name (CN) of the server certificate is not checked.
- **Don't check identity of broker (insecure):** This option ignores the result of the server certificate check.
- **Ignore expiration:** If this checkbox is selected, the expiration of the server certificate is ignored.

Preshared Key (PSK)

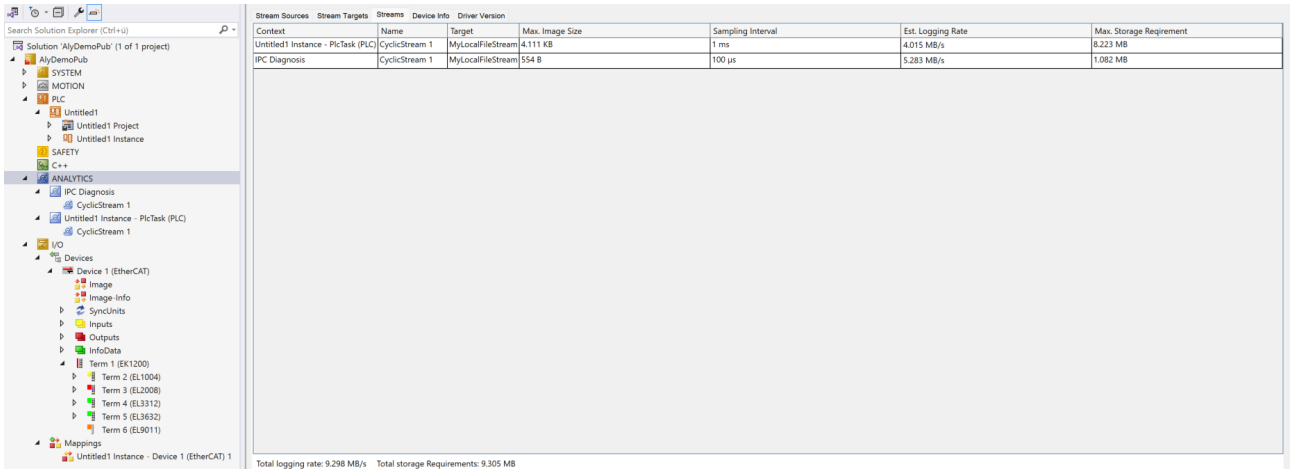
The TLS PreSharedKey (PSK) method offers a simple option for realizing encryption between client and message broker. Client and broker recognize a common password, which is used to encrypt and decrypt the packets.



6.1.3 Stream Overview

The **Stream Overview** tab of the basic settings contains an overview of all the streams created in the Analytics Logger. The tab is therefore only relevant once the streams have been completely defined. Only then is the information summarized here available.

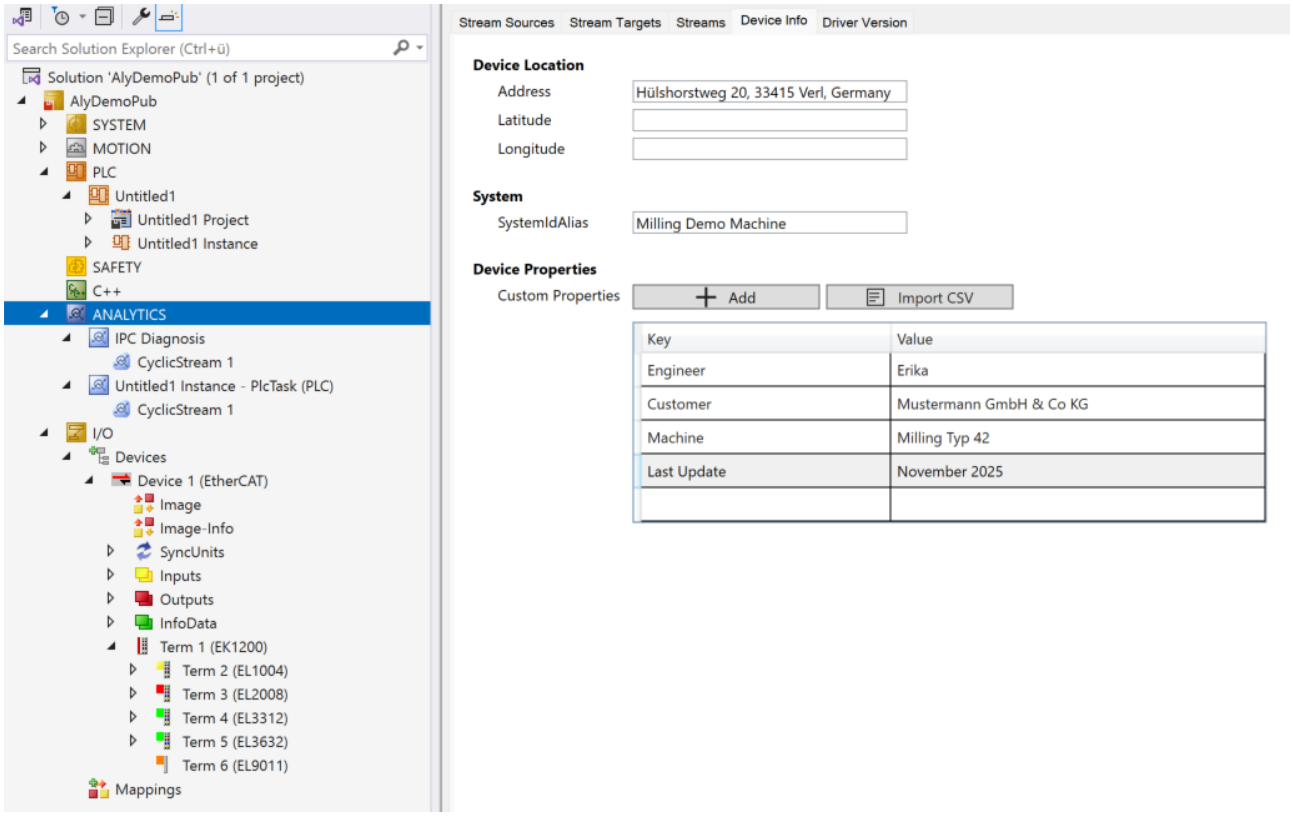
You can see the contexts with the associated stream names, their targets, the amount of data, the sampling interval, and the resulting data rate with the maximum hard disk space required.



In addition, the summed values of all streams are shown at the bottom of the page. In other words, the total data rate and the maximum hard disk space required. This eliminates the need for tedious totaling and gives you an overview of all streams.

6.1.4 Device Info

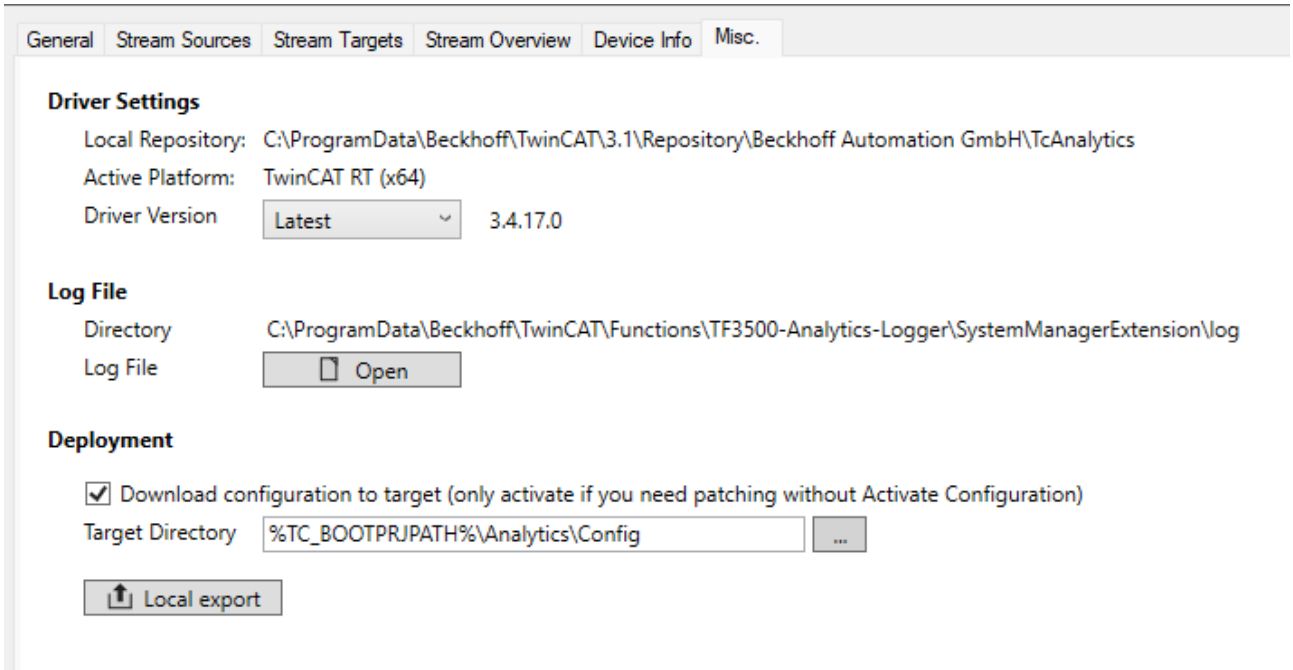
In the basic **Device Info** tab, you can enter information about the control device on which the TwinCAT Analytics Logger is running. This includes the address, coordinates and a system-ID alias, which makes it easier for the user to identify the device in the Analytics workflow.



It is also possible to define additional key-value pairs in order to use them for static metadata. If the description of the key-value pairs is available as a csv file, it can also be easily imported here.

6.1.5 Miscellaneous

The last tab in the basic settings contains various settings. The first part is about the driver versions that are used. The version selected here (default: Latest) is automatically downloaded to the target system (controller) when the TwinCAT configuration is activated. This applies to all operating systems under 4026, eliminating the need for additional installation in the form of packages on the target system itself.



The log for the Analytics Logger System Manager extension can also be viewed here. Beckhoff Support may also ask you to provide this file when handling a support request.

Copy Deployment

You can also enable and configure a type of patch mechanism using Copy Deployment.

NOTICE

TwinCAT restart

Use this option only if it will no longer be possible to activate the configuration (TwinCAT Activate Configuration) in the future, but changes still need to be made! A TwinCAT restart is still required!

Copy Deployment allows you to make changes using patch files in a target directory and apply them after restarting TwinCAT. the following workflow describes how Copy Deployment works:

Workflow:

1. **Enable Copy Deployment** – Check the “Download configuration to target” box in the Deployment section.
2. **Select the target directory** – TwinCAT saves this path in the driver settings. After step 3, the driver reads the patch files from this folder.
3. **Activate Configuration** – Activate the configuration with the target directory once on the target system using TwinCAT Activate Configuration. Only then will the driver know where to find the patch files.
4. **Make changes** - You can then make changes to the logger configuration. If you make only one change after step 6, the configuration remains unchanged.
5. **Export patch files** – Use the “Local export” button to export the modified configuration as a JSON patch file. The export creates one folder containing one patch file for each stream. The export structure follows this format: <Export folder>/<StreamName>_<ID>/Config.json. The following section describes the format of a patch file in more detail.
6. **Adding and/or removing symbols (optional)** – You can extend the JSON structures of the patch files by adding an array of symbols. This allows you to add or remove symbols for logging. (See the examples in the following section)
7. **Placing the patch files**: Place the patch files in the target system under the Target Directory (see step 2), following the same folder structure as in the export.
8. Restart TwinCAT.

⇨ After that, the driver loads the patch files and runs using this configuration.

NOTICE

Activating the configuration

Activate the configuration using the Target Directory once so that the driver can read the patch files from the Target Directory!

Patch file format

Each **Config.json** file contains:

Level	Field	Description
1	InitData	The complete stream initialization data (encoded). Always present.
1	InitParameters	This object contains the configuration for the Analytics Logger stream. These parameters should not be changed and are intended solely to provide transparency within the patch file.
1	Symbols	Optional list of symbol changes (add or disable).
2	Name	Complete name of the symbol (required)
2	TypeName	Symbol type name (required when adding a symbol)
2	Flags	Additional information. Optional. Flags = 2 means that Size is specified in bits rather than bytes.
2	Size	Symbol size (Required when adding a symbol)
2	IdxGrp	Symbol index group (decimal) (required when adding a symbol if symbolic mapping is disabled. For example, with Device Streams)
2	IdxOffs	Symbol index offset (decimal) (required when adding a symbol if symbolic mapping is disabled.) For example, with Device Streams)
2	Properties	An array consisting of multiple key-value pairs. (Required when adding a pointer or a reference, or when disabling a symbol) (See examples in the following section)

Add symbol

To add a new symbol to the stream, add an entry with name, data type, and size.

Example: Add **MAIN.VarInt64** (8-byte integer):

```
{
    "InitData": "...",
    "Symbols": [
        {
            "Name": "MAIN.VarInt64",
            "TypeName": "LINT",
            "Size": 8
        }
    ]
}
```

Example: Add **Inputs.Frm0State of a Device (**2-byte integer):

```
{
    "InitData": "...",
    "Symbols": [
        {
            "Name": "Inputs.Frm0State",
            "TypeName": "UINT",
            "Size": 2
            "IdxGrp": 61488,
            "IdxOffs": 1520,
        }
    ]
}
```

Example: Adding symbols with bit data types:

```
{
    "InitData": "...",
    "Symbols": [
        {
            "Name": "MAIN.VarBit1",
            "TypeName": "BIT",
            "Flags": 2,
            "Size": 1
        },
        {
            "Name": "MAIN.VarBit4",
            "TypeName": "BIT",
            "Flags": 2,
            "Size": 4
        }
    ]
}
```

Example: Adding **a pointer**:

```
{
    "InitData": "...",
    "Symbols": [
        {
            "Name": "MAIN.PtrToInt^",
            "TypeName": "UINT",
            "Size": 2
            "Properties": [
                {
                    "Name": "DataAccess",
                    "Value": "Handle"
                }
            ]
        }
    ]
}
```

Example: Adding **references**:

```
{
    "InitData": "...",
    "Symbols": [
        {
            "Name": "MAIN.RefToInt",
            "TypeName": "UINT",
            "Size": 2
            "Properties": [
                {
                    "Name": "DataAccess",
                    "Value": "Handle"
                }
            ]
        }
    ]
}
```

Disable symbol

To disable an existing symbol, you reference it by name and set its “Disable” property.

Example: Disable **MAIN.VarInt32**:

```
{
    "InitData": "...",
    "Symbols": [
        {
            "Name": "MAIN.VarInt32",
            "Properties": [
                { "Name": "Disable" }
            ]
        }
    ]
}
```

Combined (add + disable)

Both operations can be combined into a single patch file.

Edit stream configuration

Some settings in the stream configuration can also be edited via the patch file. The settings are described below and explained with examples.

Level	Field	Description
1	Mqtt	The MQTT section defines an MQTT destination to which the recorded data is sent.
2	SamplesPerBuffer	Number of samples per transmitted buffer.
2	CompressionWidth	Compression width in bytes (0 = no compression).
2	SamplingDivider	Sampling rate divider (0/1 = every sample).
2	SystemIdAlias	Optional alias for the system ID.
2	MqttConfig	Embedded MQTT connection parameters (see below).
3	QoS	MQTT Quality of Service (0, 1, or 2).
3	TcpPort	Broker's TCP port (default 1883)
3	ClientId	MQTT client ID
3	Host	IP address or host name of the broker.
3	Topic	Base topic under which data is published.
3	UserName	User name used for authentication.
3	UserPwd	Password for authentication.
3	Tls	TLS/certificate parameters (see the “TLS parameters” section).
4	Version	TLS version, e.g., “tlsv1.2” or “tlsv1.3”
4	CaFile	Path to the CA certificate file.
4	CertFile	Path to the client certificate file.
4	KeyFile	Path to the file containing the client's private key.
4	KeyPwd	Password/passphrase for the private key.

Example: Enable MQTT destination:

```
{
    "InitData": "...",
    "Mqtt": {
        "SamplesPerBuffer": 32,
        "CompressionWidth": 8,
        "MqttConfig": {
            "TcpPort": 1883,
            "Host": "127.0.0.1",
            "Topic": "Analytics/Stream1"
        }
    }
}
```

Example: MQTT destination with TLS encryption (certificates as file paths):

```
{
    "InitData": "...",
    "Mqtt": {
        "MqttConfig": {
            "TcpPort": 8883,
            "Host": "broker.example.com",
            "Topic": "Analytics/Stream1",
            "Tls": {
                "Version": "tlsv1.2",
                "CaFile": "C:\\Certs\\ca.crt",
                "CertFile": "C:\\Certs\\client.crt",
                "KeyFile": "C:\\Certs\\client.key",
                "KeyPwd": "secret"
            }
        }
    }
}
```

File destination configuration

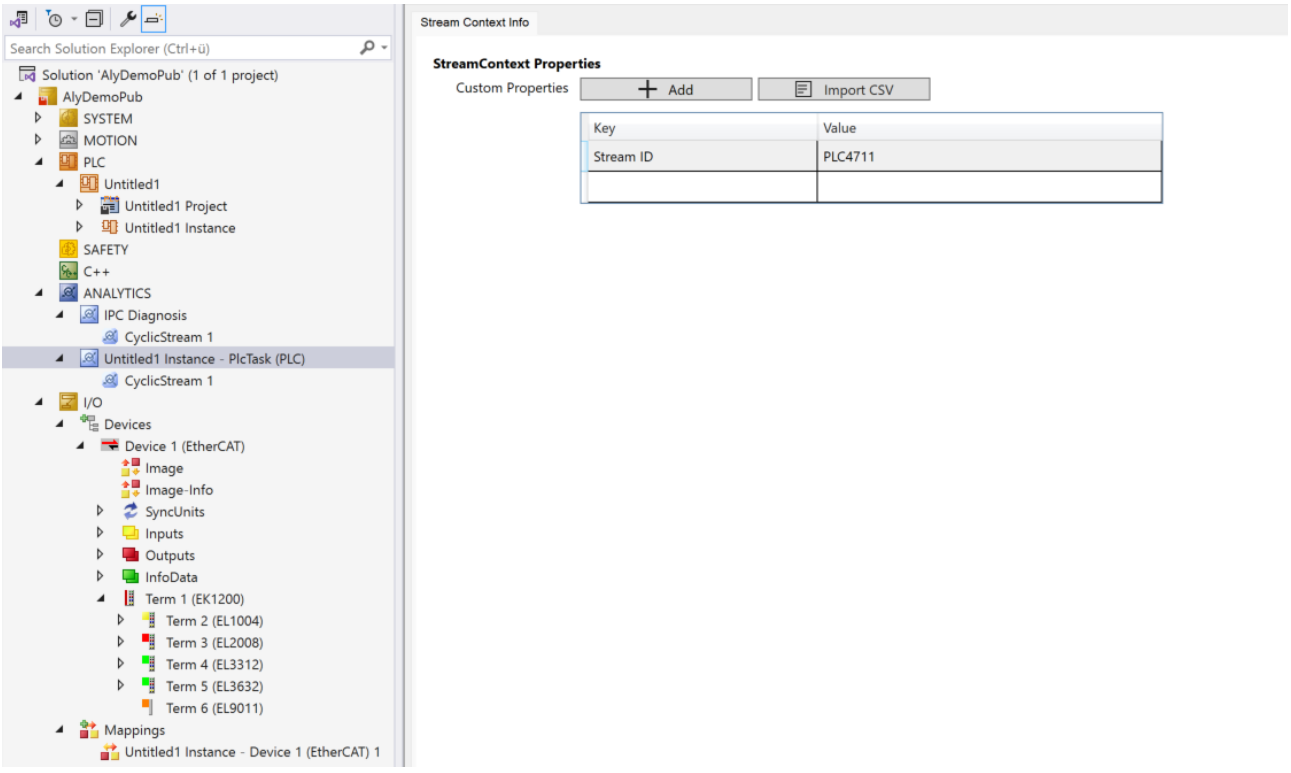
Level	Field	Description
1	File	The “File” section defines a folder where the recorded data is written.
2	BuffersPerFile	Number of buffers per file
2	SamplesPerBuffer	Number of samples per buffer.
2	CompressionWidth	Compression width in bytes (0 = no compression).
2	FilesInRingBuffer	Number of files in the ring buffer (0 = no ring buffer).
2	SamplingDivider	Sampling rate divider (0/1 = every sample).
2	FilePath	Destination path or file name for the recording.
2	SystemIdAlias	Optional alias for the system ID.

Example: Enable file destination with ring buffer:

```
{
    "InitData": "...",
    "File": {
        "SamplesPerBuffer": 64,
        "BuffersPerFile": 100,
        "FilesInRingBuffer": 10,
        "FilePath": "C:\\DataStore\\FileStreamTarget"
    }
}
```

6.2 Context settings

The second level in the Analytics Logger where settings can be made is the **Stream Context Info**. In the tree view, this level is used in particular for organization and structuring in order to separate different contexts, such as from the PLC and from different I/O devices.



Add stream

You can create new streams directly at the context level without having to switch to the tree view. The following buttons are available under **Add Stream**:

Button	Description
Cyclic	Creates a new cyclic stream that records data at the task cycle time.
On Event	Creates a new event-based stream. This button is only visible if the current context supports event-based streams.

You can also store key-value pairs with static information on the respective context here again. If required, you can import the data pairs from a CSV file.

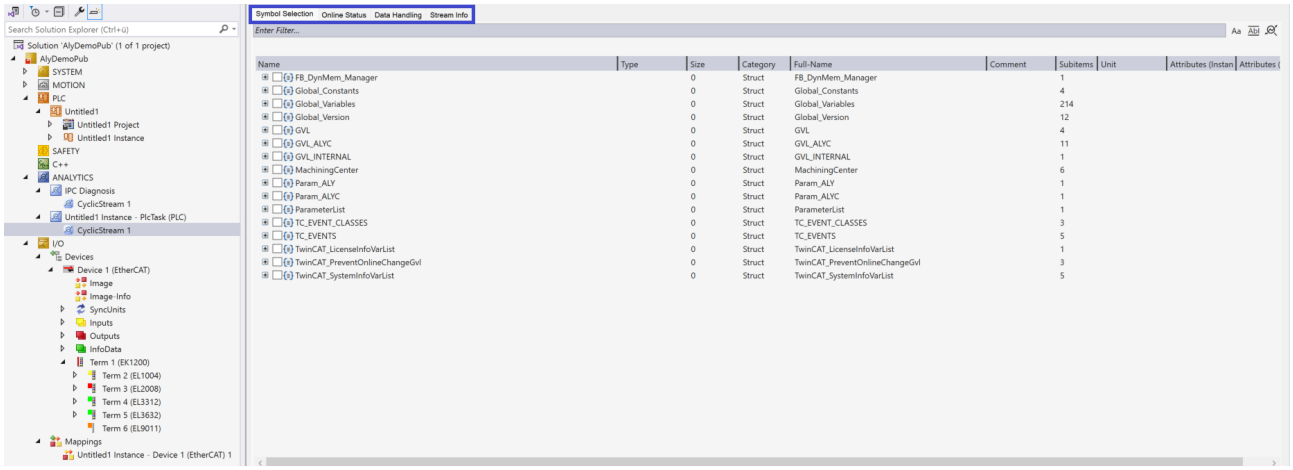
6.3 Stream settings

The third and final configuration part for the TwinCAT Analytics Logger is the level of the actual streams. This is the level where, for example, symbols, data compression, and much more can be defined.

Generally, a distinction is made between cyclic streams, event-driven streams, and specific streams. Cyclic streams record data in the control cycle, i.e., with the task cycle time. This synchronization with the control cycle offers the best performance. Specific streams basically pursue a different type of data acquisition that is specifically linked to the purpose. For example, IPC diagnostics is a specific stream in which data is also communicated cyclically, but is retrieved by the controller using the polling method. Again, this is not as precise as with a cyclic stream from the PLC, but is not as crucial for the context of IPC diagnostics, where, for example, the CPU load is communicated. Event-based streams record data at the time an event occurs. This event can be defined by a condition.

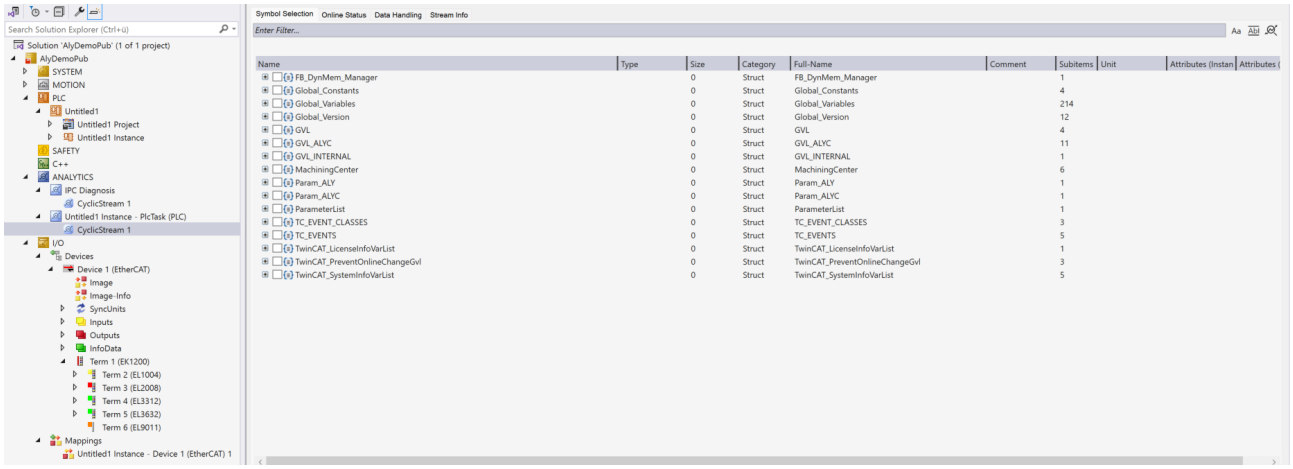
The stream settings are organized into the following tabs.

1. Symbol Selection - depending on the stream type
2. Online Status
3. Data Handling
4. Stream Info



6.3.1 Symbol Selection

In the Symbol Selection tab, you can use the checkbox to select the symbols to be communicated by the Analytics Logger. A tree view of your symbols is generated based on the TwinCAT Target Browser. You can manually browse the tree and choose individual symbols or entire paths or function blocks, structures, programs, etc.



The Target Browser integration has been available since Analytics Logger version 33.2.0. Previously, it was only possible to communicate variables, structures etc. if an attribute was set as pre-compile information in the PLC. The attribute is {attribute 'TcAnalytics'}.
This is no longer mandatory since the version mentioned. You can browse through your project with the Target Browser and select everything by checkbox. However, you can continue to use the attribute and use it as a filter in the Target Browser Integration. This will automatically display the older Analytics Logger versions.

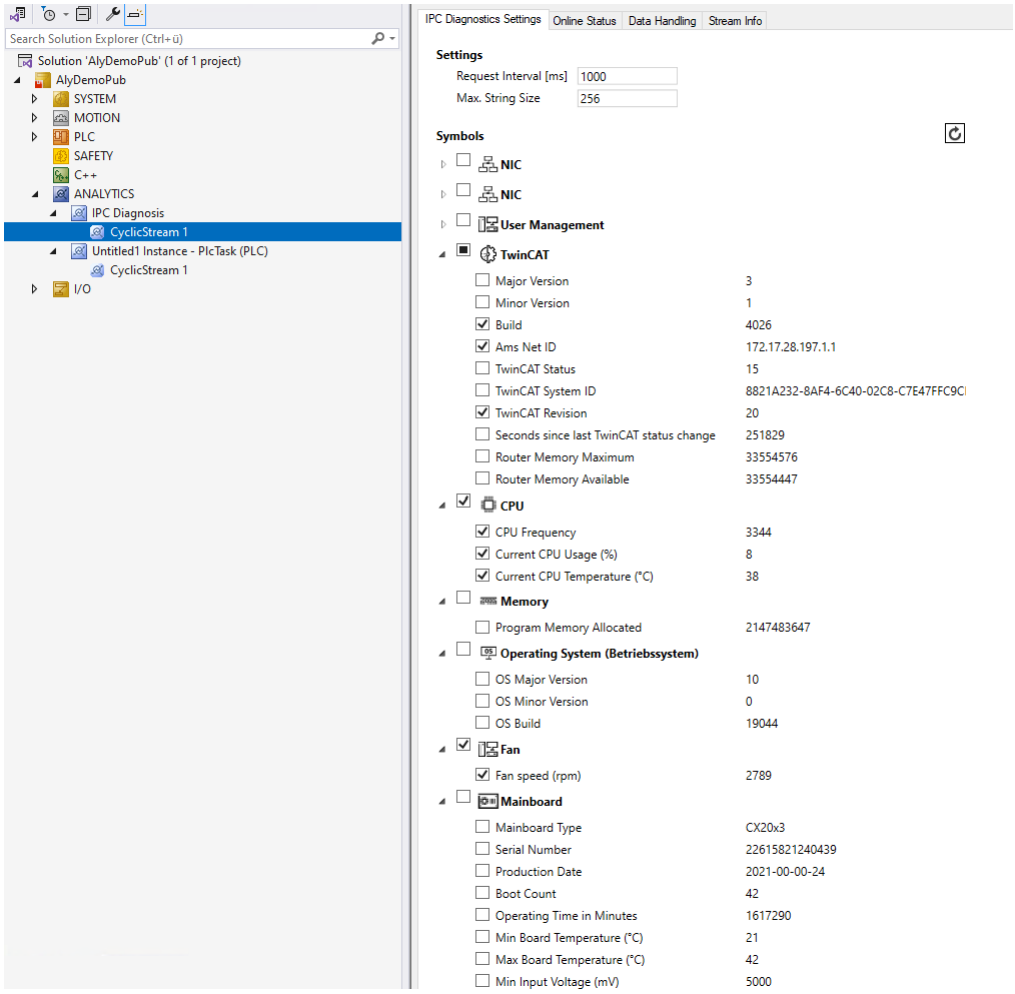
The built-in filter function offers additional functionalities. To reduce the tree view of symbols in a targeted manner, you can use the filter bar and enter name fragments, data types, units etc.

For many standard streams, the symbol selection looks as shown in the screenshot. These are primarily the PLC and I/O streams. Other specific streams may differ from this Target Browser integration in the selection of symbols. For example, the streams of the IPC diagnostics. These streams are therefore described separately in sub-chapters below.

6.3.1.1 IPC diagnostics

If you have added a cyclic stream for the context of the IPC diagnostics, then this is not a cycle-synchronous stream, like a cyclic stream of the PLC. You can therefore specify a request interval in milliseconds at the level of the IPC diagnostics stream. The data is retrieved from the logger via ADS, buffered according to the data handling and sent.

You can use the Refresh button at the top right of the window to manually trigger this data retrieval process and display the current values in the symbol choice table.



As with the other streams, you can choose the symbols using the checkbox. Only the selected ones are collected and sent in the request interval.

6.3.1.2 Vision

Vision Streams allow you to transmit camera images from a TwinCAT Vision application via the Analytics Logger. Instead of using the standard symbol table, select a single image object and specify the desired output format.

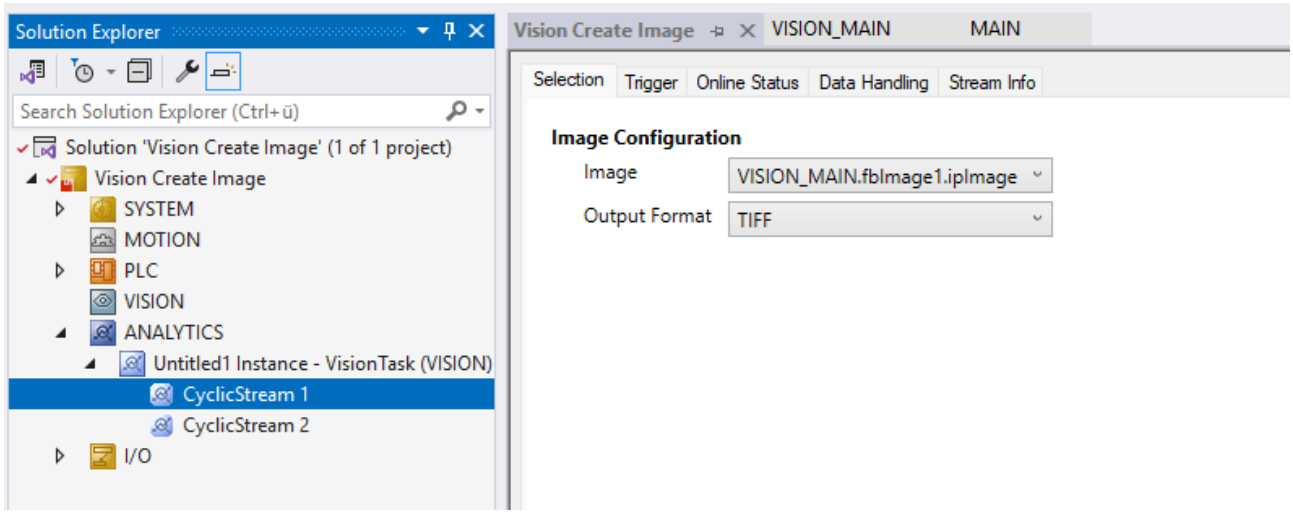


Image configuration

In the **Image Configuration** area, select which camera image you want to record and in what format it should be saved.

Field/Option	Description
Image	Select the image object from the TwinCAT Vision application that is to be communicated by the TwinCAT Analytics Logger. The available images are automatically determined based on the configuration.
Output Format	The output format in which the image is saved. Move your mouse over an entry to display the corresponding tooltip with additional notes.

Available output formats

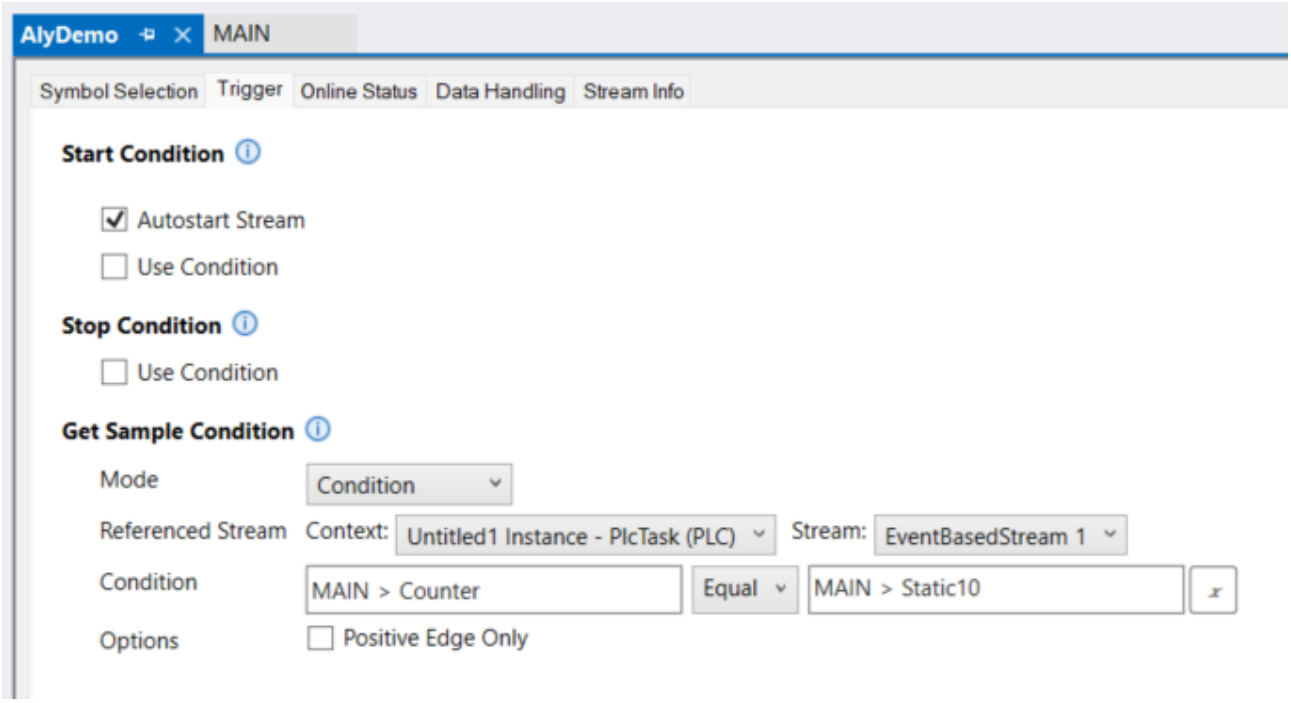
Format	Description
TIFF	Tagged Image File Format. This format provides a stable interface and is suitable for long-term archiving.
BMP	Windows Bitmap. Uncompressed standard format.
TwinCAT Vision	Native TwinCAT Vision format. This format may vary from version to version.

6.3.2 Trigger configuration

Start-Stop-GetSample conditions

The TwinCAT Analytics Logger supports event-based streams in which the start, stop, and acquisition of samples are controlled by configurable conditions. There is a separate **Trigger** tab in the stream settings for this purpose. The start-stop conditions are always available. For both types of conditions, a condition can optionally be enabled. Select the **Use Condition** checkbox to display the Condition Editor.

The **Get Sample Condition** is visible only for event-based streams. It controls when an individual sample is captured.



If the **Autostart Stream** checkbox is selected, the stream starts automatically when the TwinCAT configuration is activated, without requiring a start condition to be met.

Condition Editor

The following options define the event that should trigger the action.

Mode: Here you can choose between the options Condition, On Change, and On Change Ends.

Referenced Stream: The Referenced Stream option allows you to use symbols from a different stream as the basis for the condition. By default, the condition refers to the currently configured stream. To do this, select Context and Stream from the corresponding drop-down lists.

Condition: Configure a condition that triggers the backup. The condition has a left-hand side and, depending on the operator, also a right-hand side.

Constants: To the right of the condition is a button for switching to constant mode. You can use this to enter a constant in the text field.

On Change: In On Change mode, you can configure only the left side of the condition. If the configured variable changes, a backup is triggered.

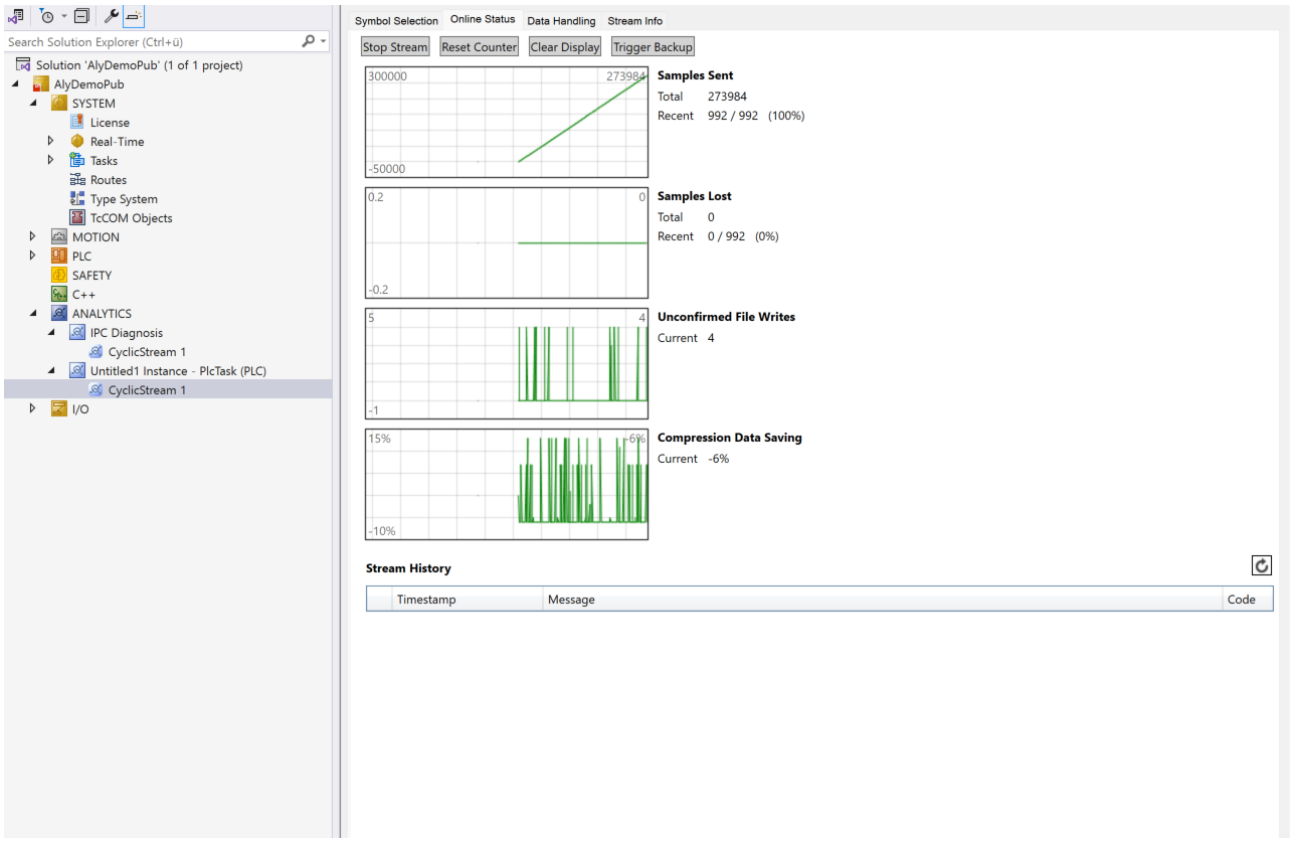
On Change Ends: With the On Change Ends mode, you can configure only the left side of the condition. When the configured variable stops changing, the action is triggered. (Example: 1 2 3 4 4 [Trigger] 4 4 …)

Options: With the Positive Edge Only option, only positive edges are used to trigger the condition. If this option is activated, the Initial Active option can also be selected. With Initial Active, the condition is satisfied at startup.

6.3.3 Online status

An Online Status page is available as a tab for each stream. Here you will find information about the respective stream. You can see whether it is transmitting or whether it is losing samples. All of this is presented graphically. Data compression is also shown in a small chart.

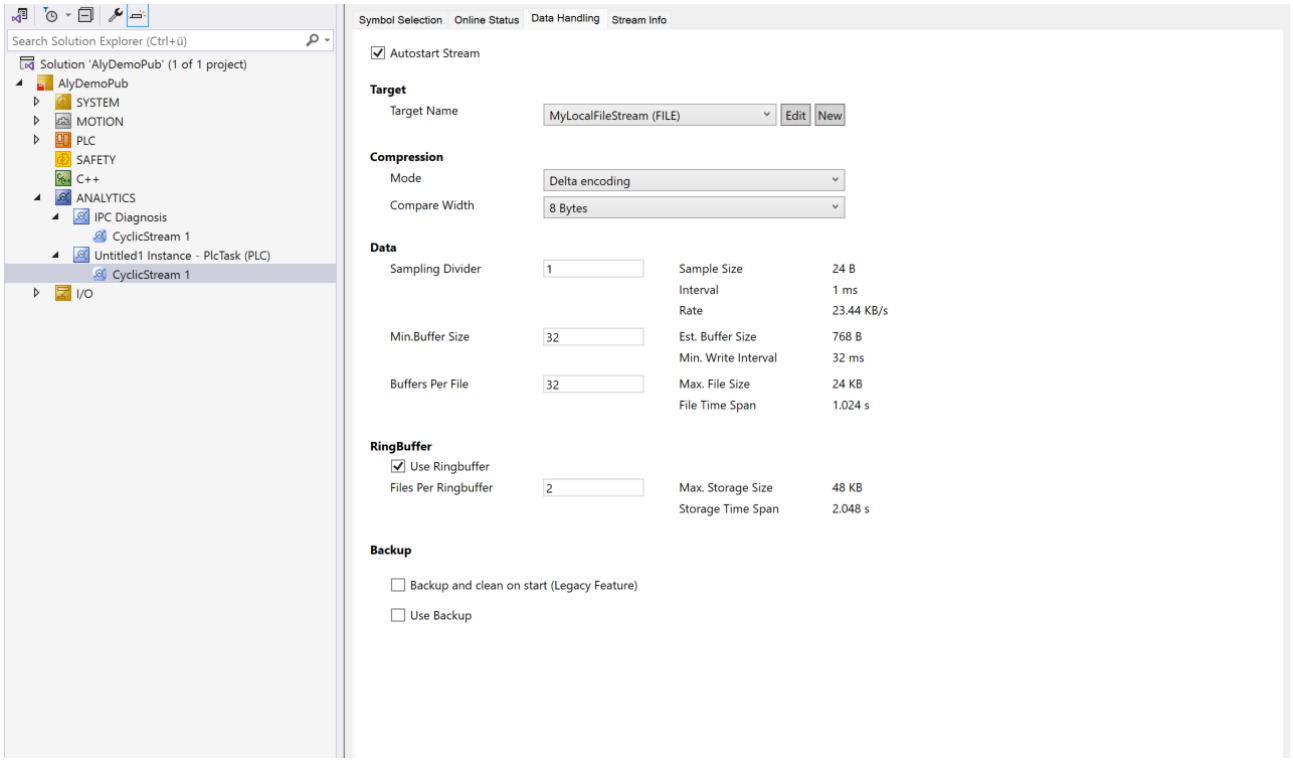
At the top of the page, you can manually start or stop the stream. You can reset all statistics on the page. In the case of a file stream, a potential backup configuration can be triggered manually.



At the bottom of the page, messages are displayed in a stream history from the Analytics Logger driver of the target system. The messages are updated automatically when you switch tabs or can be retrieved manually using the refresh button. Only the last 20 messages are saved in the driver during runtime. After restarting the target system, the old messages are gone and the memory is cleared.

6.3.4 Data Handling

The Data Handling tab once again contains very important setting parameters that are also suitable for fine-tuning, the data storage, and communication of data. For example, general settings for the packet size of the recorded data can be configured. Depending on the selected target (file or MQTT), you will receive additional or different setting parameters.



Some parameters are initially generally applicable.

Autostart Stream: Specifies whether the stream should start automatically as soon as TwinCAT enters Run Mode. If this option is not selected, you can start the stream using a PLC program [▶ 80].

Target:

Target Name: The targets [▶ 53] created in the first logger level (file and MQTT) can be individually assigned here for the actual stream. A target can be used several times in different streams.

Compression:

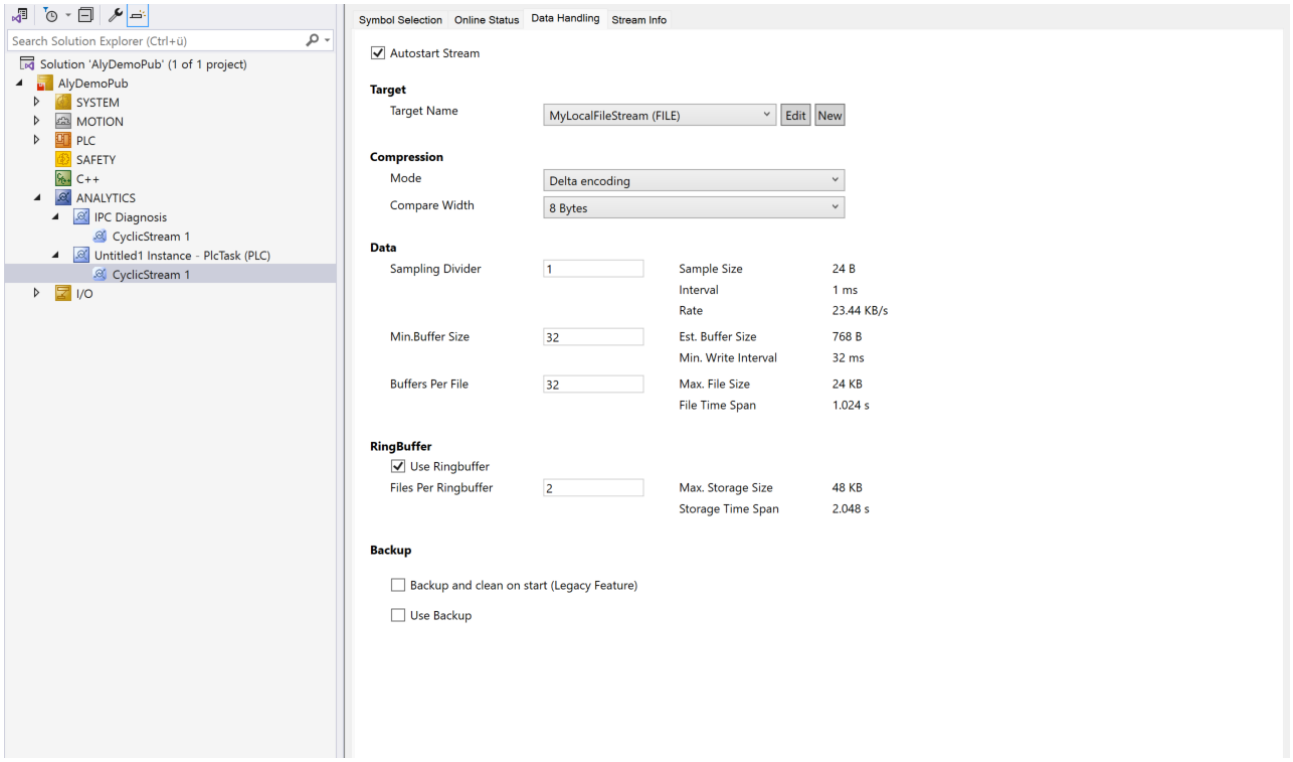
Mode: Set the data compression mode. No compression and so-called delta encoding are currently available. Details on delta encoding can be found in the Data compression [▶ 48] chapter.

Compare Width: Specifies the data comparison bandwidth [▶ 48] in bytes that the logger checks for differences.

The other setting options depend on the selected target and are described further below for File and MQTT.

6.3.4.1 File

If you have chosen a file target for a stream, the data is written by the Analytics Logger to the local hard disk in the directory you have specified. The directory can be set in the Target [▶ 53] description. By default, an Analytics subfolder is created in the TwinCAT boot directory, in which all necessary files [▶ 73] with descriptive and process data are stored.



Data:

Sampling Divider: A value can be set here to ensure that data is only acquired every n cycles. This results in a reduction in the sampling rate. The resulting sampling rate is given by 1/cycle time [s]/n in Hz. Example for 1 ms cycle time and sampling divider 1000: 1/0.001 s/1000 = 1 Hz.

Min Buffer Size: The Min Buffer Size indicates how many samples the logger first buffers before it writes the data to the Analytics .tay file in a new write process. If the utilization is high, the Analytics Logger can automatically increase this value to reduce the load on the file system.

Buffer Per File: Specifies how many buffers the logger writes to a .tay file with the samples it contains before a new file is started.

Ringbuffer:

Use Ringbuffer: Switches the ring buffer function on or off. If the function is activated, a ring buffer is activated on the basis of the files (.tay), in which the oldest files are automatically deleted. However, if the function is switched off, the data is recorded indefinitely. The user is then responsible for managing the files themselves.

Files Per Ringbuffer: Specify the number of files that make up the ring buffer. The maximum storage space required is also specified again.

Backup: Details on the backup mechanism can be found on thisdocumentation page [▶ 74].

Analytics File

Even though the term Analytics File is always used, this is actually just a term for a type of container of different files that the Analytics Logger or the Analytics Storage Provider stores on the file system.

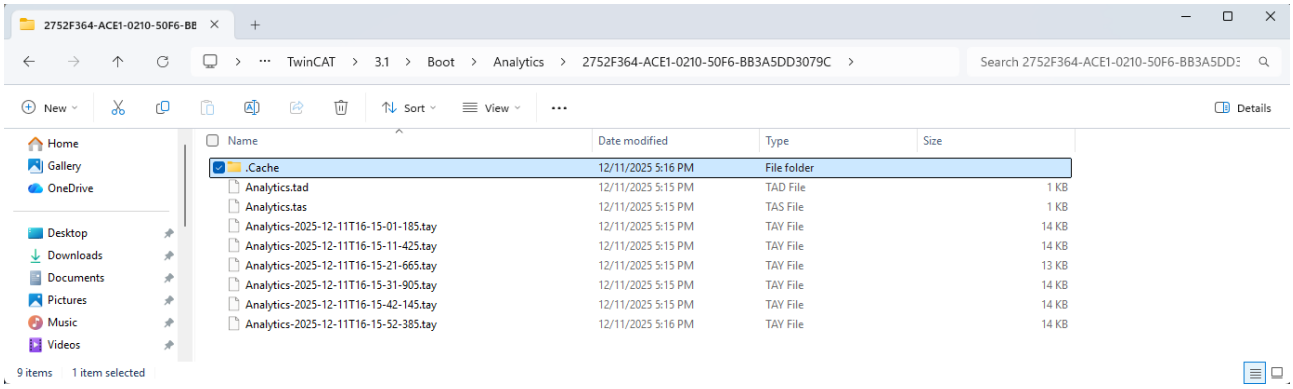
tad: The tad is a Description file. It contains static data in JSON format, such as the address data or the system alias name specified in the Device Info [▶ 60].

tas: The tas file contains the symbolism, i.e. the description of the variables, which is necessary to be able to interpret the binary tay files.

tay: The actual data is stored in a binary format in the tay files (there may be several, depending on the configuration). This data can be interpreted by all TwinCAT Analytics products or the corresponding Analytics APIs and converted into plain text.

Analytics File Cache

If the Analytics Logger is working in file mode, it does not write the data directly to the .tay files, but first stores them as temporary files in a cache folder. This cache is automatically created as a subdirectory in the target directory of the Analytics Logger.



The cache directory ensures that data collection is decoupled from possible access by other tools. The data finalized as .tay can always be queried by other tools such as Target Browser, Analytics Service Tool, etc. The cache files, however, do not. Nevertheless, this is the same format. If the system is shut down, for example, the cache is emptied and the data is converted into .tay files so that no data is lost.

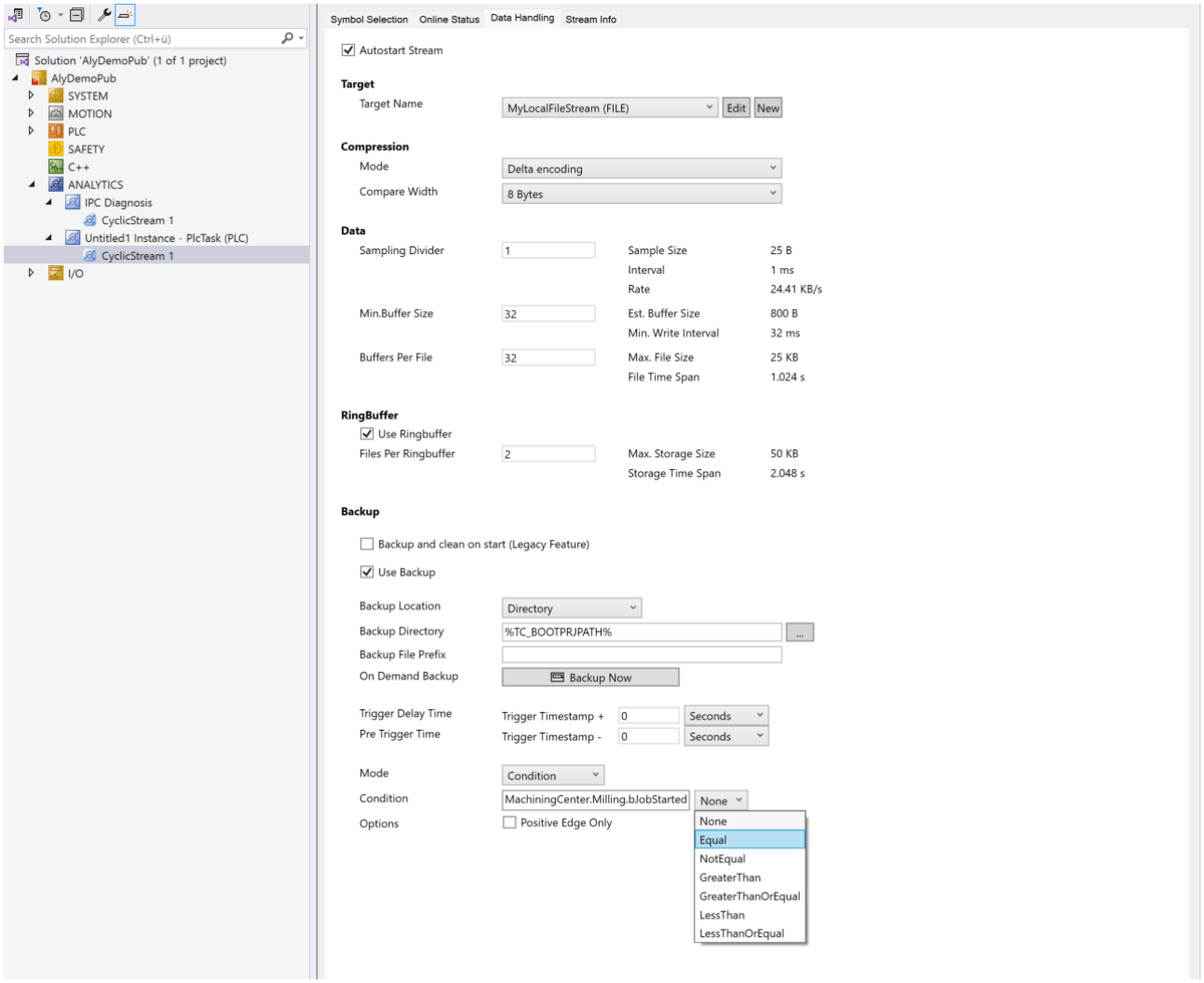
The size of the cache is automatically adjusted based on the Analytics Logger configuration. The user does not have to worry about this.

Also see about this

- Device Info [▶ 60]

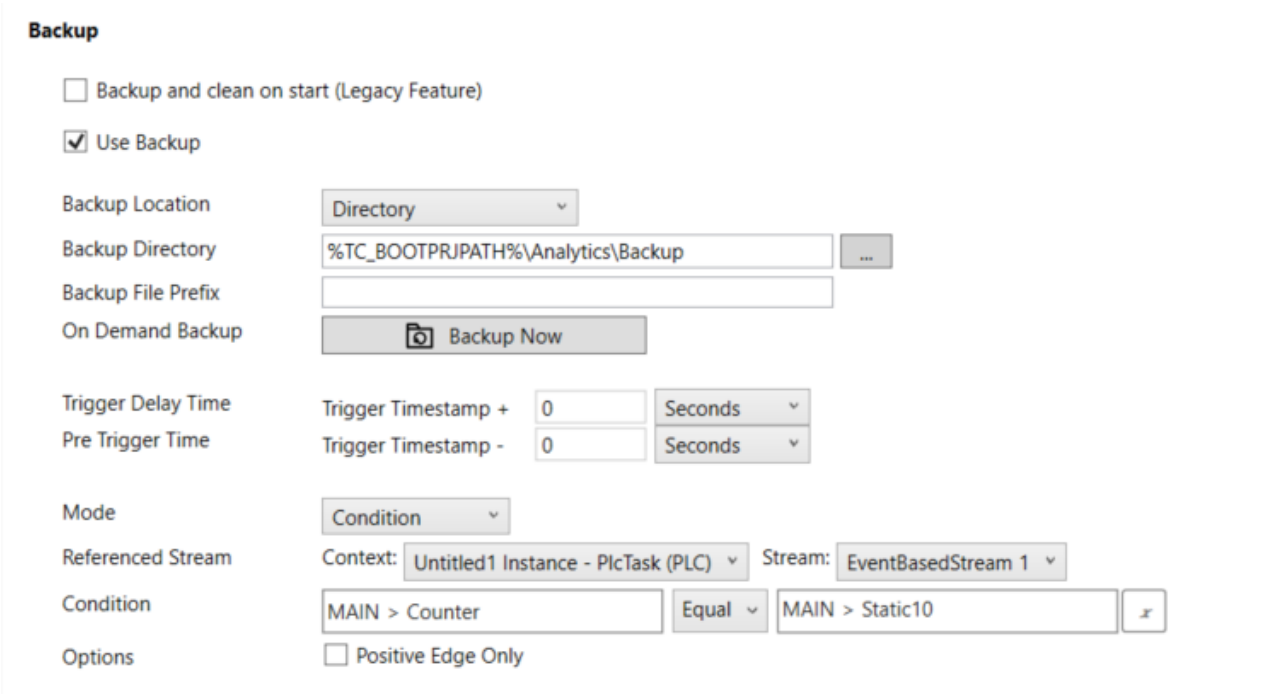
6.3.4.1.1 Backup

You can use the backup functionality in the **Data Handling** tab to persist data in an event-driven manner during the runtime of the stream. The data can also be stored on the hard disk with pre- and post-trigger times. In this case, the user is responsible for cleaning up the data.



The **Backup and clean on start** option refers to an obsolete function that is still listed here for compatibility reasons. If the function is used, the current buffer of the recording is stored in a backup folder when restarting in file mode. After the system is restarted, a new buffer is created until the system is restarted again. The current buffer is then packed back into the backup directory and the old backup is irretrievably deleted.

The backup function available with Analytics Logger version 34.3.0 offers more flexibility. As described above, several backups can be saved on an event basis.



Use Backup: This option must be activated for the following settings to take effect.

Backup Location: Allows you to choose between the Directory and Path options.

Backup Directory / Backup Path: The Backup Directory option allows the choice of a folder path. A further folder level is added below this folder. The folder has the structure “{SymbolHash}-{Prefix}-{Timestamp}”. The backup files are saved in this folder. The Backup Path also allows the choice of a folder path. However, no folder level is created in between and the backup files are stored directly in the configured path.

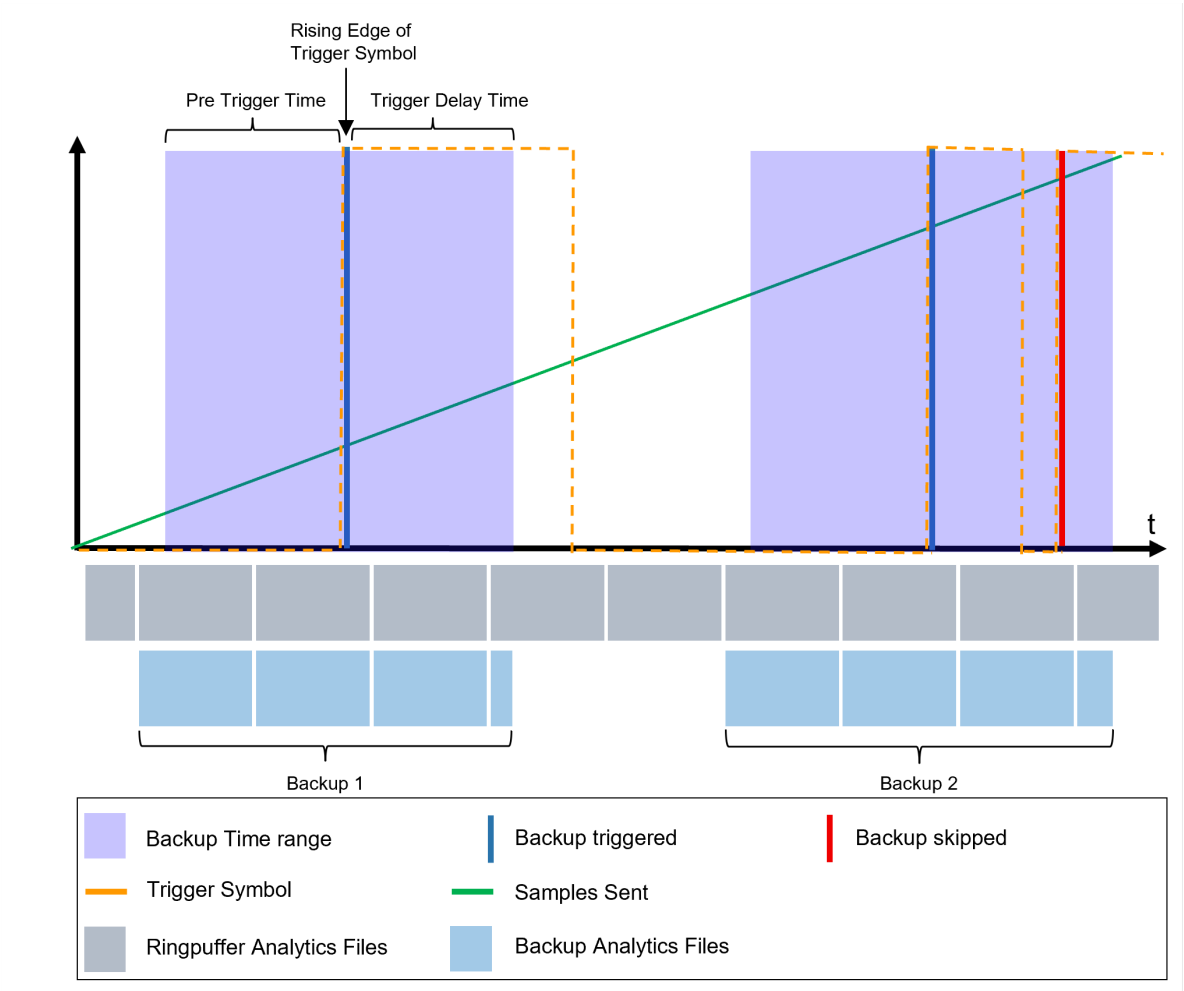
Backup File Prefix: This option is only visible if the Backup Location Directory is selected. The backup file prefix is part of the additional folder level**.**

On Demand Backup: Enables the immediate creation of a backup at the click of a mouse, regardless of the set triggers**.**

Pre Trigger Time: Time period before the trigger event. Backup files that do not contain any data within this time range are removed from the backup.

Trigger Delay Time: Time period after the trigger event. The backup continues to run during this time until it is saved.

The following illustration shows how the Trigger Delay Time and Pre Trigger Time work:



The following options define the event that should trigger the action.

Mode: Here you can choose between the options Condition, On Change, and On Change Ends.

Referenced Stream: The Referenced Stream option allows you to use symbols from a different stream as the basis for the condition. By default, the condition refers to the currently configured stream. To do this, select Context and Stream from the corresponding drop-down lists.

Condition: Configure a condition that triggers the backup. The condition has a left-hand side and, depending on the operator, also a right-hand side.

Constants: To the right of the condition is a button for switching to constant mode. You can use this to enter a constant in the text field.

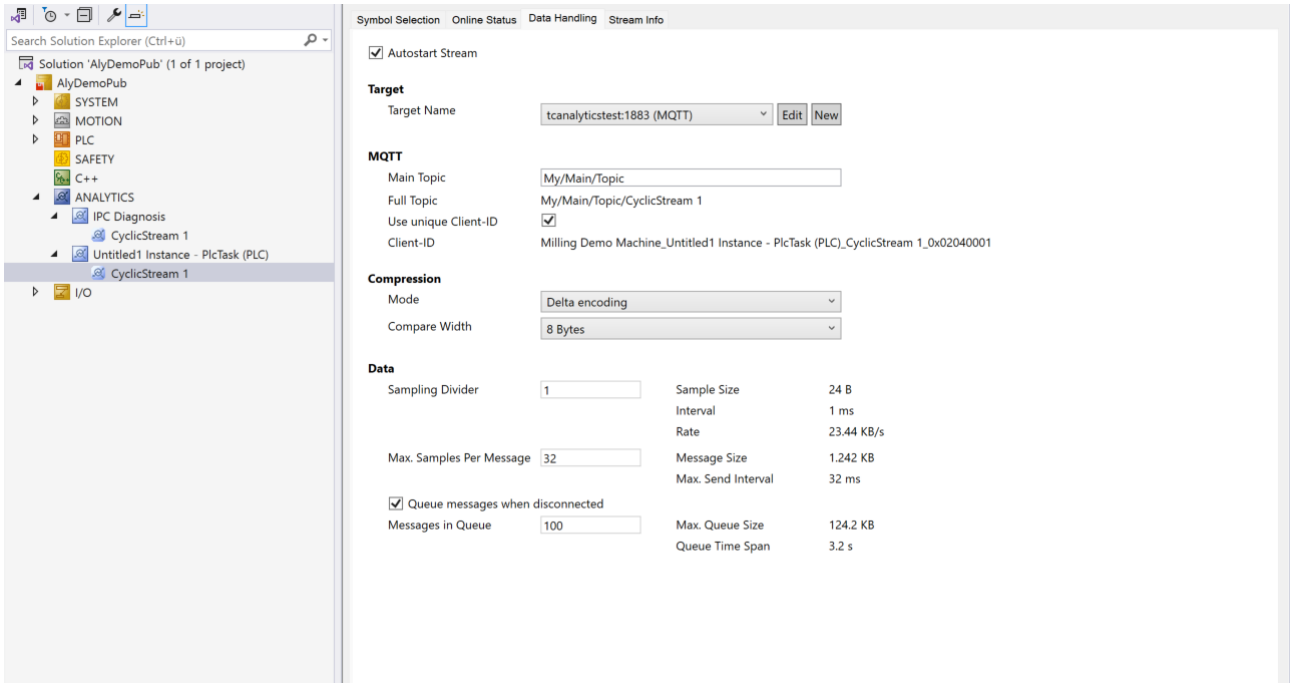
On Change: In On Change mode, you can configure only the left side of the condition. If the configured variable changes, a backup is triggered.

On Change Ends: With the On Change Ends mode, you can configure only the left side of the condition. When the configured variable stops changing, the action is triggered. (Example: 1 2 3 4 4 [Trigger] 4 4 …)

Options: With the Positive Edge Only option, only positive edges are used to trigger the condition. If this option is activated, the Initial Active option can also be selected. With Initial Active, the condition is satisfied at startup.

6.3.4.2 MQTT

If you have chosen an MQTT target for a stream, the data is sent from the Analytics Logger to an MQTT message broker in packets. You can make the settings for the message broker in the target description [▶ 53].



MQTT:

Main Topic: Enter your own unique main topic.
Example: *Beckhoff/Verl/Production/Drives/Machine5*
The Analytics Logger automatically adds its own specific subtopics.

Full Topic: The full topic is only a display for the main topic plus stream name.

Use unique Client-ID: If the checkbox is set, a client ID is automatically assigned. If the option is not chosen, the client ID is enabled and can be entered manually. Even then, it makes sense to choose a unique ID.

Client-ID: A client ID can be entered manually here if the Use unique Client-ID option is not chosen.

Data:

Sampling Divider: A value can be set to ensure that data is only acquired every n cycles. This results in a reduction in the sampling rate. The resulting sampling rate is given by 1/cycle time [s]/n in Hz. Example for 1 ms cycle time and sampling divider 1000: 1/0.001 s/1000 = 1 Hz.

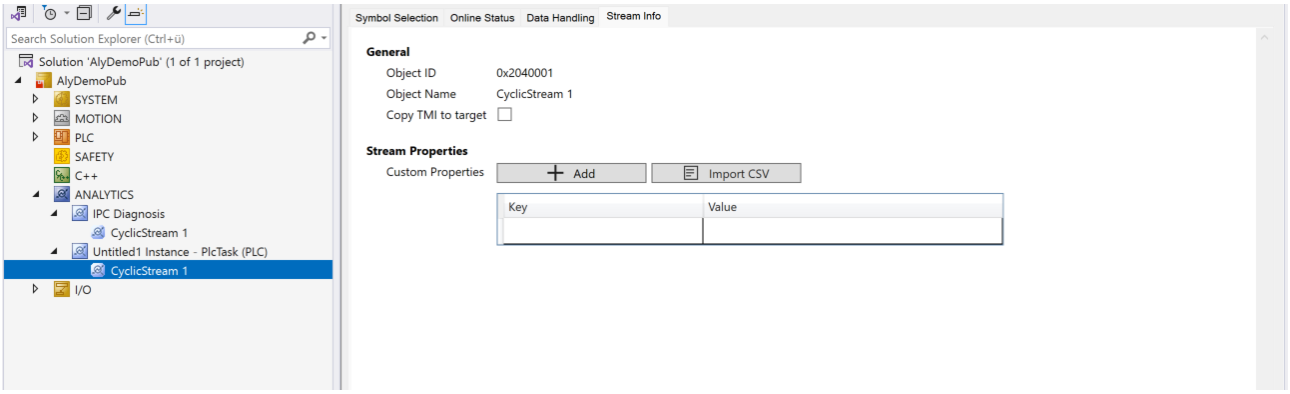
Max. Samples Per Message: The maximum number of samples (cycles) that the Analytics Logger buffers before sending a message via MQTT.

Queue messages when disconnected: If you enable this option, the Analytics Logger automatically detects when the connection to the message broker is interrupted. It then temporarily stores the data locally. As soon as the connection is reestablished, the Analytics Logger sends the stored data.

Messages in Queue: Specify how many messages the Analytics Logger stores in the event of a communication interruption. The Analytics Logger displays the calculated timespan and the associated memory requirements.

6.3.5 Stream Info

There is also an information page on the third level in the TwinCAT Analytics Logger. You will find the object ID and the object name here. You can use the checkbox to decide whether the TMI should also be downloaded to the target system.



You can use the **Duplicate Stream** button (copy icon) in the **General** section to duplicate the current stream, including all settings. The new stream is created below the same context. You can then customize it.

It is also possible to store additional static information at this point using key-value pairs. These can also be imported externally via CSV import.

7 API

7.1 PLC

7.1.1 Analytics Communication Library

7.1.1.1 Overview

The TwinCAT Analytics Communication Library is a PLC library that provides the user with an interface to the Analytics Logger and its counterpart, the Analytics Stream Helper. This enables you to start, stop and reconfigure an Analytics Stream from the PLC code during runtime. For example, you can change the endpoint, a directory or a message broker from within the application. This opens up many new application possibilities for your Analytics application.

Product components

- Driver TcAnalytics.sys
- PLC library Tc3AnalyticsCommunication.compiled-library

7.1.1.2 Installation

The TwinCAT Analytics Communication Library is installed with TwinCAT XAE and XAR version >= 4024.47. It should therefore always be available. No separate license is required to use it, but the Analytics Logger requires a TF3500 TwinCAT 3 Analytics Logger license, either perpetual or as a 7-day license.

7.1.1.3 PLC API

7.1.1.3.1 Function blocks

7.1.1.3.1.1 FB_ALYC_MqttStream

This function block represents an Analytics MQTT stream. The connection to a stream can be established using the ObjectId as an input variable. The requirement is an existing stream, e.g. created in the System Manager under the Data Logger node in the project tree. The stream can be controlled using the Start/Stop methods and reconfigured using Reconfigure. A structure of the type ST_ALYC_MqttStreamConfig containing the new configuration parameters is transferred to the method for this purpose. During reconfiguration, starting from the OP state, the TcCom states SAFEOP, PREOP, SAFEOP, OP are run through in this order. Since all states below SAFEOP no longer run in real-time mode, but the remaining TwinCAT runtime does, the reconfiguration must take place asynchronously to the task cycle, whereby the Reconfig method should be called cyclically as long as the OP state is not reached again. The properties bConnected, bStarted etc. provide information about the current state of the stream. Errors can be recorded via the bError output and the corresponding ipResultMessage.

Definition:

```
FUNCTION_BLOCK FB_ALYC_MqttStream
VAR_INPUT
    {attribute 'tcinitsymbol'}
    nObjectID : OTCID := 0;
END_VAR
VAR_OUTPUT
    bInitialized : BOOL := FALSE;
    bError : BOOL := FALSE;
    ipResultMessage : I_TcMessage := fbResult;
    eReconfigState : E_ALYC_ReconfigState := E_ALYC_ReconfigState.DONE;
END_VAR
```

Inputs

Name	Type	Description
nObjectID	OTCID	TcCom-Object ID of the referenced stream. This can be initialized under the 'Init symbols' tab of the corresponding PLC instance node in the System Manager project tree.

Outputs

Name	Type	Description
bInitialized	BOOL	TRUE if the function block is initialized and can be used. Initialization takes place automatically after setting TwinCAT to Run Mode.
bError	BOOL	TRUE if an error has occurred.
ipResultMessage	I_TcMessage	Message EventLogger
eReconfigState	E_ALYC_ReconfigState	The state of the state machine during reconfiguration.

Methods

Name	Return type	Description
Reconfigure	BOOL	Reconfigure the stream. Parameter: ST_ALYC_MqttStreamConfig. TRUE if successful.
Start	BOOL	Start the stream. TRUE if successful.
Stop	BOOL	Stop the stream. TRUE if successful.

Properties

Name	Type	Direction	Description
bConnected	BOOL	get	TRUE if MQTT connection exists.
bStarted	BOOL	get	TRUE if stream started.
nDataSize	UDINT	get	Sample data size
nSamplesIssued	ULINT	get	Number of samples that have been written.
nSamplesLost	ULINT	get	Number of discarded samples.
tCycleTime	LTIME	get	Cycle time in ns
nCompDataSaving	DINT	get	Percentage of data volume saved compared to the uncompressed alternative. If negative, there is additional work instead of data savings.

7.1.1.3.1.2 FB_ALYC_FileStream

This function block represents an Analytics Stream in file mode. The connection to a stream can be established using the ObjectId as an input variable. The requirement is an existing stream that is in file mode, e.g. created in the System Manager under the Data Logger node in the project tree. The symbols that are to be logged must also be configured via the System Manager.

The stream can be controlled using the Start/Stop methods and reconfigured using Reconfigure. A structure of the type ST_ALYC_FileStreamConfig containing the new configuration parameters is transferred to the method for this purpose. During reconfiguration, starting from the OP state, the TcCom states SAFEOP, PREOP, SAFEOP, OP are run through in this order. Since all states below SAFEOP no longer operate in real-time mode, but the remaining TwinCAT runtime does, the reconfiguration must take place asynchronously to the task cycle, whereby the Reconfigure method should be called cyclically as long as the OP state is not reached again. The properties bStarted, nSampleIssued etc. provide information about the current state of the stream. Errors can be recorded via the bError output and the corresponding ipResultMessage.

Definition:

```
FUNCTION_BLOCK FB_ALYC_FileStream
VAR_INPUT
     {attribute 'tcinitsymbol'}
     nObjectID : OTCID := 0;
END_VAR
VAR_OUTPUT
     bInitialized : BOOL := FALSE;
     bError : BOOL := FALSE;
     ipResultMessage : I_TcMessage := fbResult;
     eReconfigState : E_ALYC_ReconfigState := E_ALYC_ReconfigState.DONE;
END_VAR
```

Inputs

Name	Type	Description
nObjectID	OTCID	TcCom-Object ID of the referenced stream. This can be initialized under the 'Init symbols' tab of the corresponding PLC instance node in the System Manager project tree.

Outputs

Name	Type	Description
bInitialized	BOOL	TRUE if the function block is initialized and can be used. Initialization takes place automatically after setting TwinCAT to Run Mode.
bError	BOOL	TRUE if an error has occurred.
ipResultMessage	I_TcMessage	Message EventLogger
eReconfigState	E_ALYC_ReconfigState	The state of the state machine during reconfiguration.

Methods

Name	Return type	Description
Reconfigure	BOOL	Reconfigure the stream. Parameter: ST_ALYC_FileStreamConfig. TRUE if successful.
Start	BOOL	Start the stream. TRUE if successful.
Stop	BOOL	Stop the stream. TRUE if successful.

Properties

Name	Type	Direction	Description
bStarted	BOOL	get	TRUE if stream started.
nDataSize	UDINT	get	Sample data size
nSamplesIssued	ULINT	get	Number of samples that have been written.
nSamplesLost	ULINT	get	Number of discarded samples.
tCycleTime	LTIME	get	Cycle time in ns
nCompDataSaving	DINT	get	Percentage of data volume saved compared to the uncompressed alternative. If negative, there is additional work instead of data savings.
nUnconfFileWrites	ULINT	get	Number of file write requests that have not yet been confirmed by the AMS router. Can prevent an overflow of the router message queue.

7.1.1.3.1.3 FB_ALYC_MqttStreamHelper

This function block represents an Analytics Stream Helper in MQTT mode. The connection to an existing Stream Helper can be established via the ObjectId as an input variable. This must be configured in MQTT mode. The stream can be controlled using the Start/Stop methods and reconfigured using Reconfigure. For this purpose, a structure of the type ST_ALYC_MqttStreamHelperConfig is transferred to the method, which contains the new configuration parameters. During reconfiguration, starting from the OP state, the TcCom states SAFEOP, PREOP, SAFEOP, OP are run through in this order. Since all states below SAFEOP no longer run in real-time mode, but the remaining TwinCAT runtime does, the reconfiguration must take place asynchronously to the task cycle, whereby the Reconfigure method should be called cyclically as long as the OP state is not reached again. The properties bConnected, bStarted etc. provide information about the current state of the Stream Helper. Errors can be recorded via the bError output and the corresponding ipResultMessage.

Definition:

```
FUNCTION_BLOCK FB_ALYC_MqttStreamHelper
VAR_INPUT
     {attribute 'tcinitsymbol'}
     nObjectID : OTCID := 0;
     nNumInputBuffer : UDINT := 20;
END_VAR
VAR_OUTPUT
     ipResultMessage : I_TcMessage := fbResult;
     bError : BOOL := FALSE;
     bNewResult : BOOL := FALSE;
     bInitialized : BOOL := FALSE;
     nNumElements : UDINT;
     eReconfigState : E_ALYC_ReconfigState := E_ALYC_ReconfigState.DONE;
END_VAR
```

Inputs

Name	Type	Description
nObjectID	OTCID	TcCom-Object ID of the referenced StreamHelper. This can be initialized under the 'Init symbols' tab of the corresponding PLC instance node in the System Manager project tree.
nNumInputBuffer	UDINT	Maximum number of buffered symbol values (samples) in the symbol queues.

Outputs

Name	Type	Description
bInitialized	BOOL	TRUE if the function block is initialized and can be used. Initialization takes place automatically after setting TwinCAT to Run Mode.
bError	BOOL	TRUE if an error has occurred.
ipResultMessage	I_TcMessage	Message EventLogger
eReconfigState	E_ALYC_ReconfigState	The state of the state machine during reconfiguration.
bNewResult	BOOL	TRUE if new values have been read into the symbol queues.
nNumElements	UDINT	Number of new values in the symbol queues

Methods

Name	Return type	Description
Reconfigure	BOOL	Reconfigure the Stream Helper with a ST_ALYC_MqttStreamHelperConfig as parameter. Returns TRUE if successful.
Call	BOOL	Main method that should always be called cyclically. TRUE if successful.
AddIotSymbol	BOOL	Add a symbol of type I_ALYC_IotSymbol to the internal symbol list whose values are to be received. TRUE if successful.
ReleaseIotSymbol	BOOL	Remove a symbol of type I_ALYC_IotSymbol from the internal symbol list. TRUE if successful.
ReleaseAllIotSymbols	BOOL	TRUE if successful.
ContainsIotSymbol	BOOL	TRUE if symbol of type I_ALYC_IotSymbol is in the internal symbol list.

Properties

Name	Type	Direction	Description
bConnected	BOOL	get	TRUE if an MQTT connection exists.
bReconnect	BOOL	get/set	If TRUE, interrupt the MQTT connection, if FALSE, renew the MQTT connection.
sStream	STRING(255)	get/set	MQTT receive topic in the format <MainTopic>/<StreamTopic>
nNumIotSymbolsRegistered	UDINT	get/set	Number of Iot symbols added.

7.1.1.3.1.4 FB_ALYC_FileStreamHelper

This function block represents an Analytics Stream Helper in file mode. The connection to an existing Stream Helper can be established via the ObjectId as an input variable. This must be configured in file mode. The stream can be controlled using the Start/Stop methods and reconfigured using Reconfigure. For this purpose, a structure of the type ST_ALYC_FileStreamHelperConfig is transferred to the method, which contains the new configuration parameters. During reconfiguration, starting from the OP state, the TcCom states SAFEOP, PREOP, SAFEOP, OP are run through in this order. Since all states below SAFEOP no longer run in real-time mode, but the remaining TwinCAT runtime does, the reconfiguration must take place asynchronously to the task cycle, whereby the Reconfigure method should be called cyclically as long as the OP state is not reached again. The properties bStarted etc. provide information about the current state of the Stream Helper. Errors can be recorded via the bError output and the corresponding ipResultMessage.

Definition:

```
FUNCTION_BLOCK FB_ALYC_MqttStreamHelper
VAR_INPUT
    {attribute 'tcinitsymbol'}
    nObjectID : OTCID := 0;
    nNumInputBuffer : UDINT := 20;
END_VAR
VAR_OUTPUT
    ipResultMessage : I_TcMessage := fbResult;
    bError : BOOL := FALSE;
    bNewResult : BOOL := FALSE;
    bInitialized : BOOL := FALSE;
    nNumElements : UDINT;
    eReconfigState : E_ALYC_ReconfigState := E_ALYC_ReconfigState.DONE;
END_VAR
```

Inputs

Name	Type	Description
nObjectID	OTCID	TcCom-Object ID of the referenced StreamHelper. This can be initialized under the 'Init symbols' tab of the corresponding PLC instance node in the System Manager project tree.
nNumInputBuffer	UDINT	Maximum number of buffered symbol values (samples) in the symbol queues.

Outputs

Name	Type	Description
bInitialized	BOOL	TRUE if the function block is initialized and can be used. Initialization takes place automatically after setting TwinCAT to Run Mode.
bError	BOOL	TRUE if an error has occurred.
ipResultMessage	I_TcMessage	Message EventLogger
eReconfigState	E_ALYC_ReconfigState	The state of the state machine during reconfiguration.
bNewResult	BOOL	TRUE if new values have been read into the symbol queues.
nNumElements	UDINT	Number of new values in the symbol queues
stCurrentConfig	ST_ALYC_FileStreamHelperConfig [▶ 102]	
stCurrentState	ST_ALYC_FileStreamHelperState	Current state information on the Stream Helper.

Methods

Name	Return type	Description
Reconfigure	BOOL	Reconfigure the Stream Helper with a ST_ALYC_MqttStreamHelperConfig as parameter. Returns TRUE if successful.
Call	BOOL	Main method that should always be called cyclically. TRUE if successful.
AddIotSymbol	BOOL	Add a symbol of type I_ALYC_IotSymbol to the internal symbol list whose values are to be received. TRUE if successful.
ReleaseIotSymbol	BOOL	Remove a symbol of type I_ALYC_IotSymbol from the internal symbol list. TRUE if successful.
ReleaseAllIotSymbols	BOOL	TRUE if successful.
ContainsIotSymbol	BOOL	TRUE if symbol of type I_ALYC_IotSymbol is in the internal symbol list.

Properties

Name	Type	Direction	Description
nNumIotSymbolsRegistered	UDINT	get/set	Number of Iot symbols added.

7.1.1.3.1.5 IoT symbol

7.1.1.3.1.5.1 FB_ALYC_IotSymbol_BOOL

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALYC_IotSymbol_BOOL
VAR_INPUT
    stConfig : ST_ALYC_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALYC_IotSymbol_Config	Structure for the configuration of the FB.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	EventLogger
bError	BOOL	TRUE if an error has occurred.
bNewResult	BOOL	TRUE if a new result has been calculated.
bConfigured	BOOL	TRUE if the FB is successfully configured.
bSymbolHandlerAssigned	BOOL	TRUE if the symbol handler has been assigned.
bVariableFound	BOOL	TRUE if a variable was found in the stream.
sSymbolPath	STRING(255)	Theam/Stream.Symbol
tCycleTime	LTIME	Cycle time of the publishing system.
nMaxNumElements	UDINT	Maximum number of saved symbols influenced by StreamHelper.

Methods

Name	Definition location	Description
GetValue	Local	Get the value of the specified element.
GetOversamplingValues	Local	Get the oversampling values of the specified element.
GetArrayValues	Local	Get the array values of the specified element.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

7.1.1.3.1.5.2 FB_ALYC_IotSymbol_BYTE

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALYC_IotSymbol_BYTE
VAR_INPUT
    stConfig : ST_ALYC_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALYC_IotSymbol_Config	Structure for the configuration of the FB.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	EventLogger
bError	BOOL	TRUE if an error has occurred.
bNewResult	BOOL	TRUE if a new result has been calculated.
bConfigured	BOOL	TRUE if the FB is successfully configured.
bSymbolHandlerAssigned	BOOL	TRUE if the symbol handler has been assigned.
bVariableFound	BOOL	TRUE if a variable was found in the stream.
sSymbolPath	STRING(255)	Theam/Stream.Symbol
tCycleTime	LTIME	Cycle time of the publishing system.
nMaxNumElements	UDINT	Maximum number of saved symbols influenced by StreamHelper.

Methods

Name	Definition location	Description
GetValue	Local	Get the value of the specified element.
GetOversamplingValues	Local	Get the oversampling values of the specified element.
GetArrayValues	Local	Get the array values of the specified element.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

7.1.1.3.1.5.3 FB_ALYC_IotSymbol_DINT

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALYC_IotSymbol_DINT
VAR_INPUT
    stConfig : ST_ALYC_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALYC_IotSymbol_Config	Structure for the configuration of the FB.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	EventLogger
bError	BOOL	TRUE if an error has occurred.
bNewResult	BOOL	TRUE if a new result has been calculated.
bConfigured	BOOL	TRUE if the FB is successfully configured.
bSymbolHandlerAssigned	BOOL	TRUE if the symbol handler has been assigned.
bVariableFound	BOOL	TRUE if a variable was found in the stream.
sSymbolPath	STRING(255)	Theam/Stream.Symbol
tCycleTime	LTIME	Cycle time of the publishing system.
nMaxNumElements	UDINT	Maximum number of saved symbols influenced by StreamHelper.

Methods

Name	Definition location	Description
GetValue	Local	Get the value of the specified element.
GetOversamplingValues	Local	Get the oversampling values of the specified element.
GetArrayValues	Local	Get the array values of the specified element.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

7.1.1.3.1.5.4 FB_ALYC_IotSymbol_DWORD

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALYC_IotSymbol_DWORD
VAR_INPUT
    stConfig : ST_ALYC_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALYC_IotSymbol_Config	Structure for the configuration of the FB.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	EventLogger
bError	BOOL	TRUE if an error has occurred.
bNewResult	BOOL	TRUE if a new result has been calculated.
bConfigured	BOOL	TRUE if the FB is successfully configured.
bSymbolHandlerAssigned	BOOL	TRUE if the symbol handler has been assigned.
bVariableFound	BOOL	TRUE if a variable was found in the stream.
sSymbolPath	STRING(255)	Theam/Stream.Symbol
tCycleTime	LTIME	Cycle time of the publishing system.
nMaxNumElements	UDINT	Maximum number of saved symbols influenced by StreamHelper.

Methods

Name	Definition location	Description
GetValue	Local	Get the value of the specified element.
GetOversamplingValues	Local	Get the oversampling values of the specified element.
GetArrayValues	Local	Get the array values of the specified element.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

7.1.1.3.1.5.5 FB_ALYC_IotSymbol_INT

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALYC_IotSymbol_INT
VAR_INPUT
    stConfig : ST_ALYC_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALYC_IotSymbol_Config	Structure for the configuration of the FB.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	EventLogger
bError	BOOL	TRUE if an error has occurred.
bNewResult	BOOL	TRUE if a new result has been calculated.
bConfigured	BOOL	TRUE if the FB is successfully configured.
bSymbolHandlerAssigned	BOOL	TRUE if the symbol handler has been assigned.
bVariableFound	BOOL	TRUE if a variable was found in the stream.
sSymbolPath	STRING(255)	Theam/Stream.Symbol
tCycleTime	LTIME	Cycle time of the publishing system.
nMaxNumElements	UDINT	Maximum number of saved symbols influenced by StreamHelper.

Methods

Name	Definition location	Description
GetValue	Local	Get the value of the specified element.
GetOversamplingValues	Local	Get the oversampling values of the specified element.
GetArrayValues	Local	Get the array values of the specified element.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

7.1.1.3.1.5.6 FB_ALYC_IotSymbol_LINT

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALYC_IotSymbol_LINT
VAR_INPUT
    stConfig : ST_ALYC_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALYC_IotSymbol_Config	Structure for the configuration of the FB.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	EventLogger
bError	BOOL	TRUE if an error has occurred.
bNewResult	BOOL	TRUE if a new result has been calculated.
bConfigured	BOOL	TRUE if the FB is successfully configured.
bSymbolHandlerAssigned	BOOL	TRUE if the symbol handler has been assigned.
bVariableFound	BOOL	TRUE if a variable was found in the stream.
sSymbolPath	STRING(255)	Theam/Stream.Symbol
tCycleTime	LTIME	Cycle time of the publishing system.
nMaxNumElements	UDINT	Maximum number of saved symbols influenced by StreamHelper.

Methods

Name	Definition location	Description
GetValue	Local	Get the value of the specified element.
GetOversamplingValues	Local	Get the oversampling values of the specified element.
GetArrayValues	Local	Get the array values of the specified element.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

7.1.1.3.1.5.7 FB_ALYC_IotSymbol_LREAL

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALYC_IotSymbol_LREAL
VAR_INPUT
    stConfig : ST_ALYC_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALYC_IotSymbol_Config	Structure for the configuration of the FB.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	EventLogger
bError	BOOL	TRUE if an error has occurred.
bNewResult	BOOL	TRUE if a new result has been calculated.
bConfigured	BOOL	TRUE if the FB is successfully configured.
bSymbolHandlerAssigned	BOOL	TRUE if the symbol handler has been assigned.
bVariableFound	BOOL	TRUE if a variable was found in the stream.
sSymbolPath	STRING(255)	Theam/Stream.Symbol
tCycleTime	LTIME	Cycle time of the publishing system.
nMaxNumElements	UDINT	Maximum number of saved symbols influenced by StreamHelper.

Methods

Name	Definition location	Description
GetValue	Local	Get the value of the specified element.
GetOversamplingValues	Local	Get the oversampling values of the specified element.
GetArrayValues	Local	Get the array values of the specified element.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

7.1.1.3.1.5.8 FB_ALYC_IotSymbol_LWORD

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALYC_IotSymbol_LWORD
VAR_INPUT
    stConfig : ST_ALYC_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALYC_IotSymbol_Config	Structure for the configuration of the FB.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	EventLogger
bError	BOOL	TRUE if an error has occurred.
bNewResult	BOOL	TRUE if a new result has been calculated.
bConfigured	BOOL	TRUE if the FB is successfully configured.
bSymbolHandlerAssigned	BOOL	TRUE if the symbol handler has been assigned.
bVariableFound	BOOL	TRUE if a variable was found in the stream.
sSymbolPath	STRING(255)	Theam/Stream.Symbol
tCycleTime	LTIME	Cycle time of the publishing system.
nMaxNumElements	UDINT	Maximum number of saved symbols influenced by StreamHelper.

Methods

Name	Definition location	Description
GetValue	Local	Get the value of the specified element.
GetOversamplingValues	Local	Get the oversampling values of the specified element.
GetArrayValues	Local	Get the array values of the specified element.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

7.1.1.3.1.5.9 FB_ALYC_IotSymbol_REAL

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALYC_IotSymbol_REAL
VAR_INPUT
    stConfig : ST_ALYC_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALYC_IotSymbol_Config	Structure for the configuration of the FB.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	EventLogger
bError	BOOL	TRUE if an error has occurred.
bNewResult	BOOL	TRUE if a new result has been calculated.
bConfigured	BOOL	TRUE if the FB is successfully configured.
bSymbolHandlerAssigned	BOOL	TRUE if the symbol handler has been assigned.
bVariableFound	BOOL	TRUE if a variable was found in the stream.
sSymbolPath	STRING(255)	Theam/Stream.Symbol
tCycleTime	LTIME	Cycle time of the publishing system.
nMaxNumElements	UDINT	Maximum number of saved symbols influenced by StreamHelper.

Methods

Name	Definition location	Description
GetValue	Local	Get the value of the specified element.
GetOversamplingValues	Local	Get the oversampling values of the specified element.
GetArrayValues	Local	Get the array values of the specified element.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

7.1.1.3.1.5.10 FB_ALYC_IotSymbol_SINT

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALYC_IotSymbol_SINT
VAR_INPUT
    stConfig : ST_ALYC_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALYC_IotSymbol_Config	Structure for the configuration of the FB.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	EventLogger
bError	BOOL	TRUE if an error has occurred.
bNewResult	BOOL	TRUE if a new result has been calculated.
bConfigured	BOOL	TRUE if the FB is successfully configured.
bSymbolHandlerAssigned	BOOL	TRUE if the symbol handler has been assigned.
bVariableFound	BOOL	TRUE if a variable was found in the stream.
sSymbolPath	STRING(255)	Theam/Stream.Symbol
tCycleTime	LTIME	Cycle time of the publishing system.
nMaxNumElements	UDINT	Maximum number of saved symbols influenced by StreamHelper.

Methods

Name	Definition location	Description
GetValue	Local	Get the value of the specified element.
GetOversamplingValues	Local	Get the oversampling values of the specified element.
GetArrayValues	Local	Get the array values of the specified element.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

7.1.1.3.1.5.11 FB_ALYC_IotSymbol_STRING

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALYC_IotSymbol_STRING
VAR_INPUT
    stConfig : ST_ALYC_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALYC_IotSymbol_Config	Structure for the configuration of the FB.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	EventLogger
bError	BOOL	TRUE if an error has occurred.
bNewResult	BOOL	TRUE if a new result has been calculated.
bConfigured	BOOL	TRUE if the FB is successfully configured.
bSymbolHandlerAssigned	BOOL	TRUE if the symbol handler has been assigned.
bVariableFound	BOOL	TRUE if a variable was found in the stream.
sSymbolPath	STRING(255)	Theam/Stream.Symbol
tCycleTime	LTIME	Cycle time of the publishing system.
nMaxNumElements	UDINT	Maximum number of saved symbols influenced by StreamHelper.

Methods

Name	Definition location	Description
GetValue	Local	Get the value of the specified element.
GetOversamplingValues	Local	Get the oversampling values of the specified element.
GetArrayValues	Local	Get the array values of the specified element.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

7.1.1.3.1.5.12 FB_ALYC_IotSymbol_UDINT

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALYC_IotSymbol_UDINT
VAR_INPUT
    stConfig : ST_ALYC_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALYC_IotSymbol_Config	Structure for the configuration of the FB.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	EventLogger
bError	BOOL	TRUE if an error has occurred.
bNewResult	BOOL	TRUE if a new result has been calculated.
bConfigured	BOOL	TRUE if the FB is successfully configured.
bSymbolHandlerAssigned	BOOL	TRUE if the symbol handler has been assigned.
bVariableFound	BOOL	TRUE if a variable was found in the stream.
sSymbolPath	STRING(255)	Theam/Stream.Symbol
tCycleTime	LTIME	Cycle time of the publishing system.
nMaxNumElements	UDINT	Maximum number of saved symbols influenced by StreamHelper.

Methods

Name	Definition location	Description
GetValue	Local	Get the value of the specified element.
GetOversamplingValues	Local	Get the oversampling values of the specified element.
GetArrayValues	Local	Get the array values of the specified element.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

7.1.1.3.1.5.13 FB_ALYC_IotSymbol_UINT

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALYC_IotSymbol_UINT
VAR_INPUT
    stConfig : ST_ALYC_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALYC_IotSymbol_Config	Structure for the configuration of the FB.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	EventLogger
bError	BOOL	TRUE if an error has occurred.
bNewResult	BOOL	TRUE if a new result has been calculated.
bConfigured	BOOL	TRUE if the FB is successfully configured.
bSymbolHandlerAssigned	BOOL	TRUE if the symbol handler has been assigned.
bVariableFound	BOOL	TRUE if a variable was found in the stream.
sSymbolPath	STRING(255)	Theam/Stream.Symbol
tCycleTime	LTIME	Cycle time of the publishing system.
nMaxNumElements	UDINT	Maximum number of saved symbols influenced by StreamHelper.

Methods

Name	Definition location	Description
GetValue	Local	Get the value of the specified element.
GetOversamplingValues	Local	Get the oversampling values of the specified element.
GetArrayValues	Local	Get the array values of the specified element.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

7.1.1.3.1.5.14 FB_ALYC_IotSymbol_ULINT

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALYC_IotSymbol_ULINT
VAR_INPUT
    stConfig : ST_ALYC_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALYC_IotSymbol_Config	Structure for the configuration of the FB.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	EventLogger
bError	BOOL	TRUE if an error has occurred.
bNewResult	BOOL	TRUE if a new result has been calculated.
bConfigured	BOOL	TRUE if the FB is successfully configured.
bSymbolHandlerAssigned	BOOL	TRUE if the symbol handler has been assigned.
bVariableFound	BOOL	TRUE if a variable was found in the stream.
sSymbolPath	STRING(255)	Theam/Stream.Symbol
tCycleTime	LTIME	Cycle time of the publishing system.
nMaxNumElements	UDINT	Maximum number of saved symbols influenced by StreamHelper.

Methods

Name	Definition location	Description
GetValue	Local	Get the value of the specified element.
GetOversamplingValues	Local	Get the oversampling values of the specified element.
GetArrayValues	Local	Get the array values of the specified element.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

7.1.1.3.1.5.15 FB_ALYC_IotSymbol_USINT

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALYC_IotSymbol_USINT
VAR_INPUT
    stConfig : ST_ALYC_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALYC_IotSymbol_Config	Structure for the configuration of the FB.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	EventLogger
bError	BOOL	TRUE if an error has occurred.
bNewResult	BOOL	TRUE if a new result has been calculated.
bConfigured	BOOL	TRUE if the FB is successfully configured.
bSymbolHandlerAssigned	BOOL	TRUE if the symbol handler has been assigned.
bVariableFound	BOOL	TRUE if a variable was found in the stream.
sSymbolPath	STRING(255)	Theam/Stream.Symbol
tCycleTime	LTIME	Cycle time of the publishing system.
nMaxNumElements	UDINT	Maximum number of saved symbols influenced by StreamHelper.

Methods

Name	Definition location	Description
GetValue	Local	Get the value of the specified element.
GetOversamplingValues	Local	Get the oversampling values of the specified element.
GetArrayValues	Local	Get the array values of the specified element.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

7.1.1.3.1.5.16 FB_ALYC_IotSymbol_WORD

Syntax

Definition:

```
FUNCTION_BLOCK FB_ALYC_IotSymbol_WORD
VAR_INPUT
    stConfig : ST_ALYC_IotSymbol_Config;
END_VAR
VAR_OUTPUT
    ipResultMessage: I_TcMessage;
    bError: BOOL;
    bNewResult: BOOL;
    bConfigured: BOOL;
    bSymbolHandlerAssigned: BOOL;
    bVariableFound: BOOL;
    sSymbolPath: STRING(255);
    tCycleTime: LTIME;
    nMaxNumElements: UDINT;
    nNumElements: UDINT;
END_VAR
```

Inputs

Name	Type	Description
stConfig	ST_ALYC_IotSymbol_Config	Structure for the configuration of the FB.

Outputs

Name	Type	Description
ipResultMessage	I_TcMessage	EventLogger
bError	BOOL	TRUE if an error has occurred.
bNewResult	BOOL	TRUE if a new result has been calculated.
bConfigured	BOOL	TRUE if the FB is successfully configured.
bSymbolHandlerAssigned	BOOL	TRUE if the symbol handler has been assigned.
bVariableFound	BOOL	TRUE if a variable was found in the stream.
sSymbolPath	STRING(255)	Theam/Stream.Symbol
tCycleTime	LTIME	Cycle time of the publishing system.
nMaxNumElements	UDINT	Maximum number of saved symbols influenced by StreamHelper.

Methods

Name	Definition location	Description
GetValue	Local	Get the value of the specified element.
GetOversamplingValues	Local	Get the oversampling values of the specified element.
GetArrayValues	Local	Get the array values of the specified element.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics

7.1.1.3.2 Data types

7.1.1.3.2.1 ST_ALYC_MqttStreamConfig

```
TYPE ST_ALYC_MqttStreamConfig :
STRUCT
     bAutoStartStream : BOOL := TRUE;
     nAdsBuffer : DINT := 32; // Samples in buffer
     nSamplingDivider : UDINT := 1;
     eCompressionMethod : ANALYTICS_COMPRESSION := ANALYTICS_COMPRESSION.ANALYTICS_COMPRESSION_RL;
     eCompressionWidth : ANALYTICS_COMPRESSION_WIDTH := ANALYTICS_COMPRESSION_WIDTH.ANALYTICS_COMP_W
IDTH_8;
     eExternalTimeType : ETcExternalTimeType := ETcExternalTimeType.SystemTime;
     stDeviceLocation: ST_ALYC_Address := (sAddress := '', sLongitude := '', sLatitude := '');
     sSystemIdAlias : STRING;
     stConnection : ST_ALYC_MqttConnectionSettings;
     nQueueSize : UDINT := 0;
     sMqttTopic : STRING(255) := ''; // Combination of main topic and stream topic: MainTopic/
StreamTopic
     bQueueWhenDisconnected : BOOL := FALSE;
     bQueueInFile : BOOL := FALSE;
END_STRUCT
END_TYPE
```

7.1.1.3.2.2 ST_ALYC_FileStreamConfig

```
TYPE ST_ALYC_FileStreamConfig :
STRUCT
     bAutoStartStream : BOOL := TRUE;
     nAdsBuffer : DINT := 32; // Samples in buffer
     nSamplingDivider : UDINT :=  1;
     eCompressionMethod :  ANALYTICS_COMPRESSION := ANALYTICS_COMPRESSION.ANALYTICS_COMPRESSION_RL;
     eCompressionWidth : ANALYTICS_COMPRESSION_WIDTH := ANALYTICS_COMPRESSION_WIDTH.ANALYTICS_COMP_W
IDTH_8;
     eExternalTimeType :  ETcExternalTimeType := ETcExternalTimeType.SystemTime;
     stDeviceLocation : ST_ALYC_Address := (sAddress := '', sLongitude := '', sLatitude := '');
     sSystemIdAlias : STRING;
     nMaxFileSize : UDINT := 16;
     nFilesInRingBuffer : UDINT := 2; // Number of files in ring buffer
     sFileDir : STRING(255) := ''; // Optional, Default: %TC_BOOTPRJPATH%
     bEnableRingBuffer : BOOL := TRUE;
     nMaxUnconfWrites: UDINT := 1000; // Number of file writes
without a confirmation from ADS router
END_STRUCT
END_TYPE
```

7.1.1.3.2.3 ST_ALYC_MqttStreamHelperConfig

```
TYPE ST_ALYC_MqttStreamHelperConfig :
STRUCT
     stConnection : ST_ALYC_MqttConnectionSettings;
     sMqttTopic : STRING(255); // Combination of main topic and stream topic
     bAutostartReceive : BOOL := TRUE;
END_STRUCT
END_TYPE
```

7.1.1.3.2.4 ST_ALYC_FileStreamHelperConfig

```
TYPE ST_ALYC_FileStreamHelperConfig :
STRUCT
     nSamplesInReadBuf : UINT := 32; //How many samples to read from file every cycle
     nReadBufsInStreamBuf : UINT := 32; //How many read
     buffers to put into a stream buffer (treated as ring buffer)
     sFileDir : STRING(255) := ''; //Directory in which to find the Analytics data files
     bReadFilesCyclically : BOOL := TRUE; //If all files have been read, start over reading them
from the beginning
END_STRUCT
END_TYPE
```

7.1.1.4 Samples

7.1.1.4.1 Analytics Streams and MQTT Stream Helper.

The sample shows how an MQTT stream and a stream in File mode can be configured. At the same time, a Stream Helper is configured to receive the data from the MQTT stream. The next sample shows how the file stream data can be analyzed. The most important function blocks of the type FB_ALYC_MqttStream [▶ 80], FB_ALYC_FileStream [▶ 81] and FB_ALYC_MqttStreamHelper [▶ 83] are used.

The sample is available for download here:

https://infosys.beckhoff.com/content/1033/tf3500_tc3_analytics_logger/Resources/14831744651.zip

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.47	PC or CX (x64, x86)	Tc3_AnalyticsCommunication

7.1.1.4.2 File Stream Helper

The sample shows how a File-StreamHelper can be used to read Analytics Files and integrate them into the PLC program. The function block FB_ALYC_FileStreamHelper [▶ 84] is used. The Analytics Files from the previous sample can be used here.

The sample is available for download here:

https://infosys.beckhoff.com/content/1033/tf3500_tc3_analytics_logger/Resources/14831745163.zip

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.47	PC or CX (x64, x86)	Tc3_AnalyticsCommunication

7.1.2 Obsolete

7.1.2.1 Using the Programming Interface

As described in the Technical Introduction a stream can be started and stopped from PLC code using Structured Text. For that, the streams of a data logger, themselves being TcCom-objects provide an interface called *ITcAnalyticsStream* comprising two methods, *StartAnalyticsStream()* and *StopAnalyticsStream()*. Follow the following steps to use the interface, the code samples may be useful, too:

Declare a variable of the type *ITcAnalyticsStream* and another one of the type *OTCID* for the object ID of the correspondent stream. For diagnostic purposes, an *HRESULT* variable is advisable.

```
HR : HRESULT := S_OK;
{attribute 'tcinitsymbol'}
oidPlcStream1 : OTCID;
ipPlcStream1 : ITcAnalyticsStream;
```

Next, add the attribute ‘tcinitsymbol’ above the OTCID variable. This way, it doesn’t need to be initialized statically in the source code, instead it be initialized at configuration time by double clicking the PLC instance node in the project tree and selecting the relevant stream in the combo box as illustrated in the following picture.

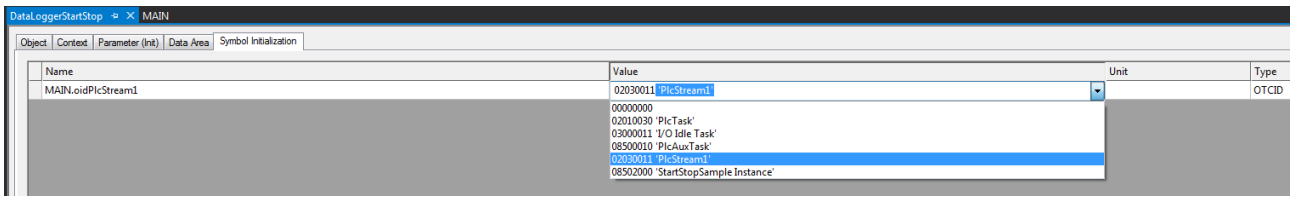


Fig. 1:

After that, get an interface pointer using following the TcCom-object server method and the interface's IID.

```
IF ipPlcStream1 = 0 AND oidPlcStream1 <> 0 THEN
    HR := FW_ObjMgr_GetObjectInstance(oidPlcStream1, IID_ITcAnalyticsStream, ADR(ipPlcStream1));
END_IF
```

Now you can use the interface pointer to call interface's methods. As shown in the following example:

```
IF ipPlcStream1 <> 0 THEN
    IF bStartPlcStream1 THEN
        ipPlcStream1.StartAnalyticsStream();
        bStartPlcStream1 := FALSE;
    END_IF
    IF bStopPlcStream1 THEN
        ipPlcStream1.StopAnalyticsStream();
        bStopPlcStream1 := FALSE;
    END_IF
END_IF
```

The stream is started in the same cycle StartAnalyticsStream() is called and will include the logged variable values. The stream is stopped in the same cycle StopAnalyticsStream() is called but *will not* include the variable values of this cycle.

In the Samples section of this documentation you can find a sample program that includes the here presented code snippets.

7.2 Automation Interface

Please refer to the Automation Interface documentation:
Creating and using the Data Logger and Stream Helper

8 Samples

Data Logger Start/Stop stream from PLC code sample:

https://infosys.beckhoff.com/content/1033/tf3500_tc3_analytics_logger/Resources/6904617099.zip

9 Appendix

9.1 FAQ - frequently asked questions and answers

In this section frequently asked questions are answered in order to make your work with TwinCAT Analytics Logger easier. If you have any further questions, please contact our Support [▶ 108] team.

Should I always use TLS with MQTT? [▶ 106]

Is it possible to have multiple connections? [▶ 106]

Is it possible to control the Analytics Logger by a PLC function block? [▶ 106]

I still have a 4024 system and would like to use the Logger. How do I install it? [▶ 106]

?Should I always use TLS with MQTT?

!Yes, you should if you can. If you can afford the overhead in CPU and bandwidth, then a secure communication channel is invaluable. Depending on the general CPU performance it could be possible to have a noticeable reduction of communication performance.

?Is it possible to have multiple connections?

!Yes, you can connect the Analytics Logger at the same time to different Message Broker just by adding a new instance of the Logger. Also it is possible to have one instance of the Logger for an MQTT communication to a Message Broker and at the same time one instance for writing data into Analytics File to the local system.

Is it possible to control the Analytics Logger by a PLC function block?

!Yes, it is possible to control the Logger with the Analytics Communication Library. Details and examples can be found on the Overview [▶ 80] page in the API chapter.

I still have a 4024 system and would like to use the Logger. How do I install it?

!The drivers and the configuration interface for the Analytics Logger are already available in every 4024 XAE and XAR system. Nothing needs to be installed additionally. Only the configuration needs to be carried out and activated on the target system.



Due to continuous development, the Logger functionalities differ from one TwinCAT Build to the next. The newer a TwinCAT Build is, the more functionalities are available.

9.2 Cybersecurity risk analysis

If you have installed this product, Beckhoff will provide you with the following information for a cybersecurity risk analysis for your system.

Engineering

Workload: TF3500.AnalyticsLogger.XAE

In addition to the standard TwinCAT XAE workload, the following components are registered and started/ loaded with this product:

COM component	Path
TwinCAT.Analytics.Logger.SystemManagerExtension	*C:\...\TwinCAT\Functions\TF3500-Analytics-Logger\SystemManagerExtension*
TwinCAT.Analytics.Logger.AutomationInterface	*C:\...\TwinCAT\Functions\TF3500-Analytics-Logger\SystemManagerExtension*

Runtime

Workload: No separate workload – when a TwinCAT configuration is activated, the necessary runtime components are transferred to the target system via auto-deployment.

Additional information:

Driver	Path
/	

Service	Path	Run as a user
/		

Open ports	Description
/	No server ports – client ports must be open if necessary

10 Support and Service

Beckhoff Support and Service provides assistance with questions regarding Beckhoff products and system solutions, as well as with their planning, commissioning, and operation.

Beckhoff subsidiaries and representative offices

Please contact your Beckhoff subsidiary or your representative office for local support and service for Beckhoff products. You can find the addresses of Beckhoff subsidiaries and representative offices at: www.beckhoff.com/worldwide.

For Beckhoff products and technologies, we offer Training programs, tutorials, and webinars.

Beckhoff Support

Beckhoff Support provides assistance with technical questions regarding the use, planning, programming, and commissioning of Beckhoff products.

☎ Hotline:	+49 5246 963-157
✉ Email:	support@beckhoff.com, Support form
Web page:	www.beckhoff.com/support

Beckhoff Service

Our service specialists are here to assist you with any questions you may have regarding after-sales service processes, such as replacement parts, repairs, etc.

☎ Hotline:	+49 5246 963-460
✉ Email:	service@beckhoff.com, Service form
Web page:	www.beckhoff.com/service

Download finder

The Download finder serves as a central source for downloadable documents and files related to Beckhoff products, such as application reports, technical documentation, drawings, and configuration files.

Beckhoff Information System

The Beckhoff Information System is the central online documentation resource for Beckhoff products and contains documentation on products and automation topics, as well as sample code.

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl, Germany

☎ Phone:	+49 5246 963-0
✉ Email:	info@beckhoff.com
Web page:	www.beckhoff.com

Trademark statements

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.

Third-party trademark statements

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

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

More Information:
www.beckhoff.com/tf3500

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