

BECKHOFF New Automation Technology

Manual | EN

TF3500

TwinCAT 3 | Analytics Logger

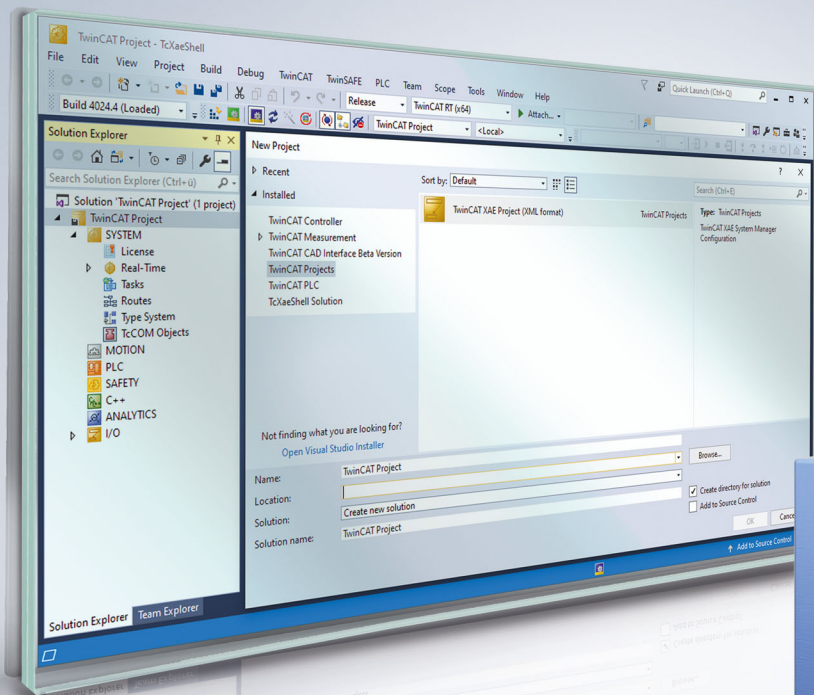


Table of contents

1 Foreword	5
1.1 Notes on the documentation	5
1.2 For your safety	5
1.3 Notes on information security.....	7
2 Overview	8
3 Installation	9
3.1 System requirements	9
3.2 Licensing	9
4 Analytics Workflow - First Steps	12
4.1 Recording data from the machine	12
4.2 Communication	15
4.3 Historicize data.....	16
4.4 Analyse data	23
4.5 24h Analytics application.....	28
5 Technical introduction	37
5.1 Basic Concepts	37
5.2 MQTT basics	38
5.3 Data Compression	44
6 Configuration	45
6.1 Basic settings	45
6.1.1 TLS.....	47
6.1.2 Timestamp correction.....	50
6.1.3 Device-specific information	52
6.2 Data Streams	53
6.2.1 Data Handling	56
7 API	59
7.1 PLC	59
7.1.1 Analytics Communication Library.....	59
7.1.2 Obsolete.....	82
7.2 Automation Interface	83
8 Samples	84
9 Appendix	85
9.1 FAQ - frequently asked questions and answers	85

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.

For installation and commissioning of the components, it is absolutely necessary to observe the documentation and the following notes and explanations.

The qualified personnel is obliged to always use the currently valid documentation.

The responsible staff must ensure that the application or use of the products described satisfies all requirements for safety, including all the relevant laws, regulations, guidelines, and standards.

Disclaimer

The documentation has been prepared 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.

No claims to modify products that have already been supplied may be made on the basis of the data, diagrams, and descriptions in this documentation.

Trademarks

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

If third parties make use of designations or trademarks used in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

Patents

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:

EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
and similar applications and registrations in several other countries.

EtherCAT®

EtherCAT® is 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.

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 well as data visualization with TwinCAT Scope.

Components

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

List of key features

Functionality	TC3 Analytics Logger as MQTT Client	TC3 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 with TwinCAT XAE and XAR. Therefore, the Logger should always be available but in order to use it, one "TC3 Analytics Logger" license per target device is needed which may either be a permanent or a 7-day trial license.

For general information about licensing, please refer to the licensing paragraph below.

3.1 System requirements

Technical Data	TF3500 TC3 Analytics Logger
Operating System	Windows 7, Windows 8, Windows 10, WinCE, TwinCAT/BSD PC (x86, x64 and ARM)
Min. TwinCAT Version	3.1.4022.31
Min. TwinCAT Level	TC1100 TC3 I/O

3.2 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

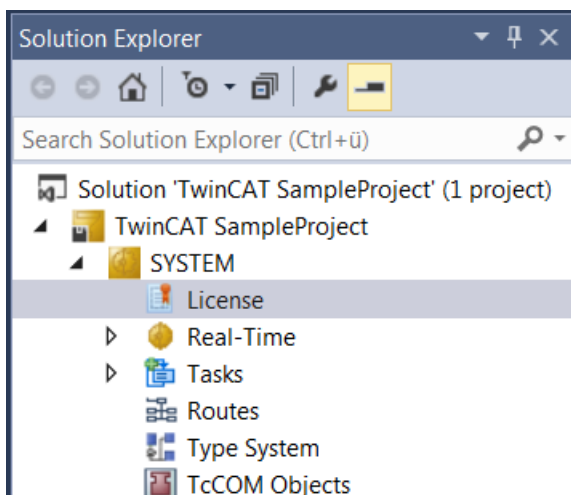
A description of the procedure to license a full version can be found in the Beckhoff Information System in the documentation "[TwinCAT 3 Licensing](#)".

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.

- 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	<input type="checkbox"/> cpu license
TF3650	TC3 Power Monitoring	<input type="checkbox"/> cpu license
TF3680	TC3 Filter	<input type="checkbox"/> cpu license
TF3800	TC3 Machine Learning Inference Engine	<input type="checkbox"/> cpu license
TF3810	TC3 Neural Network Inference Engine	<input type="checkbox"/> cpu license
TF3900	TC3 Solar-Position-Algorithm	<input type="checkbox"/> cpu license
TF4100	TC3 Controller Toolbox	<input checked="" type="checkbox"/> cpu license
TF4110	TC3 Temperature-Controller	<input type="checkbox"/> cpu license
TF4500	TC3 Speech	<input type="checkbox"/> cpu license

- Open the **Order Information (Runtime)** tab.
 - ⇒ In the tabular overview of licenses, the previously selected license is displayed with the status "missing".
- 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.

- Enter the code exactly as it is displayed and confirm the entry.
- 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

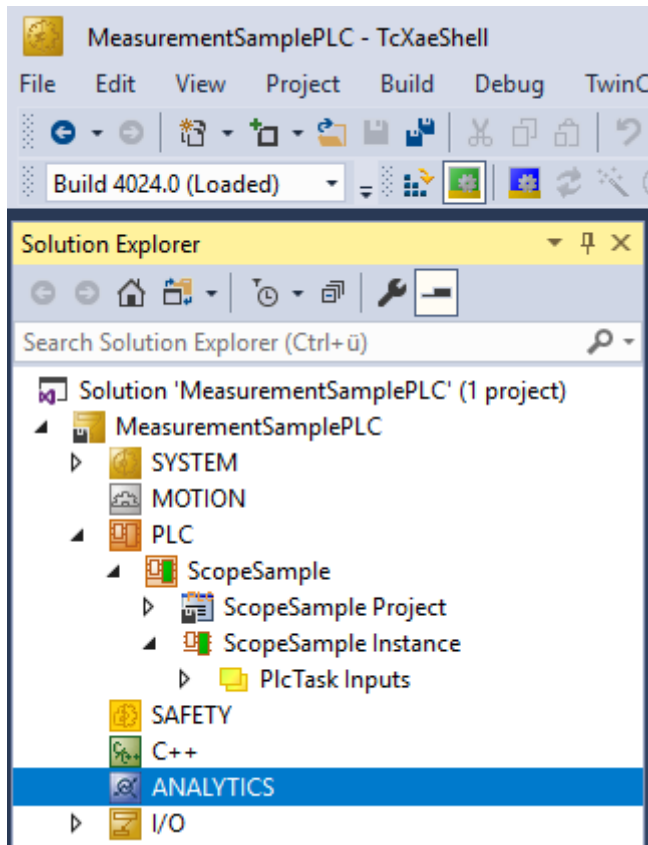
This step by step documentation presents the complete TwinCAT Analytics workflow. From the data acquisition over the communication and historizing up to the evaluation and analysis of the data and to the presentation of the data in web-based dashboard.

4.1 Recording data from the machine

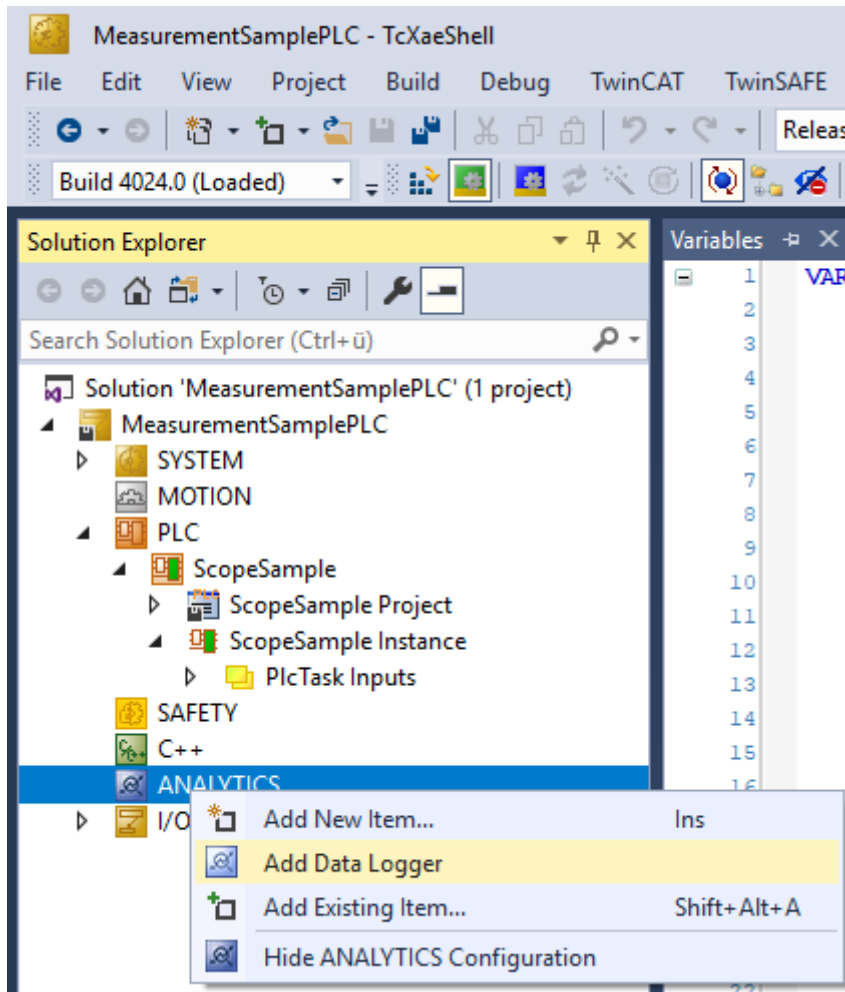
On the machine side is the Analytics Logger the recorder of process data from the machine image, PLC, NC and so on. The Logger is working in the real-time context of TwinCAT.

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

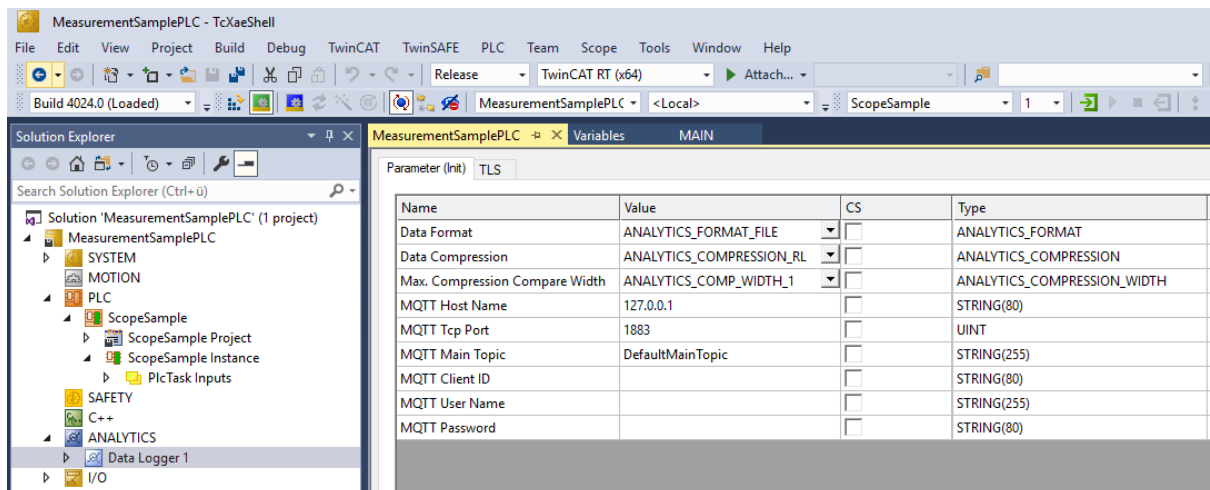
- To configure the Analytics Logger you have to navigate in your existing TwinCAT Project to the Analytics tree node



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



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



You can make your specific Analytics Logger settings

-Data Format: Binary file or MQTT stream

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

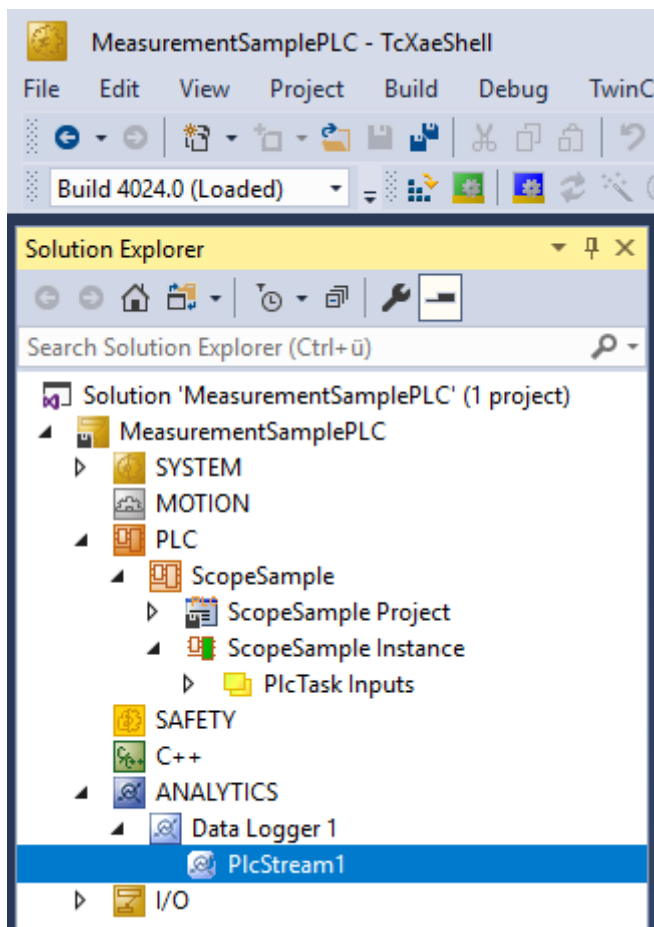
-BINARY: Data will be sent to the configured MQTT Message Broker. You can have multiple Logger in one TwinCAT project to communicate data to different MQTT Message Broker.

-Data Compression: on (default) or off

- Max Compression: mode of the compression
- MQTT host name
- MQTT Tcp port
- MQTT main topic for own hierarchical levels to keep the identification easy
- MQTT Client ID should be unique in the network
- MQTT username
- MQTT password to make authentication at the message broker

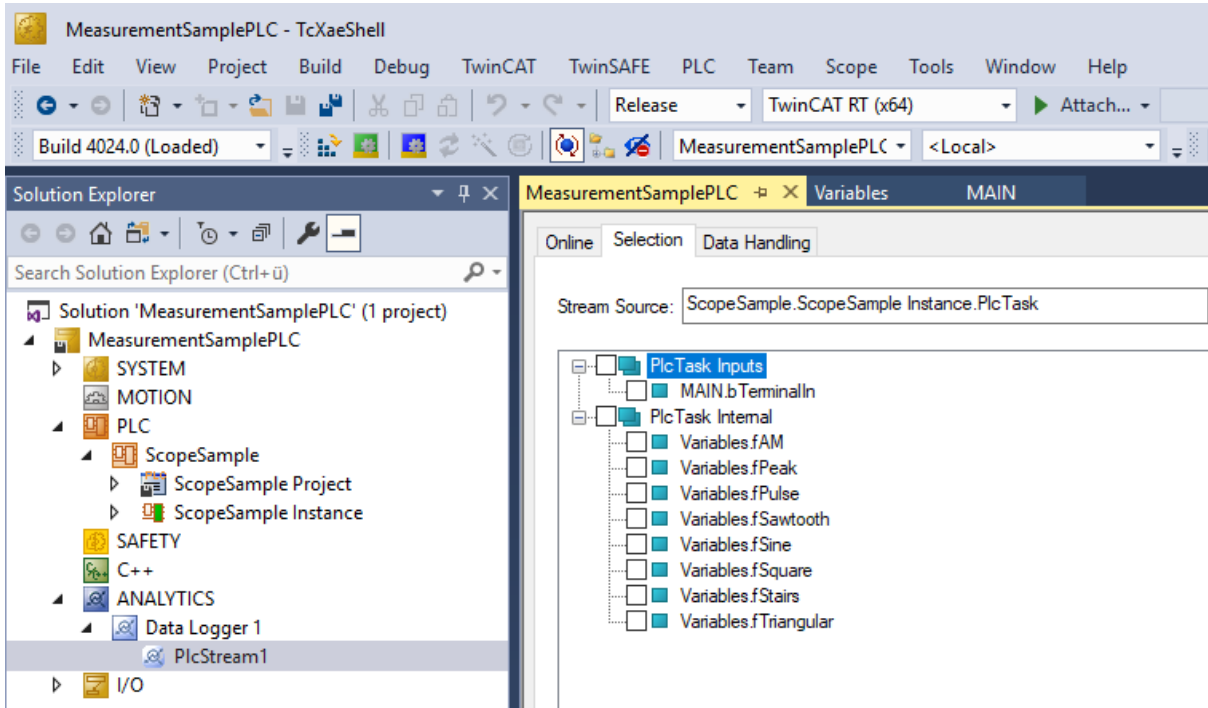
-At the TLS (Transport Layer Security) tab, security settings can be configured. TLS is a secure communication channel between client and server. By the usage of certificates, the TCP port 8883 is exclusively reserved for MQTT over TLS. Analytics Logger is supporting the modes CA Certificates, CA Certificates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

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

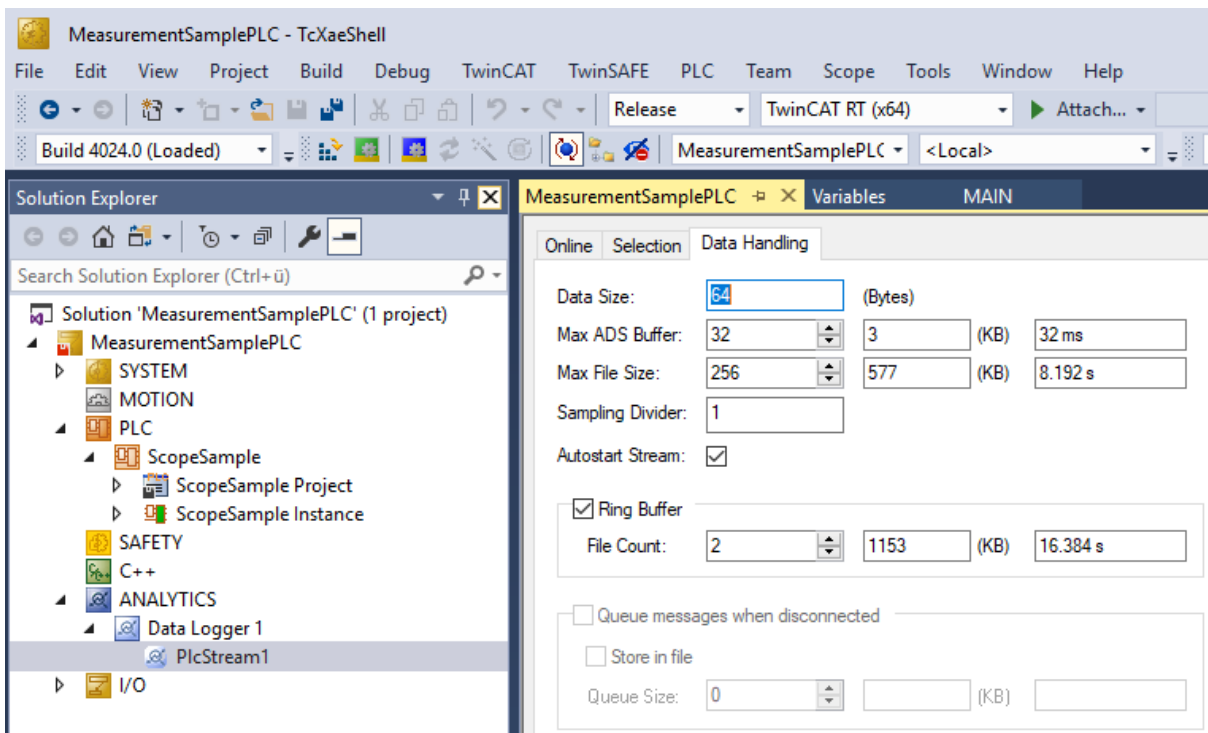


An additional device stream will be shown if your configuration provides an EtherCAT Process Image.

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

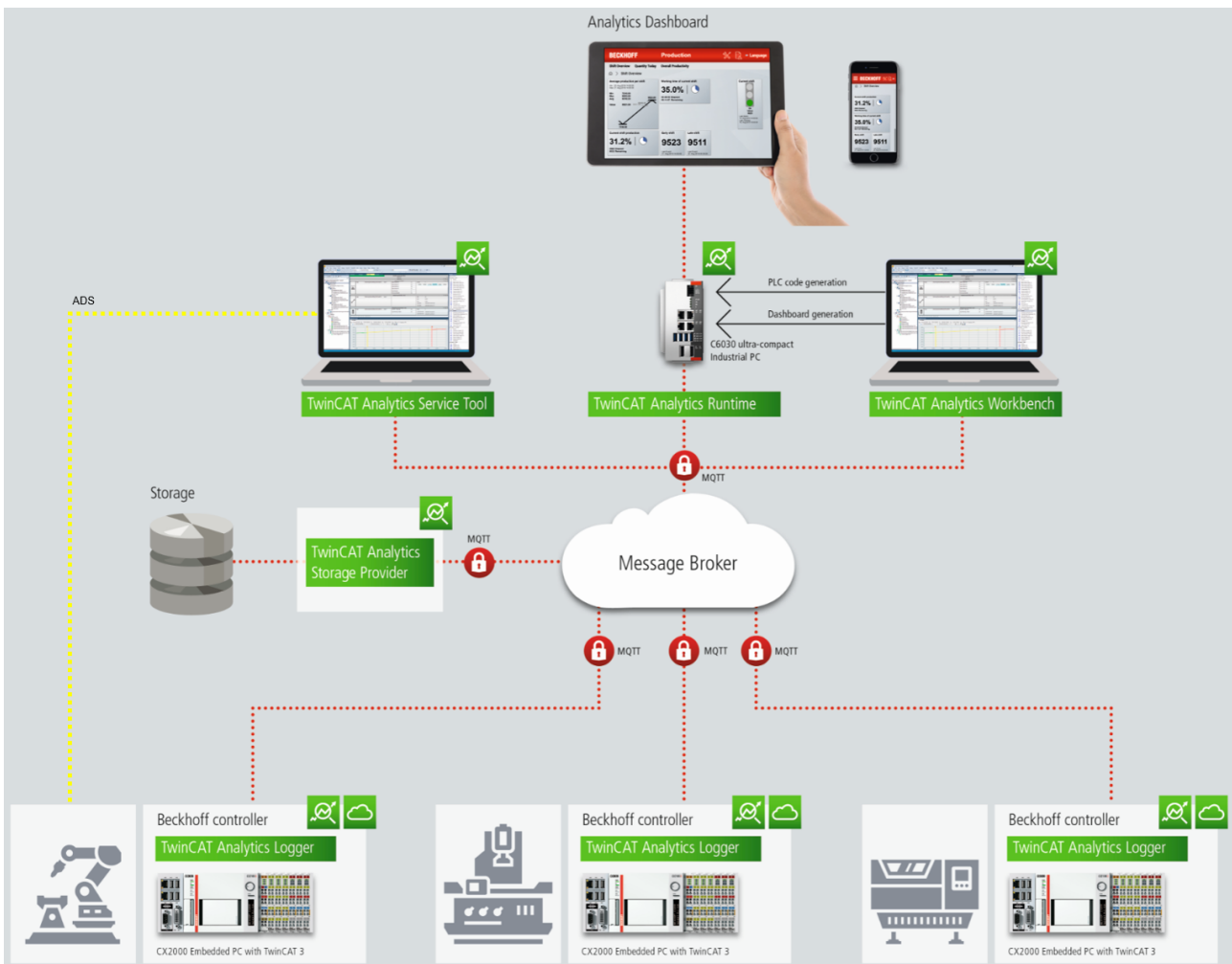


- Finally it is possible to change the package size for the frames or to configure the ring buffer for disconnects and file in the Data Handling tab.



4.2 Communication

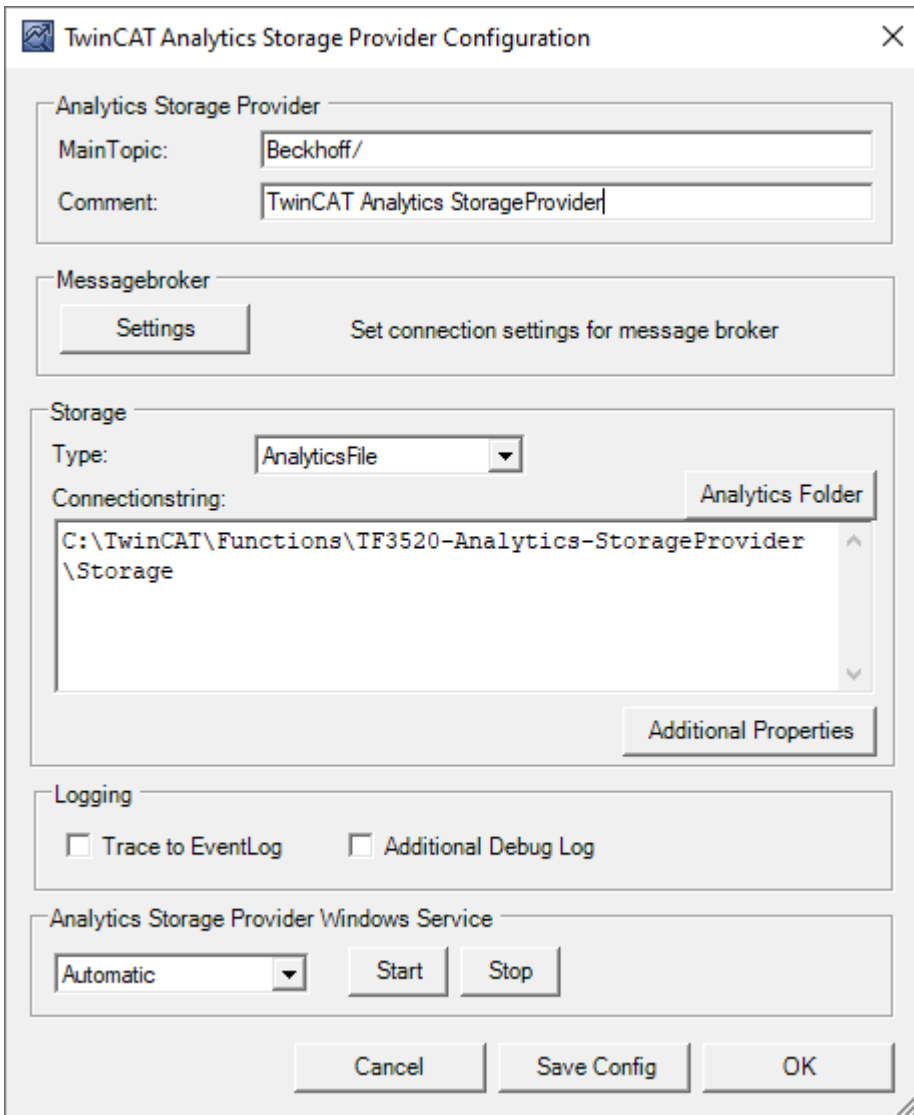
Currently, the Analytics workflow is fully mappable via MQTT. The engineering tools can also access the data of the machines via ADS and carry out analyzes.



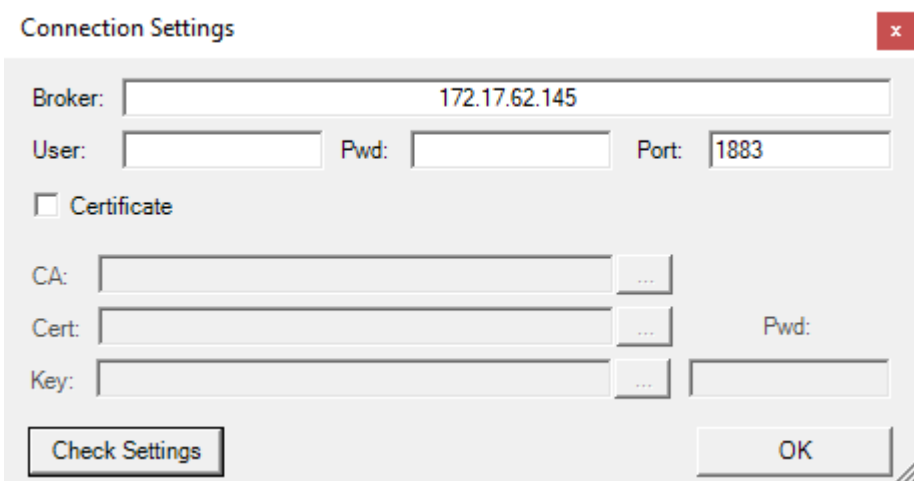
If you choose for the IoT communication protocol MQTT you have to setup a native MQTT Message Broker somewhere in the network (VM in a cloud system is also possible). This Message Broker provides a decoupling of the different applications in the Analytics Workflow.

4.3 Historicize data

After the TwinCAT Analytics Storage Provider has been installed, the service running in the background can be configured. For this purpose you can find the TcAnalyticsStorageProvider_Config application in the C:\TwinCAT\Functions\TF3520-Analytics-StorageProvider\WinService folder.



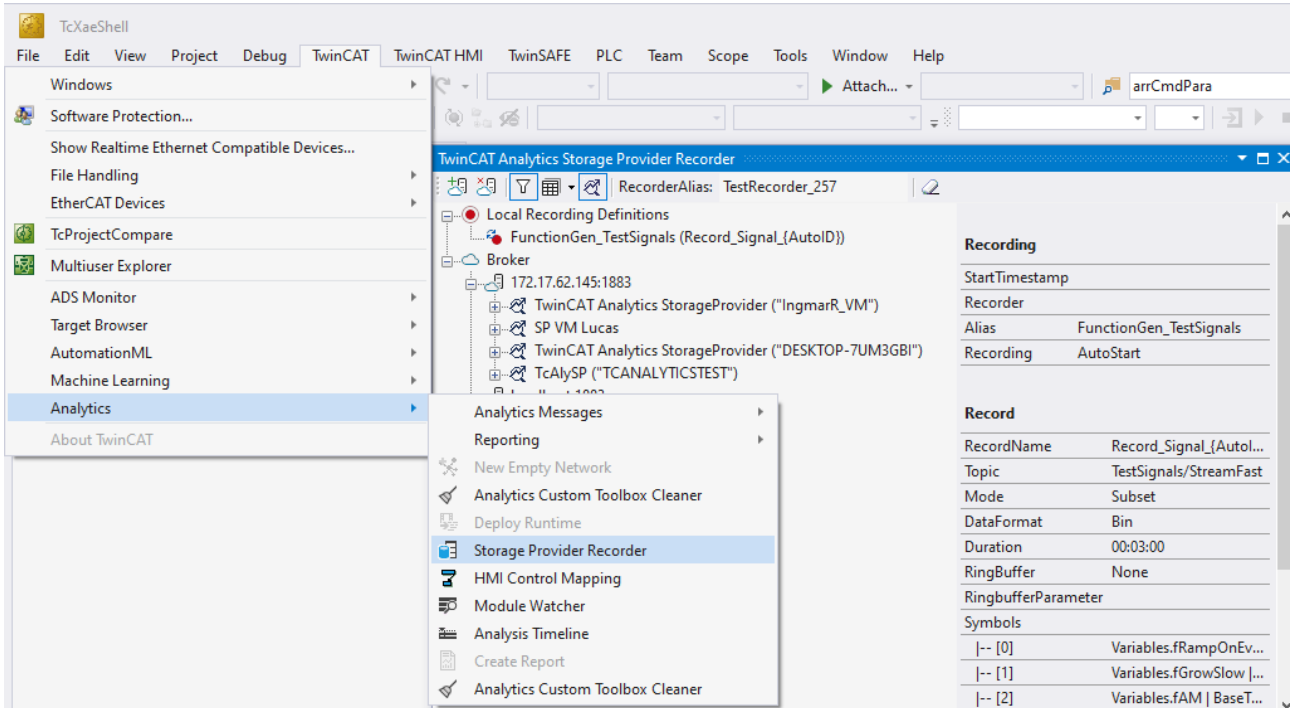
The main part of the topic can be defined in the configuration as well as the comment, which is used for identification if more than one Storage Provider is registered with the message broker.



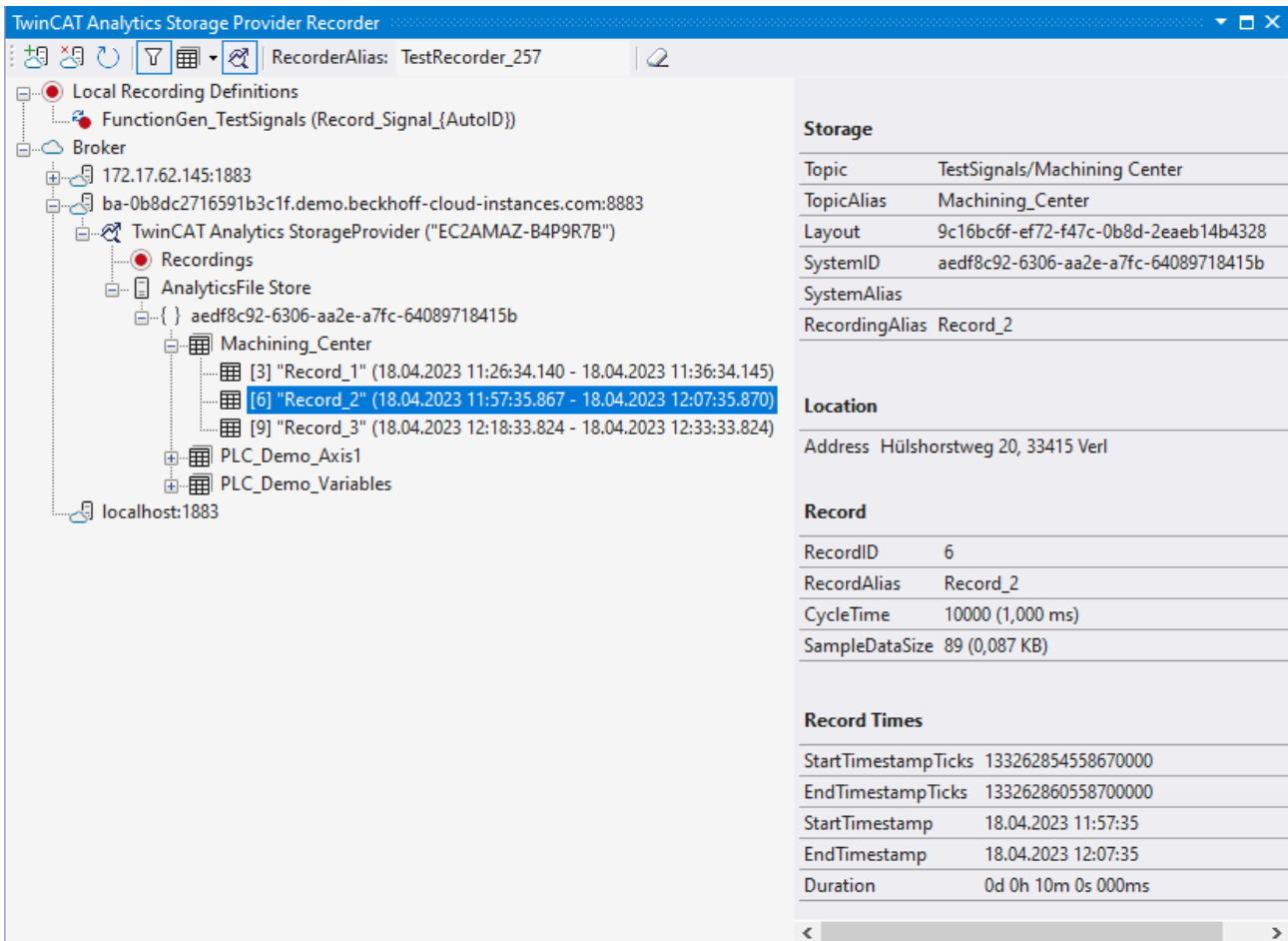
You can make the message broker settings and decide on a storage type:

- Analytics File (binary file)
- Microsoft SQL (binary)
- Microsoft Azure Blob (Azure Cloud required)

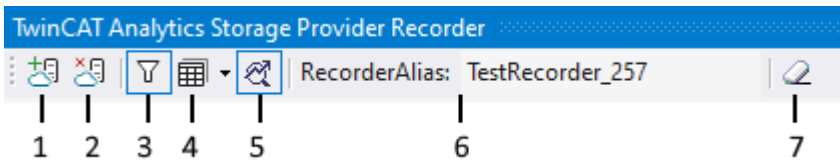
At last you can save the configuration and start the service. The next step is to configure the specific recording. For this you should select the **Storage Provider Recorder** in your development environment.



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



Toolbar

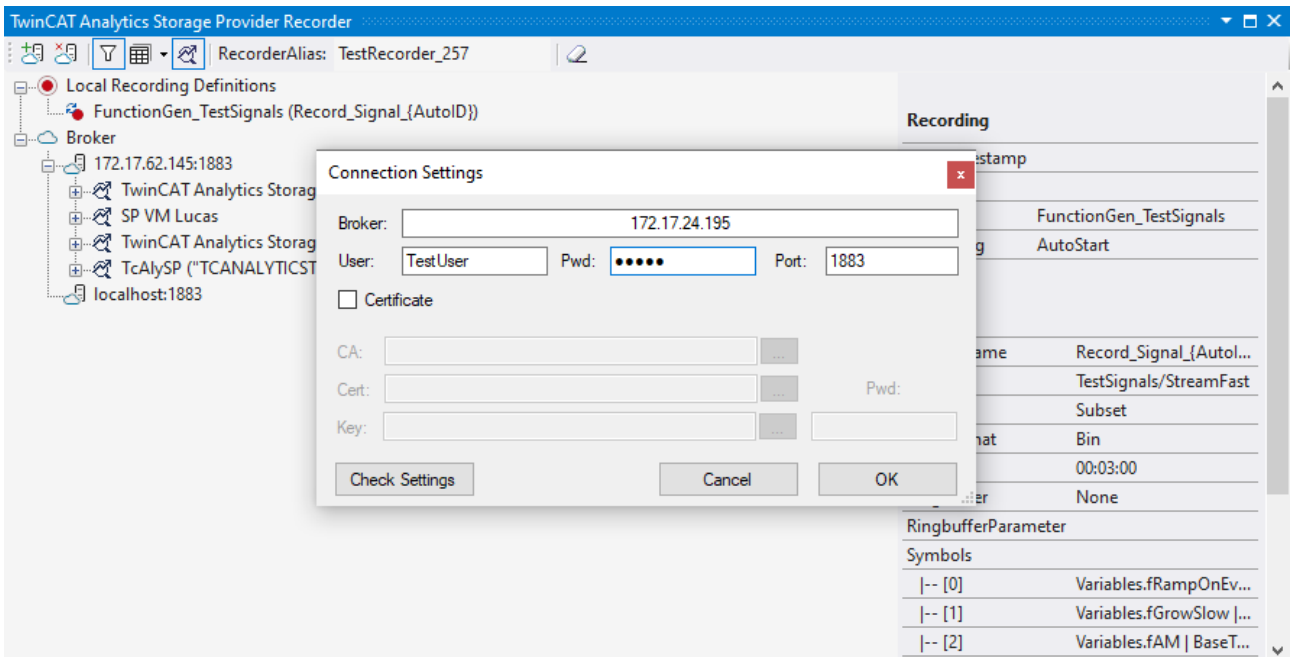


1	Add new broker
2	Remove selected broker
3	Display filters (All / Show my active recordings)
4	Select display type (Alias / Topics)
5	Show / Hide Offline Analytics Storage Provider
6	"RecorderAlias" - grouping name of recordings
7	Remove messages from error list

Recorder window setup

First assign a "RecorderAlias". This helps to group the started recordings and to find its self started ones again. The filter can also be used to display recordings started by others.

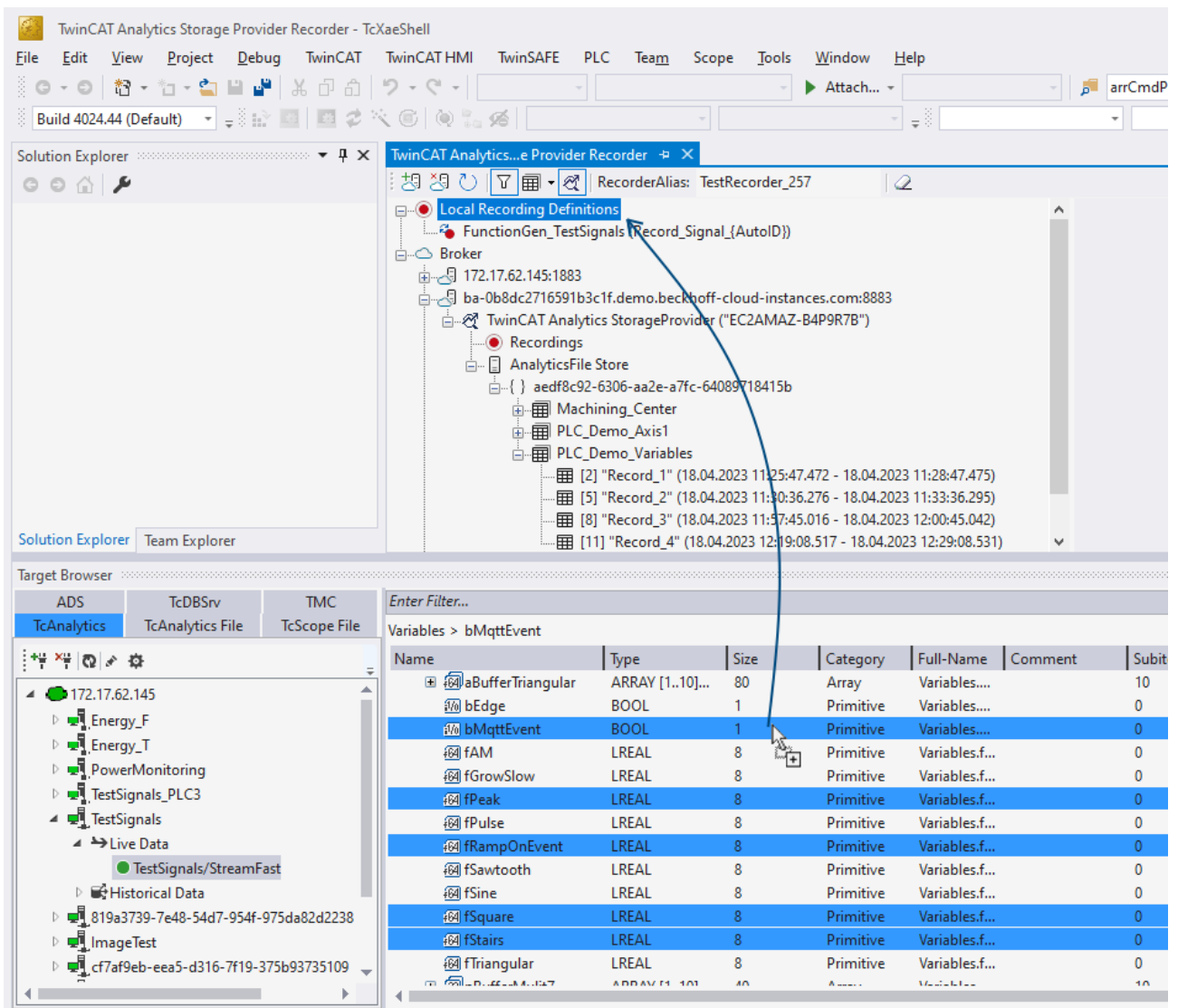
After that, one or more brokers can be set up. This is done via the already known input mask for MQTT connection properties.



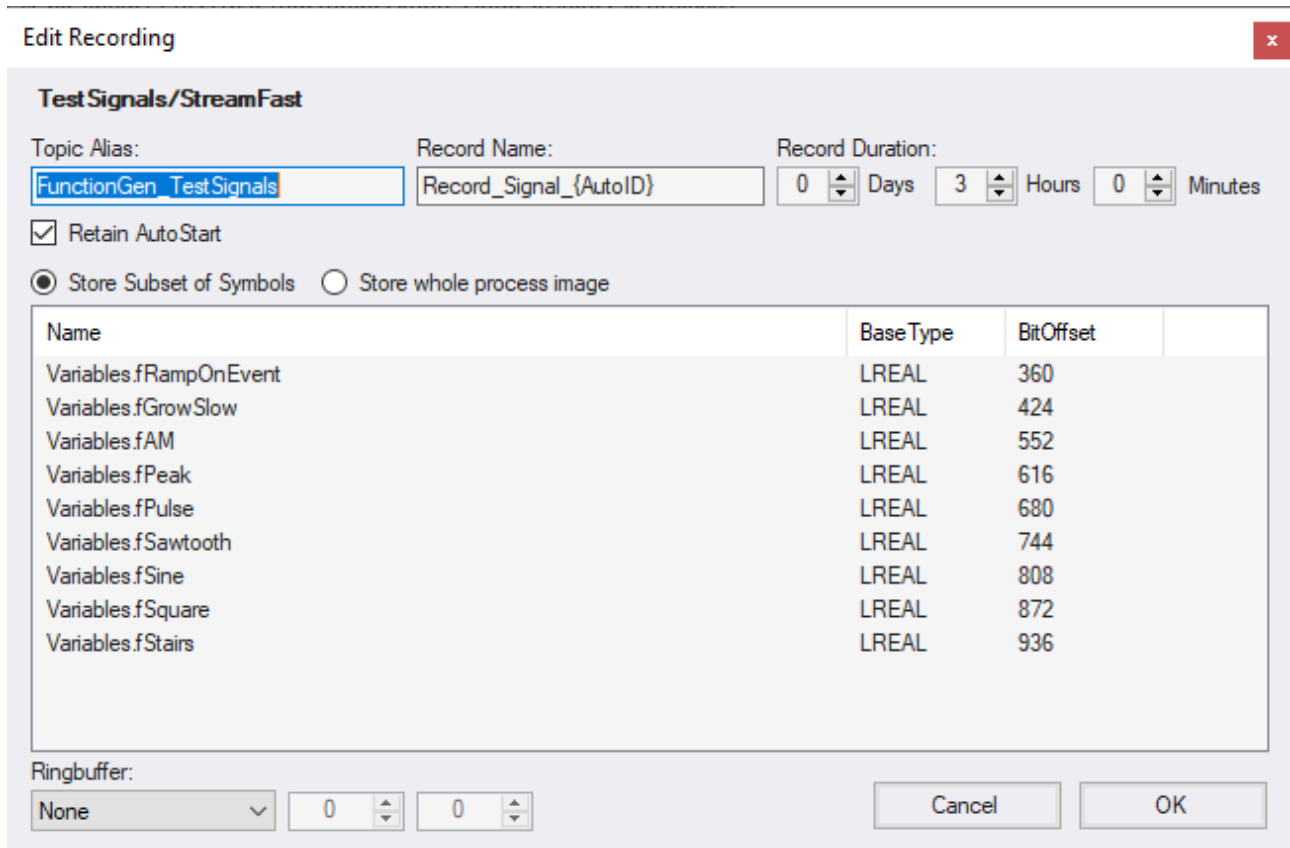
Once a connection to the broker could be established, all Analytics Storage Providers connected to it will be listed.

Create recording definition

To configure the recording, select your target in the Target Browser. Click **Live Data** and select one or more variables by multiple selection and drag and drop them to the **Local Recording Definitions** node in the Recorder window.



In the recorder you can add the selected variables or the complete source process image of the variables.

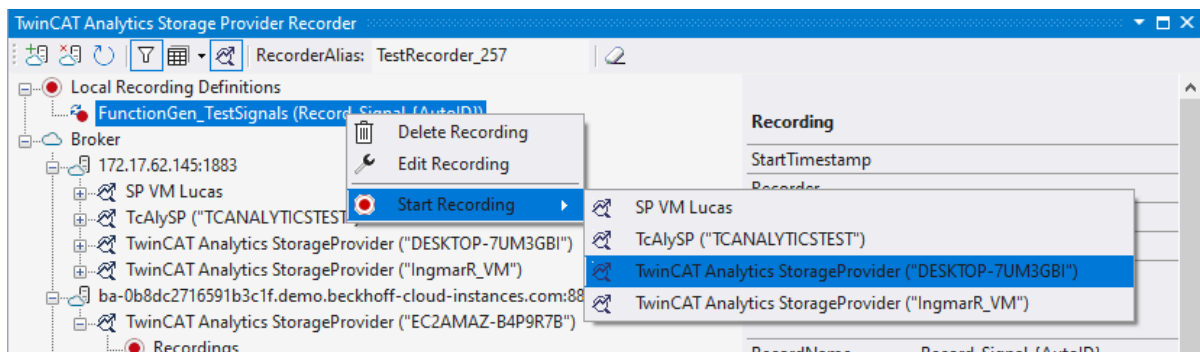


You can also configure recording names and a duration (otherwise the recording will run endlessly until it is stopped manually). A ring buffer can be set by storage or time.

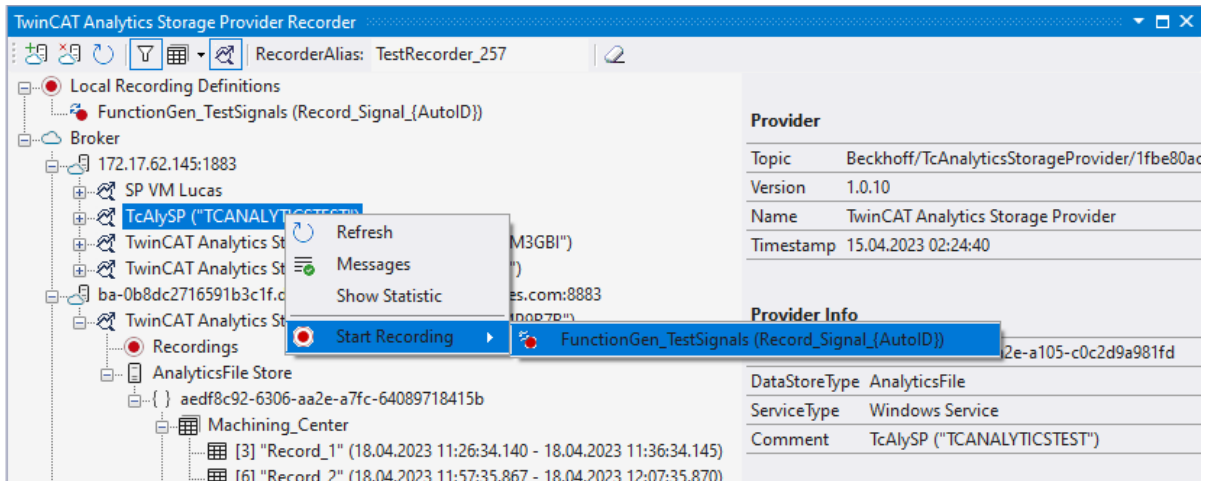
Start recording

There are three different ways to start a recording.

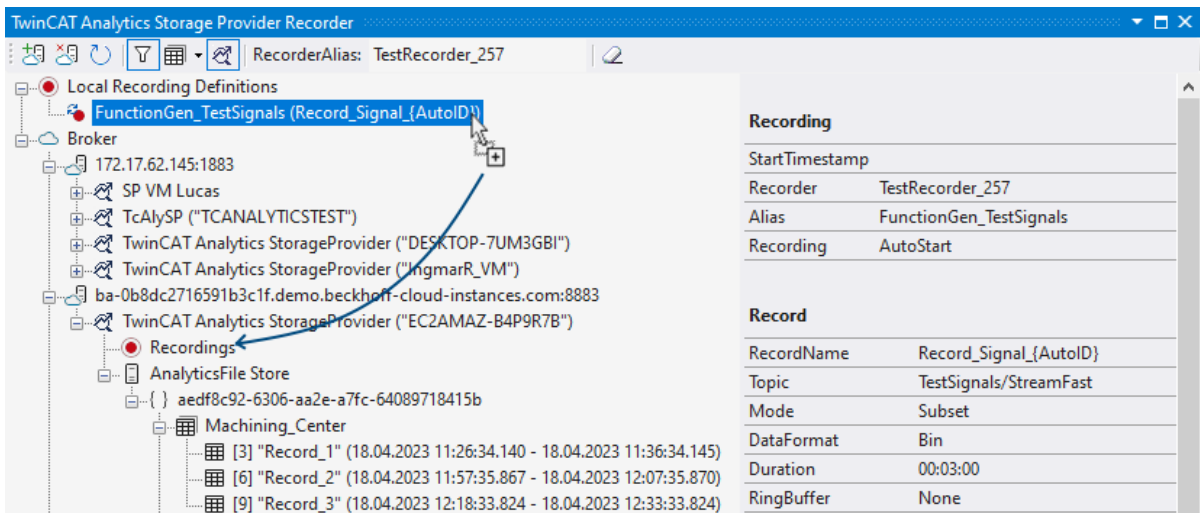
1. Via the context menu at a local recording definition node. Here you can select the desired Analytics Storage Provider, which should record the data. Only the storage providers that have access to the selected topic are displayed.



- From the context menu on the Analytics Storage Provider node. The desired recording definition can be selected here. Only the definitions that can also be processed by the Analytics Storage Provider are displayed.

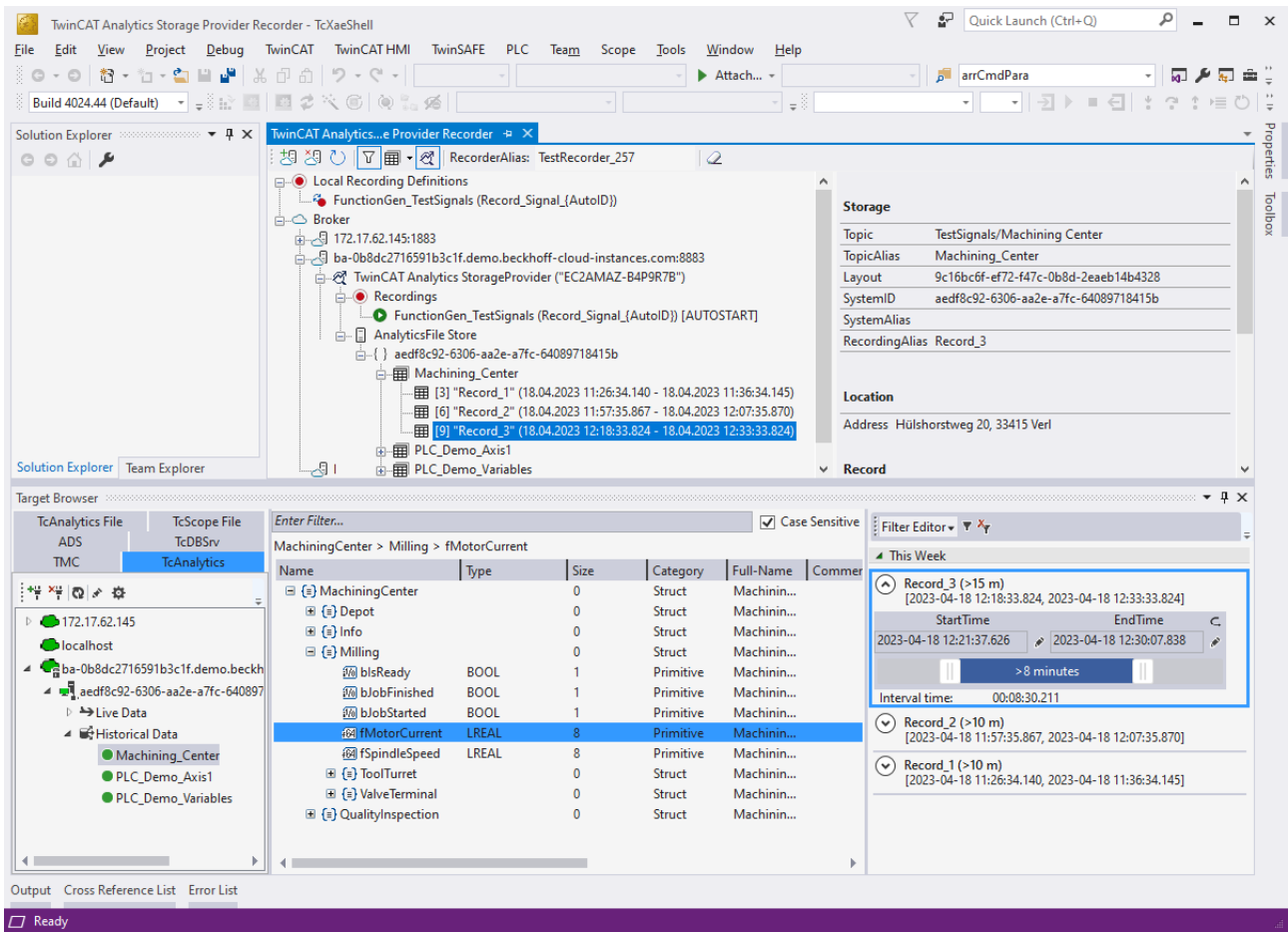


- Drag and drop the local recording definition node. The desired recording definition can be started on the desired Analytics Storage Provider via drag-and-drop. If the Storage Provider is unable to execute the recording definition, an error message is displayed in the error list.



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.

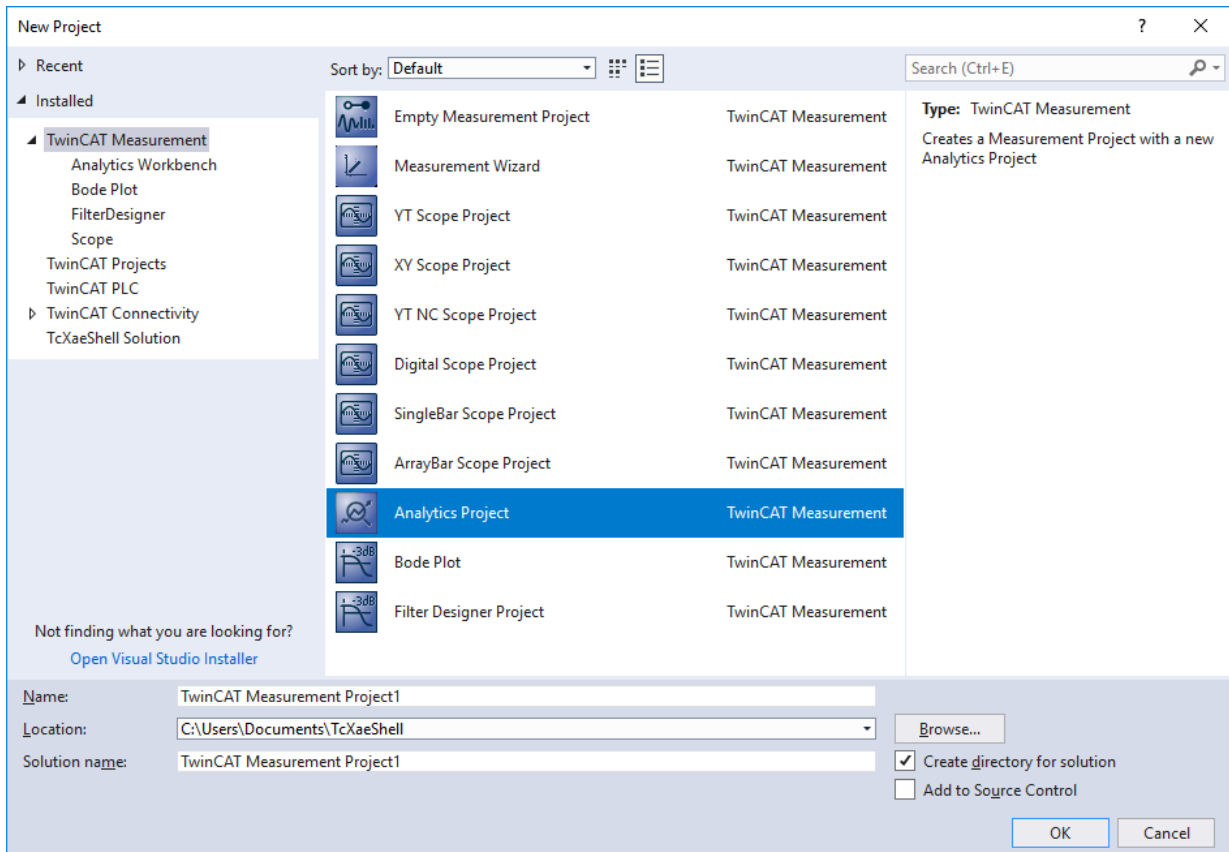


4.4 Analyse data

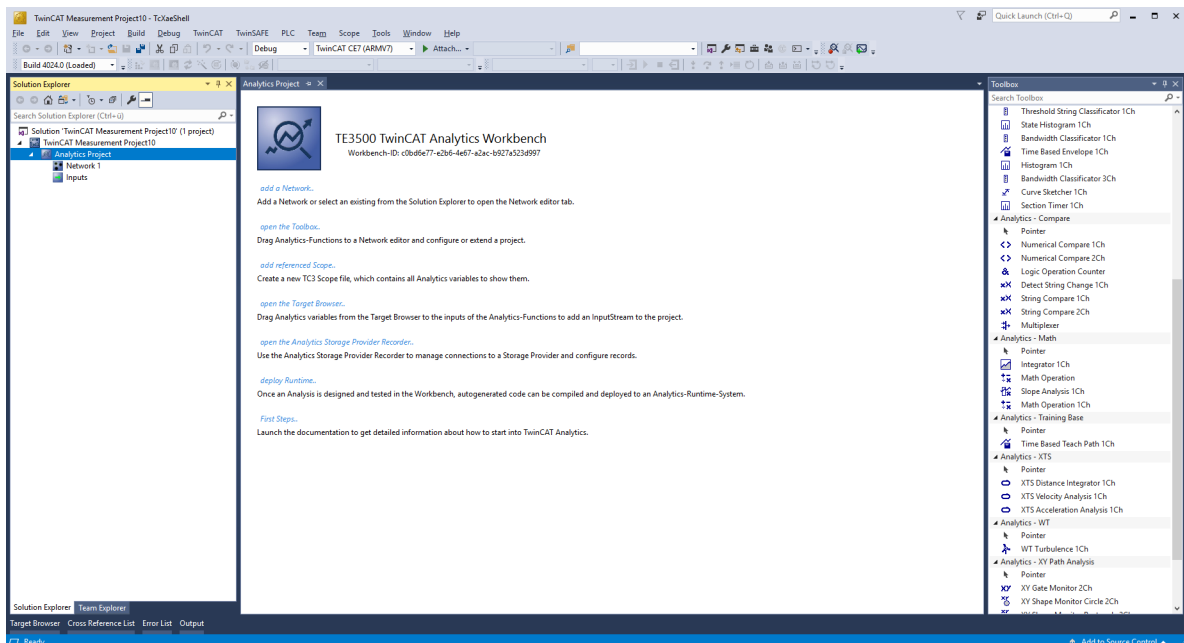
✓ Open your TwinCAT Engineering environment to start the data analysis.

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

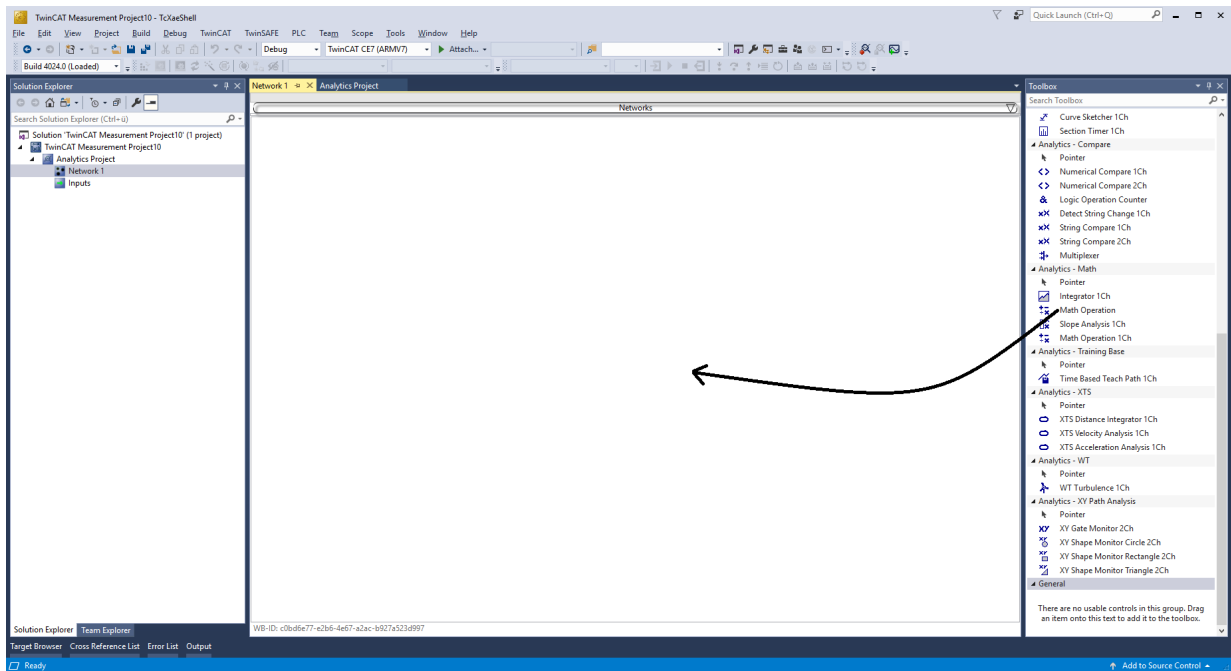
2. Select the **Analytics project template** from **TwinCAT Measurement**.



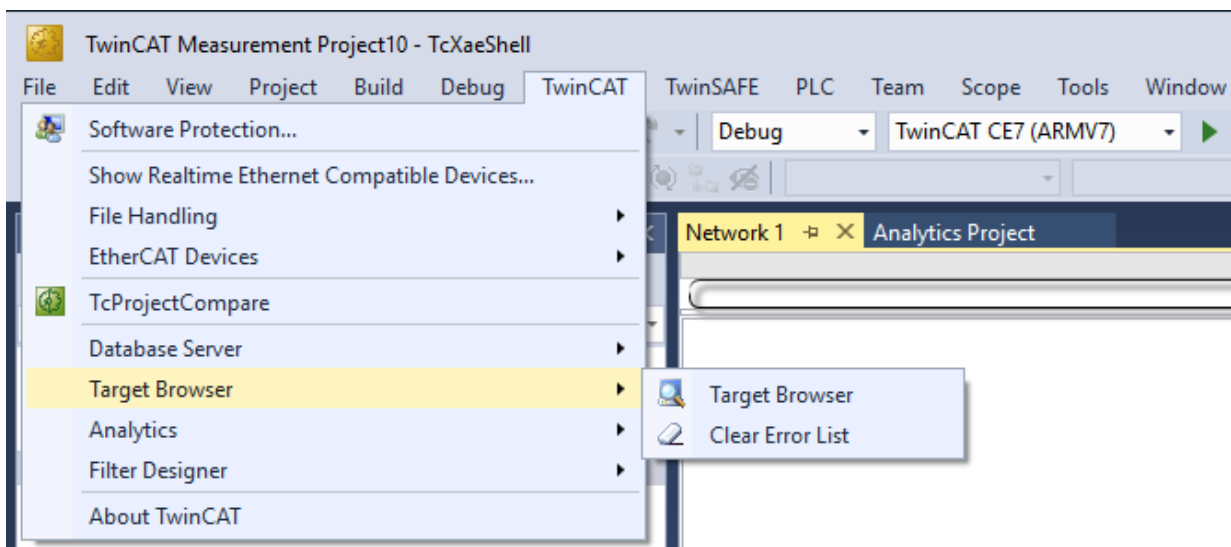
⇒ The new project is displayed in the Solution Explorer. After clicking the **Analytics Project** tree node element a start window opens where you can select your first action. From here you can add a network, open the **Toolbox**, open the **Target Browser** or open the **Analytics Storage Provider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



- 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.

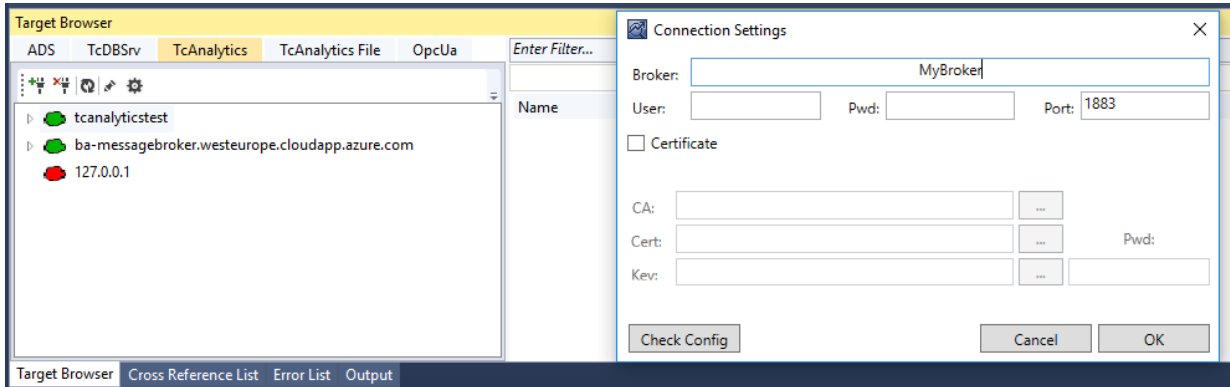


- When you click on the network, an editor opens. Now you can drag and drop the desired algorithm into the editor interface.
- After selecting the algorithm, you need to connect input variables to the modules (algorithm). To do this, open the **Target Browser**.
TwinCAT > Target Browser > Target Browser

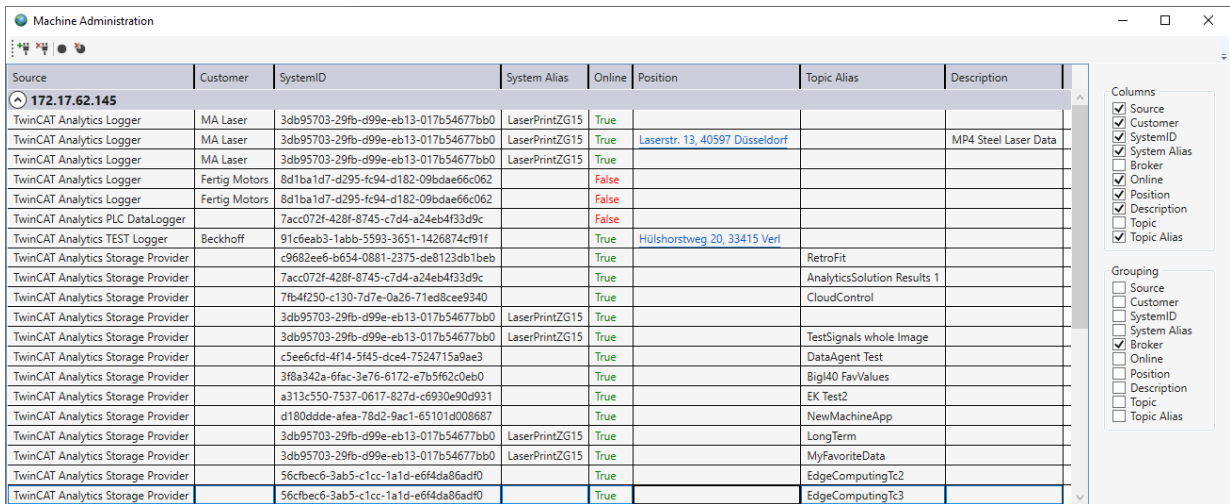


- Now select the **TcAnalytics** or **TcAnalyticsFile** tab in the Target Browser. Continue with the tab **TcAnalytics** (MQTT).

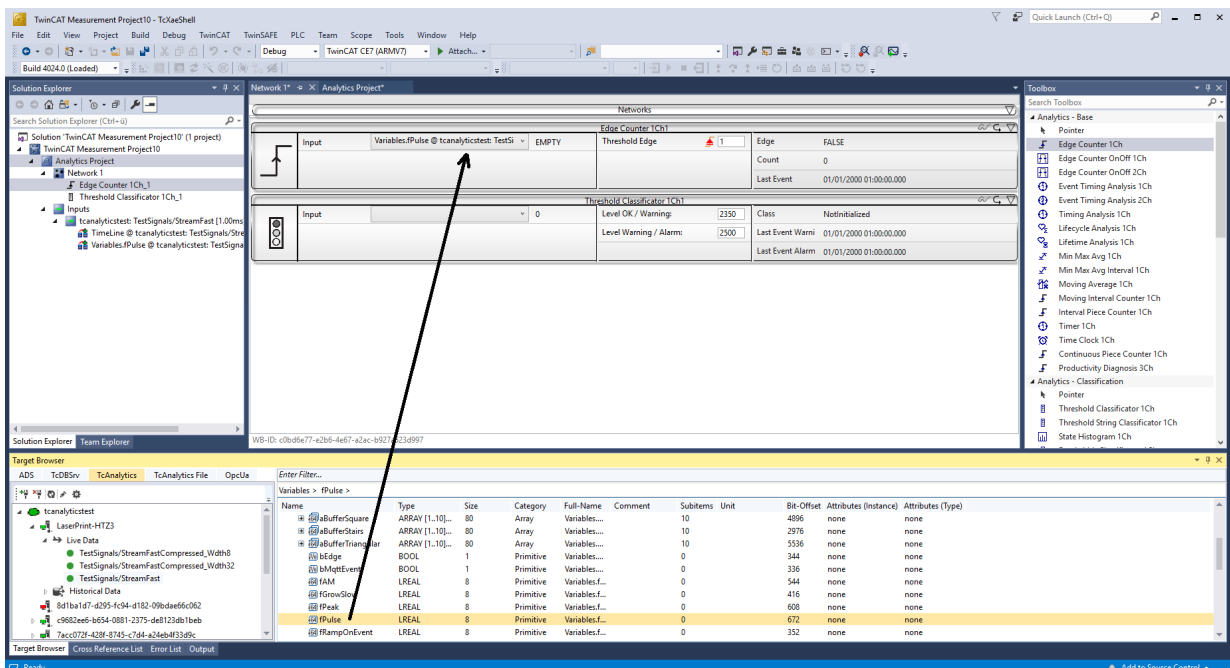
- Click the icon highlighted in green in the toolbar of this Analytics extension. A window opens in which you can specify the connectivity data of your message broker.



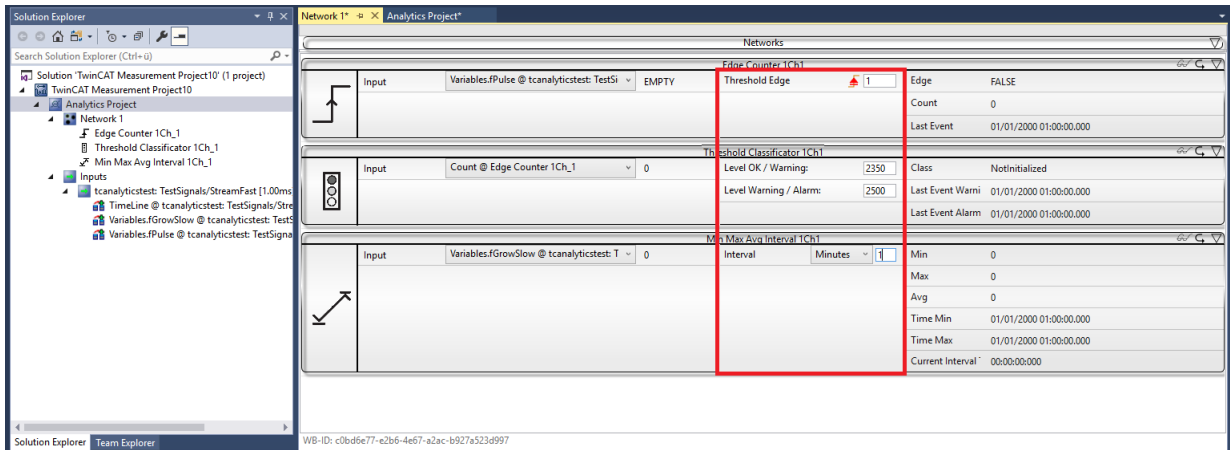
- Select your MQTT Analytics client (TwinCAT Analytics Logger, TwinCAT IoT Data Agent or Beckhoff EK9160). There is a unique ID for each control. This ID is displayed in the Target Browser.
- Clicking on the **gear icon**, you will get to the Machine Administration page. Here you can assign a system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



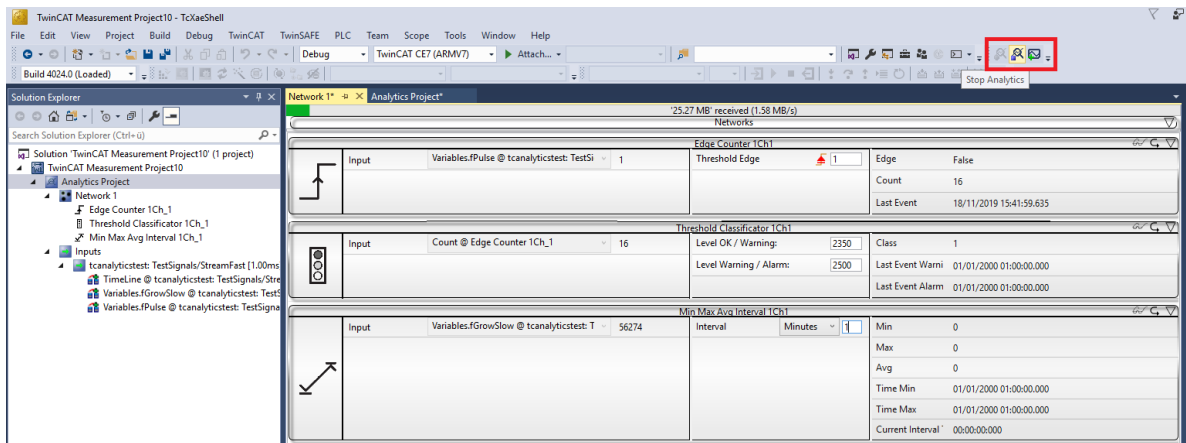
- In the next step, you can choose between live data and historical data for each MQTT Analytics client. In this case, the historical data is provided by the TwinCAT Analytics Storage Provider.



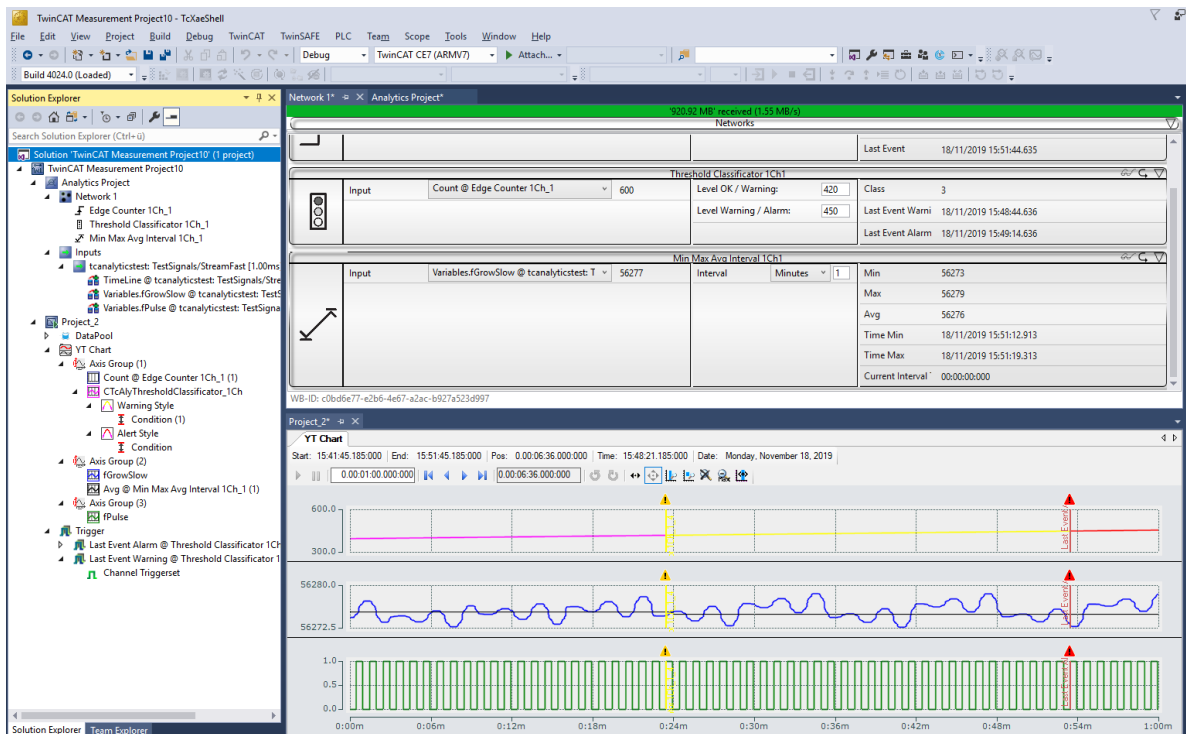
11. You can drag and drop the variables into the inputs of the specific algorithm. In most algorithms, conditions such as thresholds, time intervals, logical operators etc. can be specified. These settings are made in the middle of each module.



⇒ 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.

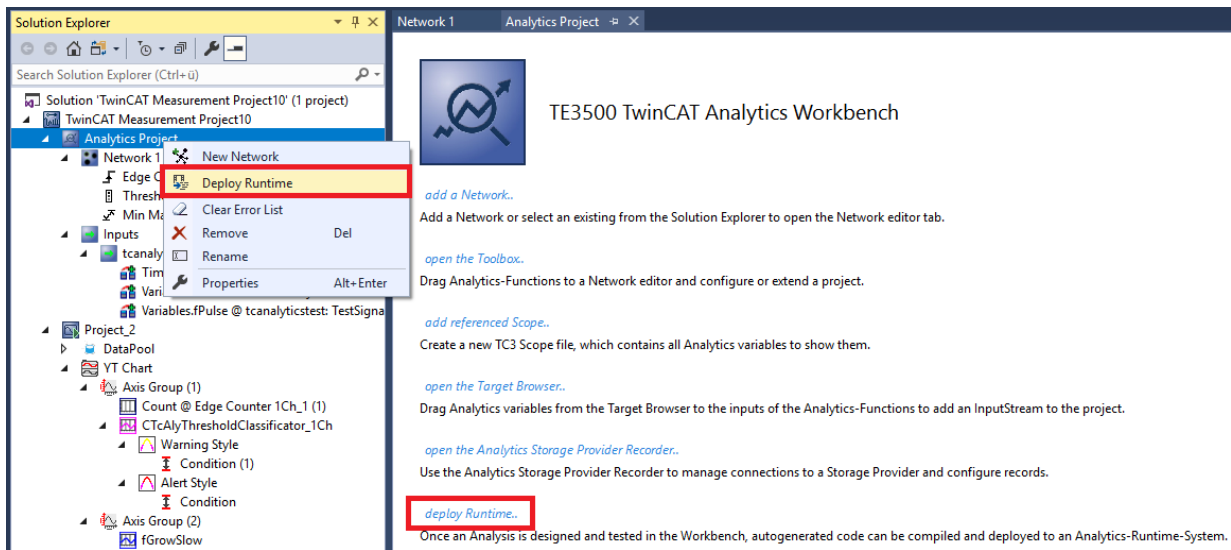


- ⇒ The analysis results can be displayed in the Scope View graphs using drag-and-drop. For example, a mean value can be displayed as a new channel in the view. Timestamps as markers on the X-axes show significant values.

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 (TC3 PLC and HMI Server) and provide the same analysis results as the configurator tool in the engineering environment.

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



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

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

Deploy Analytics Runtime

Codegeneration: Latest Version ("Version 2.1")

Solution | TwinCAT PLC Target | Results | HMI Dashboard | Visual Studio | Summary

Create new Solution

Solution Path: C:\temp\Analytics\Test

Solution Name: Production

Project Name: MachineAnalysis

Add to existing Solution

Solution Path:

Project Name: AnalyticsProject

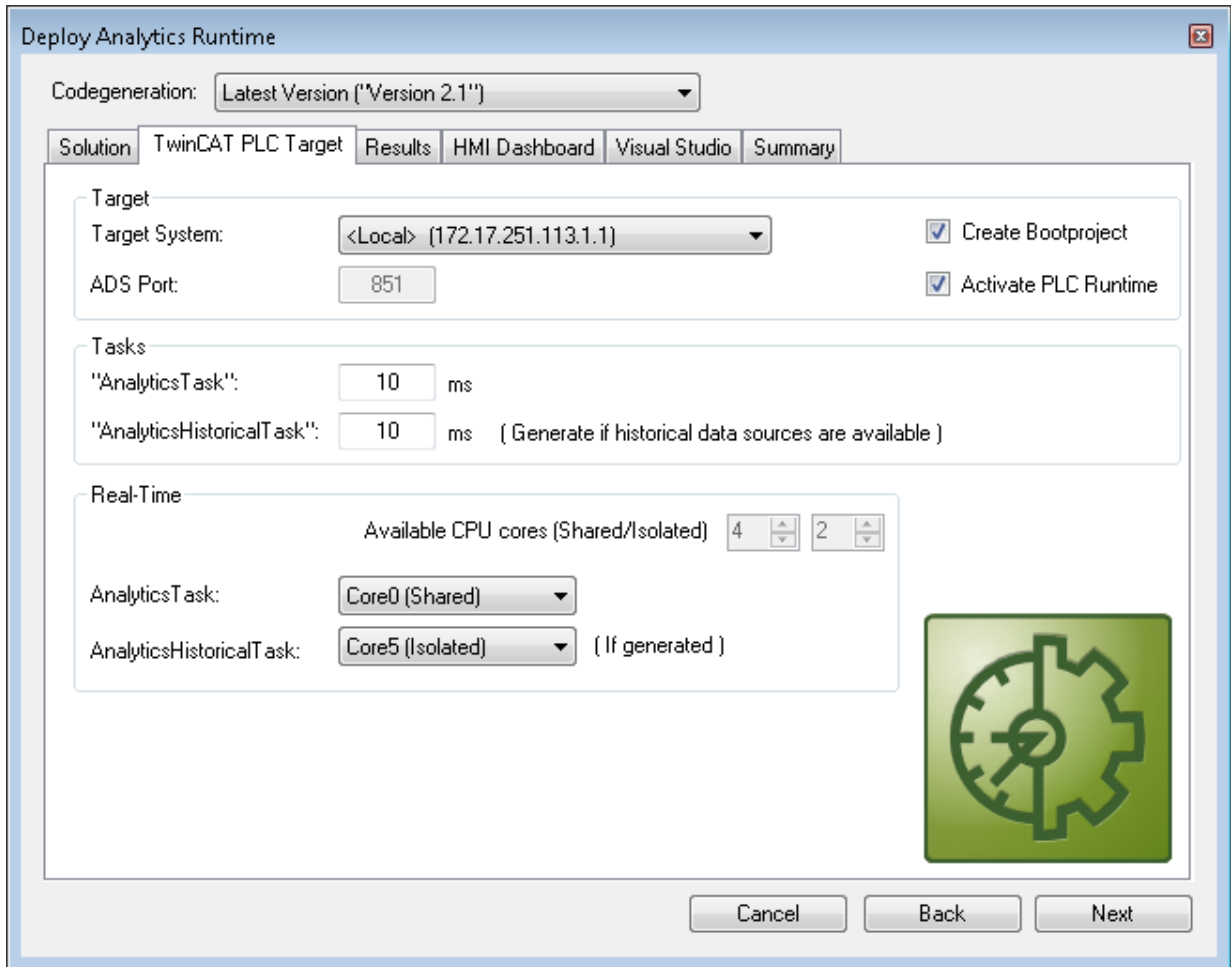
Merge to existing Project (TwinCAT Project Compare)

Solution Path:

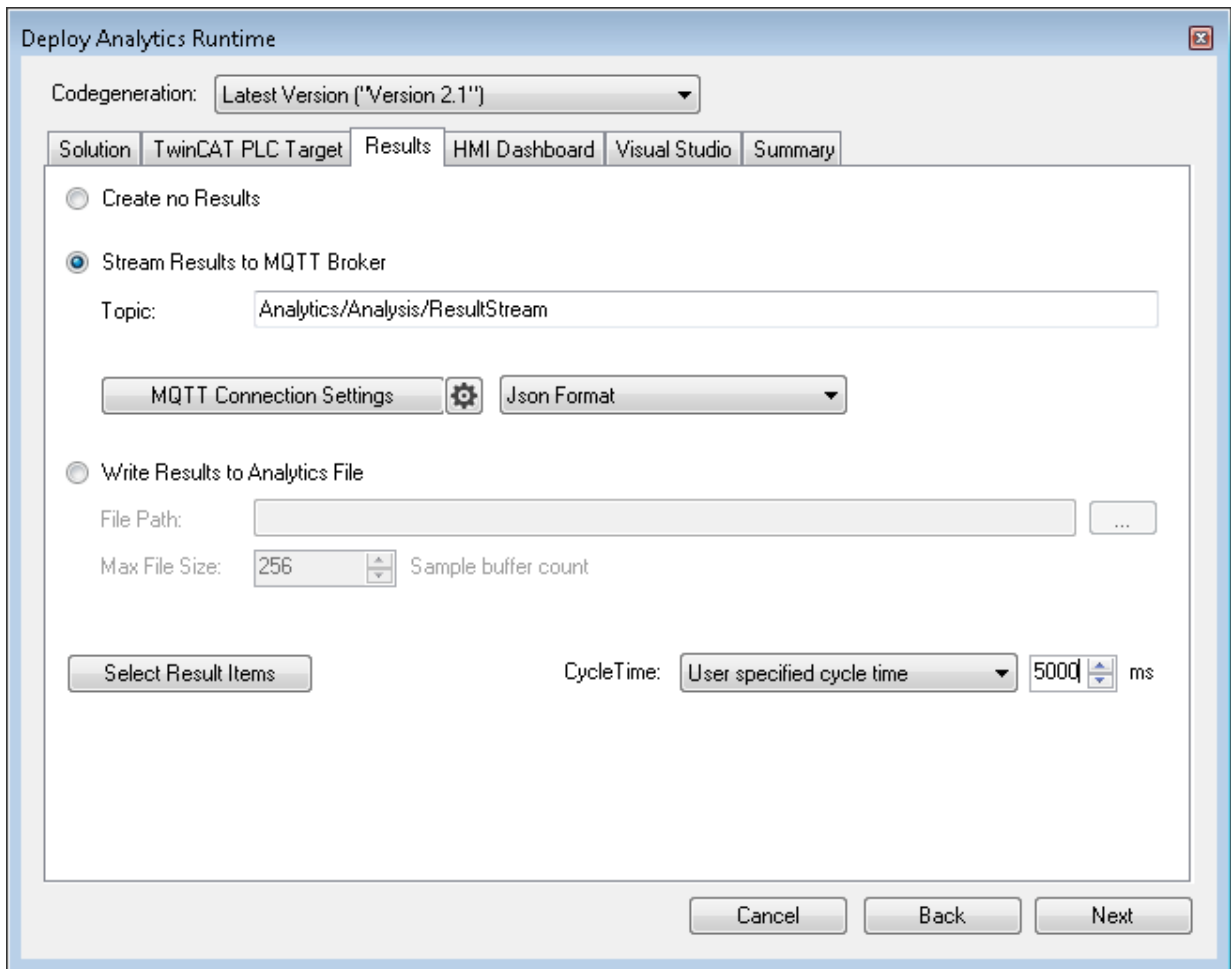
Project Name:

Cancel Back Next

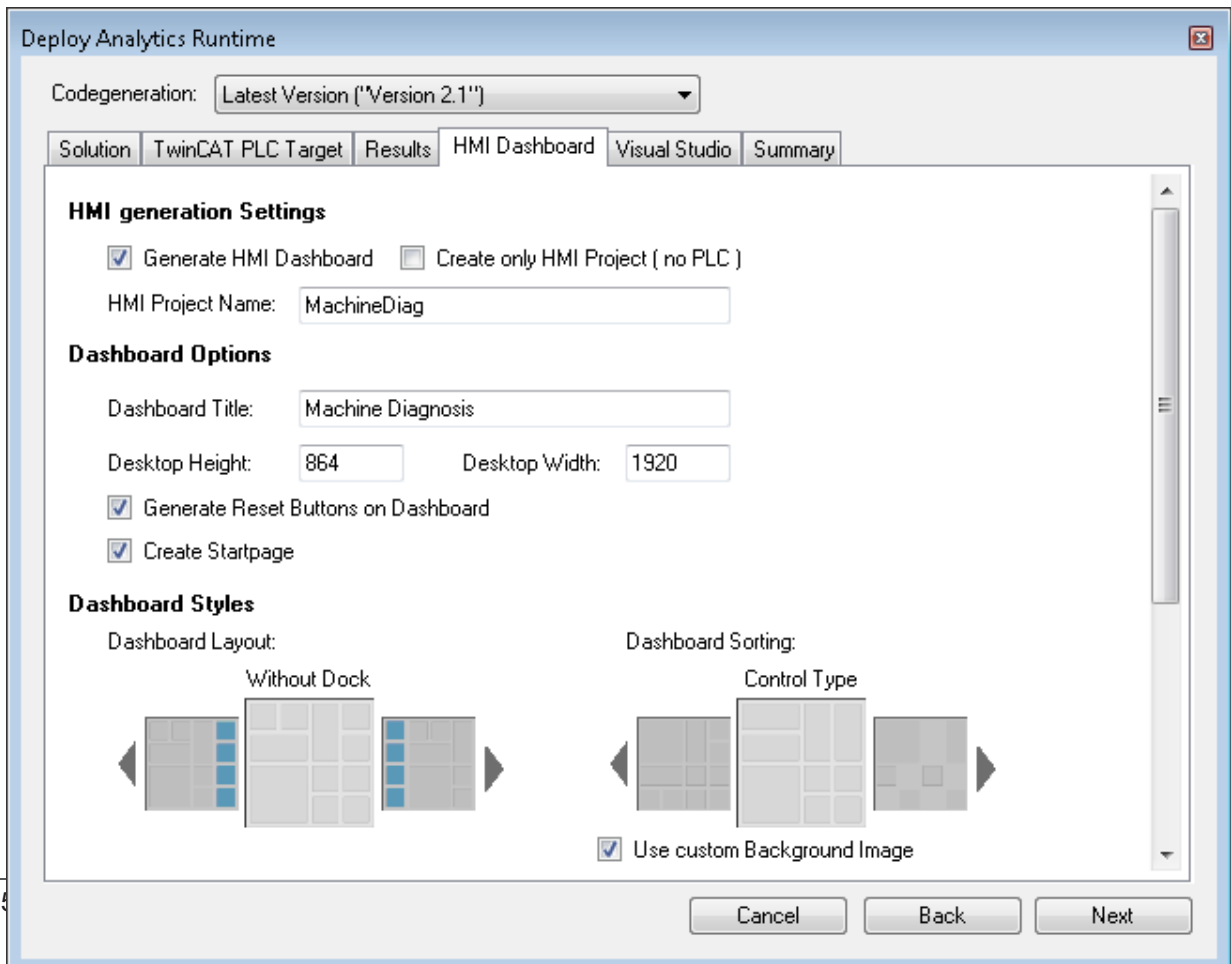
- In the **TwinCAT PLC Target** tab you can select the ADS target system that runs the TwinCAT Analytics Runtime (TF3550). The created project is immediately executable. For this purpose you can set the Activate PLC Runtime option. In addition, it can be selected that a boot project is created directly.



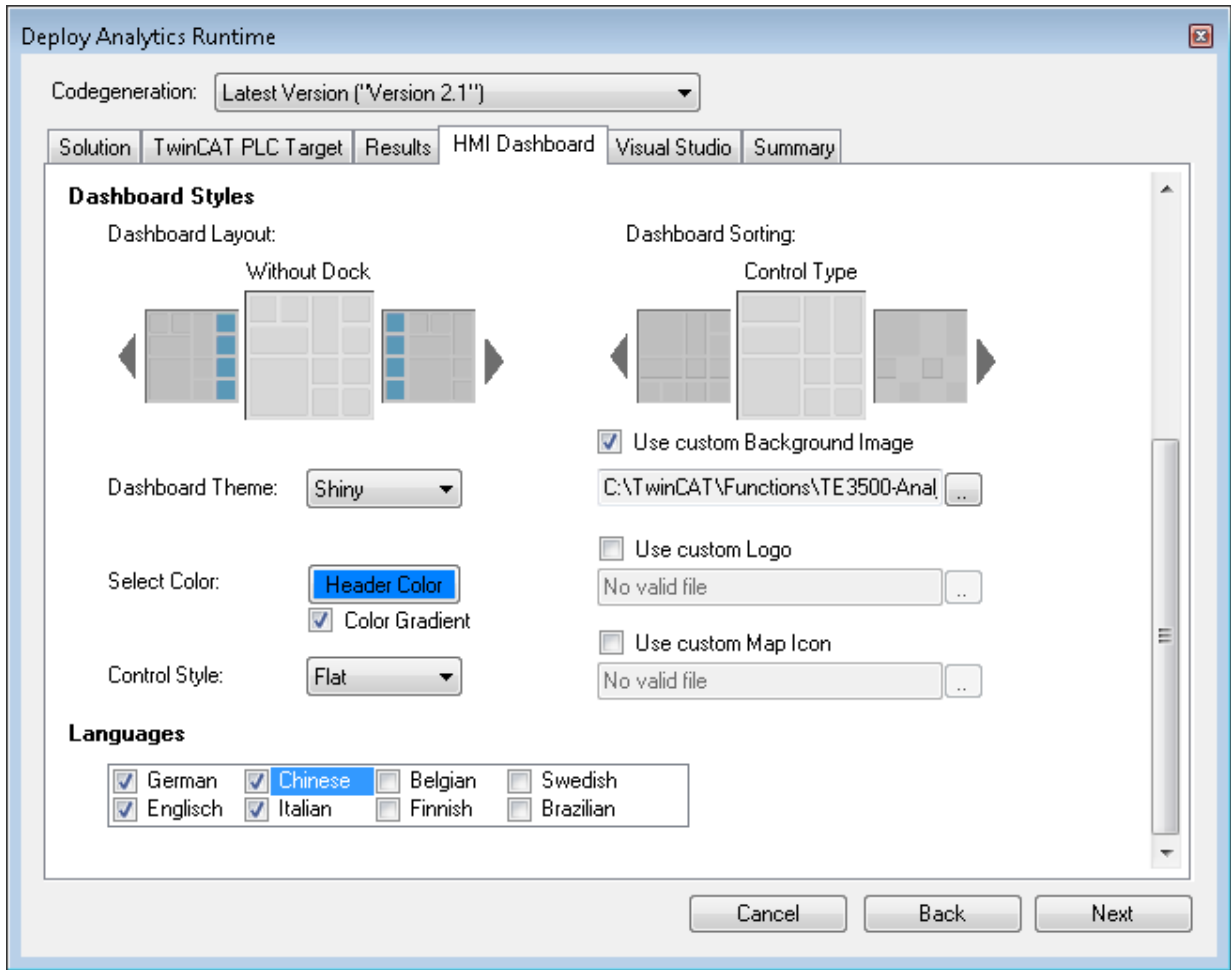
- Especially for virtual machines, it is important to run the project on isolated cores, which is also an option in this tab. The next tab **Results** is needed only if you have selected the **Stream Results** option in the algorithm properties. If you want to send results, you can decide here in which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



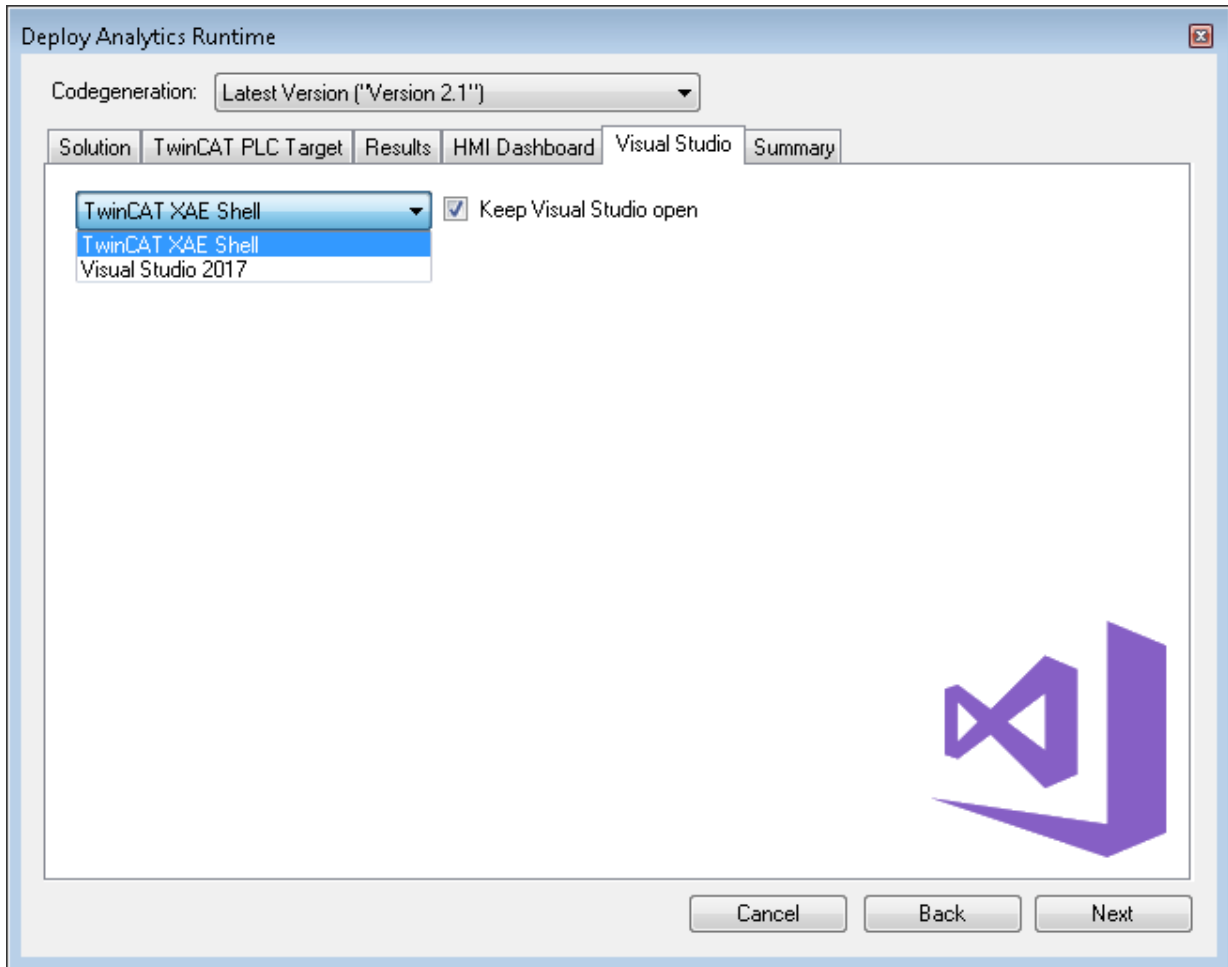
Downsampling of the results is possible by specifying a cycle time. The next tab is for the **HMI Dashboard**. A prerequisite for the automatic generation of the dashboard is the selection of HMI Controls for the corresponding algorithms whose results are to be displayed in the dashboard.



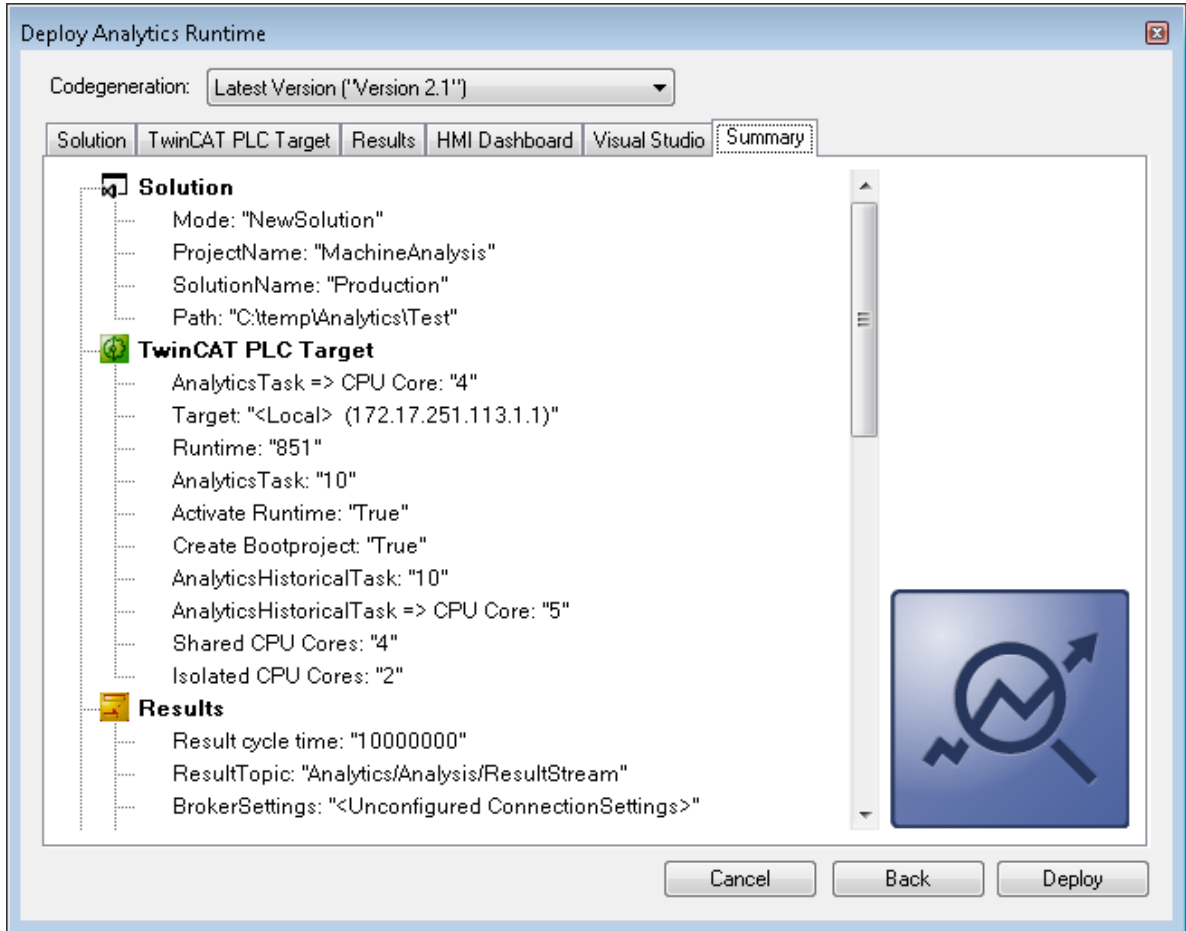
- You can choose different options for your Analytics Dashboard, such as a start page with a map, layouts, sorting algorithms, custom colors and logos. If you select multiple languages for the Analytics Controls, a language switching menu will also be generated.



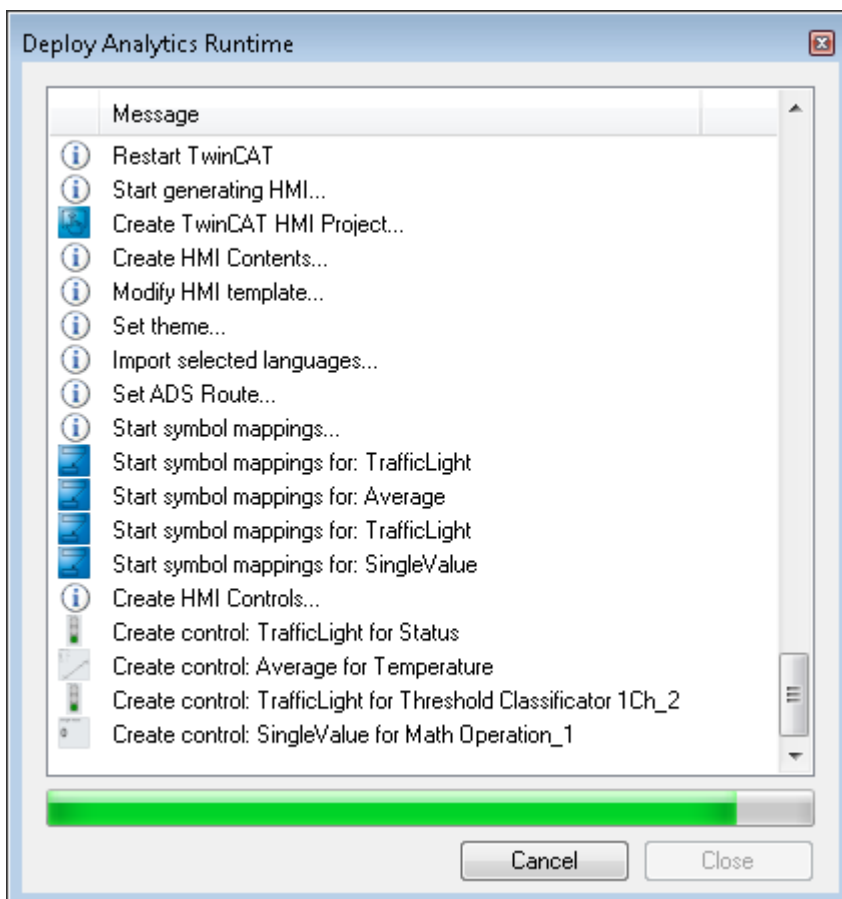
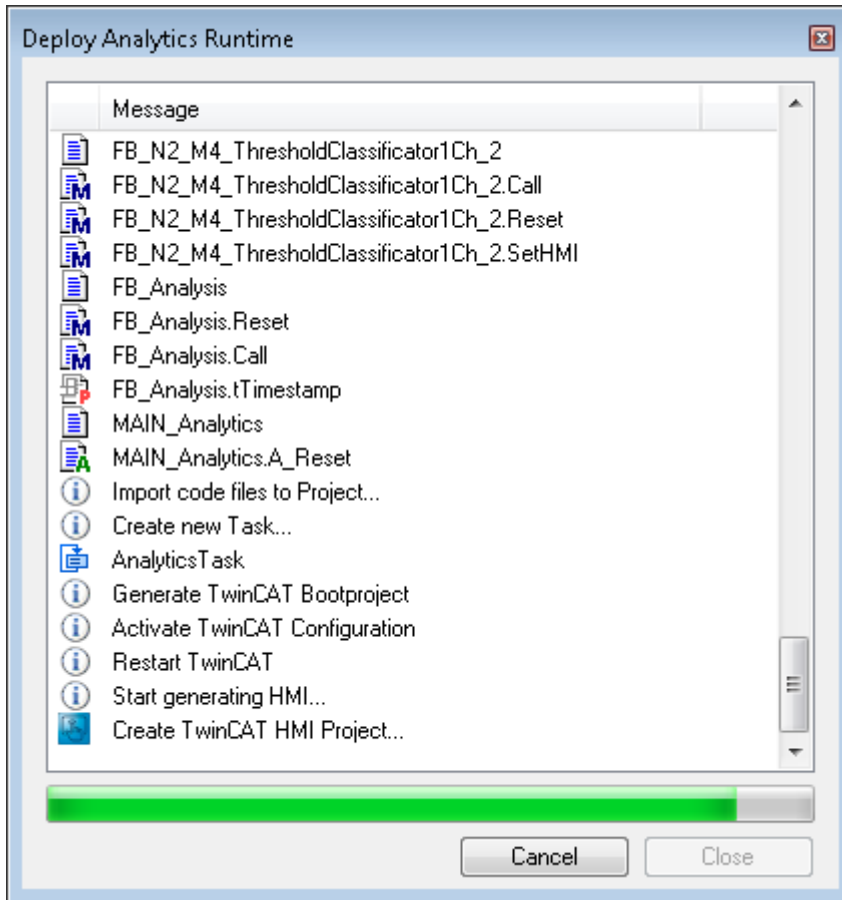
5. Select one of the installed versions of Visual Studio® and, whether the instance should start visibly or just be set up and activated in the background.



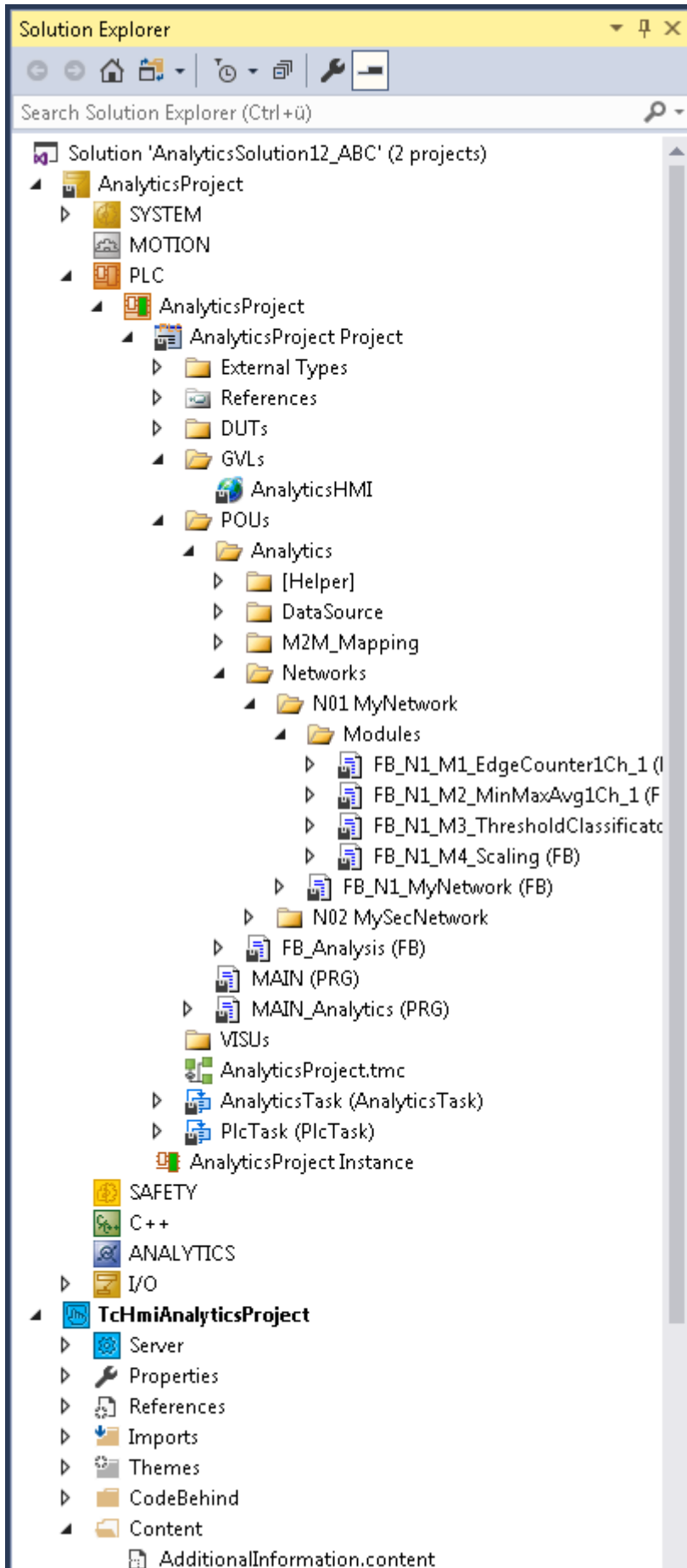
⇒ At last you can find an overview.



- 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.

Modes of Operation

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 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, whereat <>-brackets indicate variables as opposed to the other parts that are fixed.

Component	Topic Structure	Format	Purpose
Description	<MainTopic>/<StreamTopic>/Desc	JSON	Informs if there is a stream source online or offline. Includes timestamp of info.
Tx-Description	<MainTopic>/<StreamTopic>/Bin/Tx/Desc	JSON	Informs about the transmission parameters when a stream source is active 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**. The system manager configuration allows a stream to be started by default as soon as TwinCAT starts in run-mode. Additionally, streams can be started and stopped from PLC code.

When connected to a broker, the logger first sends the stream description followed by the Tx-description and symbol info as soon as the stream starts. This enables recipients to take all measures that are needed before data arrives. Finally, data is sent cyclically.

For the **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. Inside the stream's directory, a dedicated .tas-file holds the symbol info whereas .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:



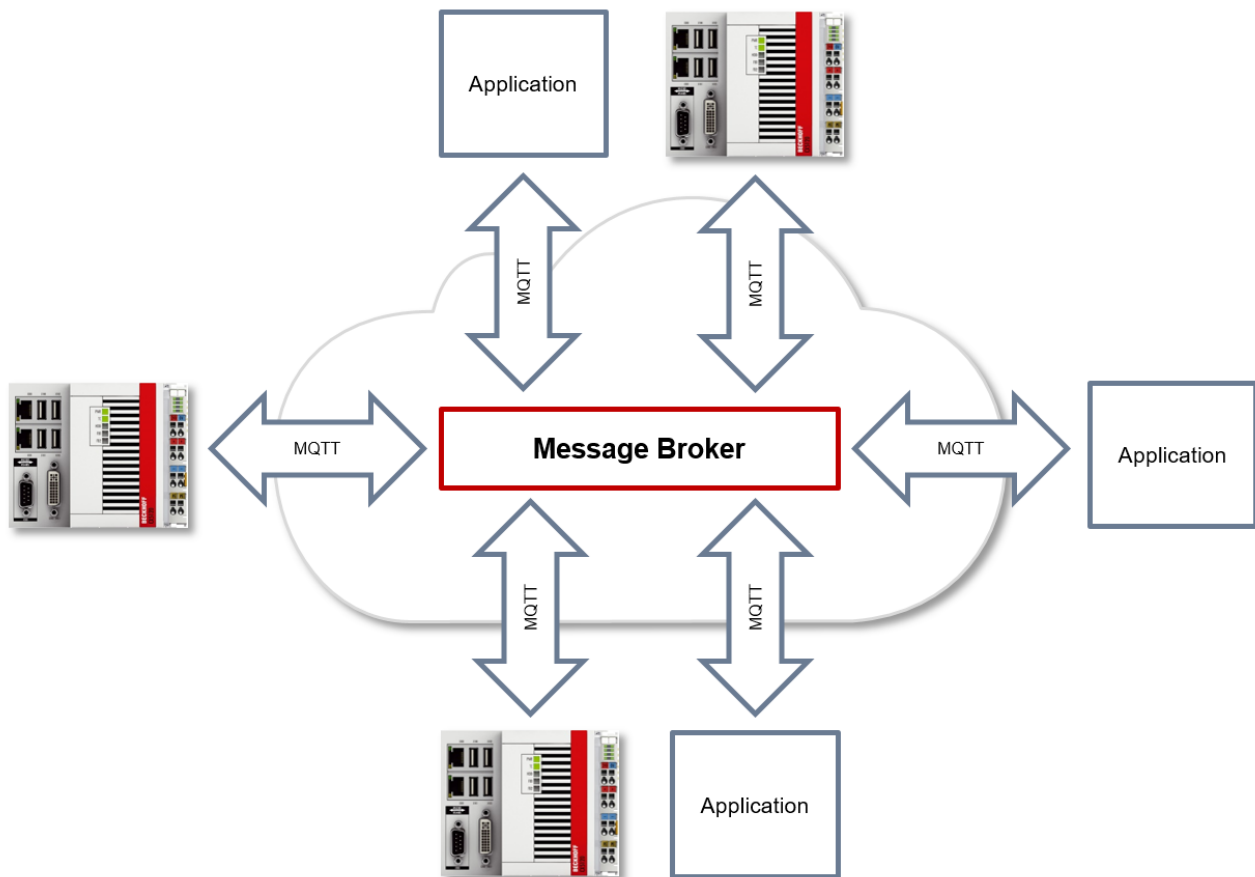
For every stream source and every task that drives variables of that source, a stream is created.

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_lotMqttClient and FB_lotMqtt5Client 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

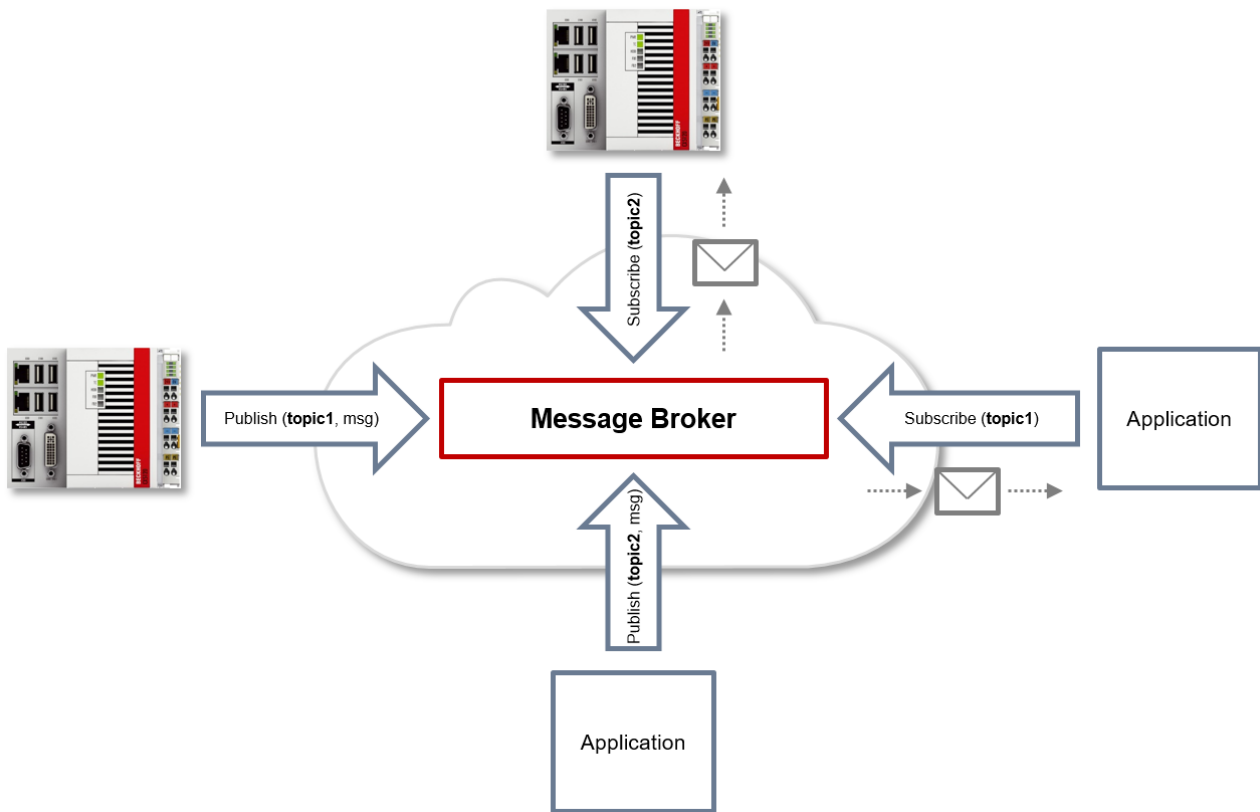
i Note that the data type and the formatting of the content must be known to the sender and receiver side, particularly when binary information (alignment) or strings (with or without zero termination) are sent.

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)

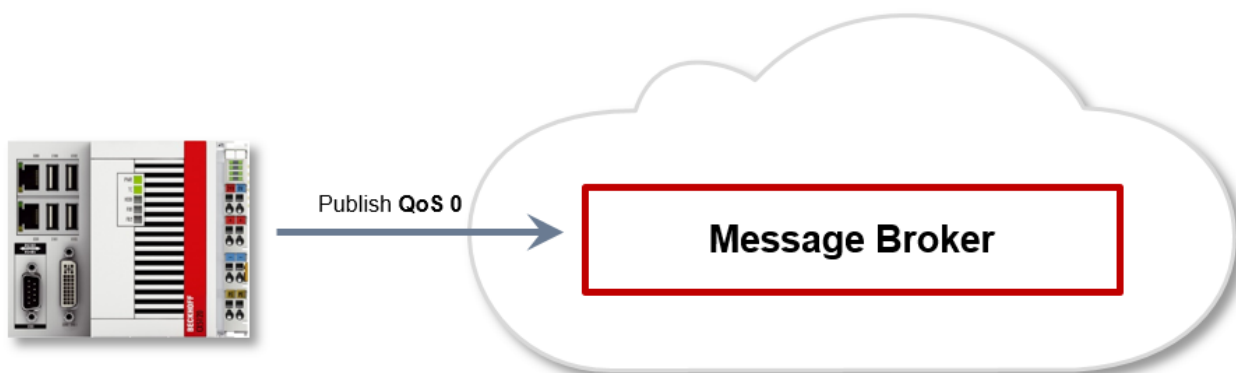
QoS is an arrangement between the sender and receiver of a message with regard to guaranteeing of the message transfer. MQTT features three different levels:

- 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.

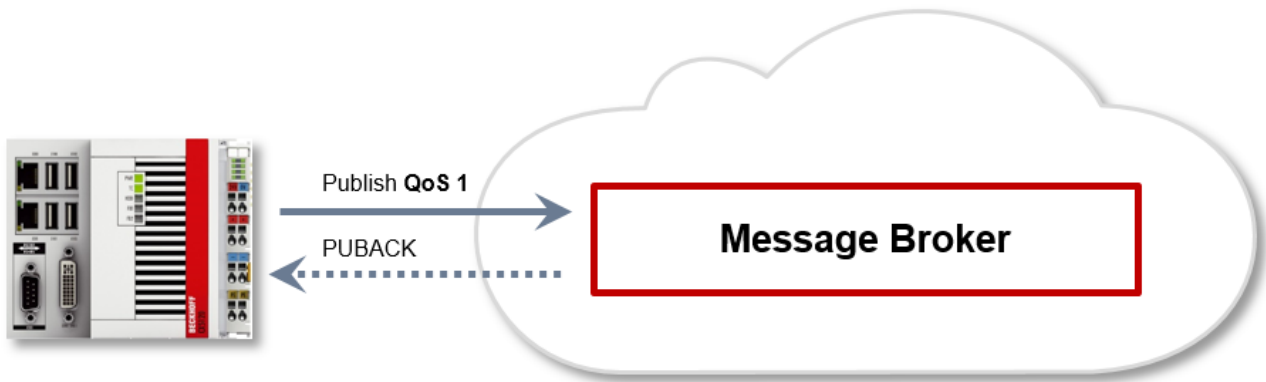
QoS-Level 0

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



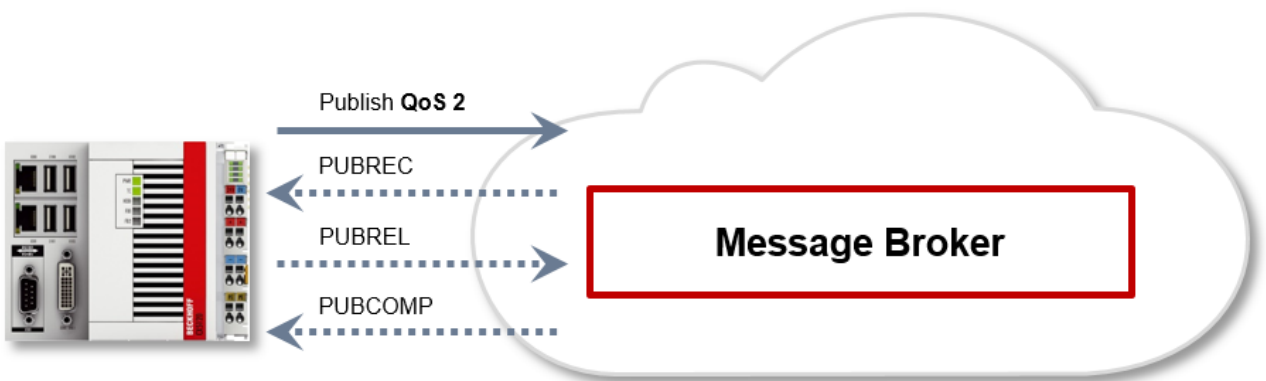
QoS-Level 1

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)

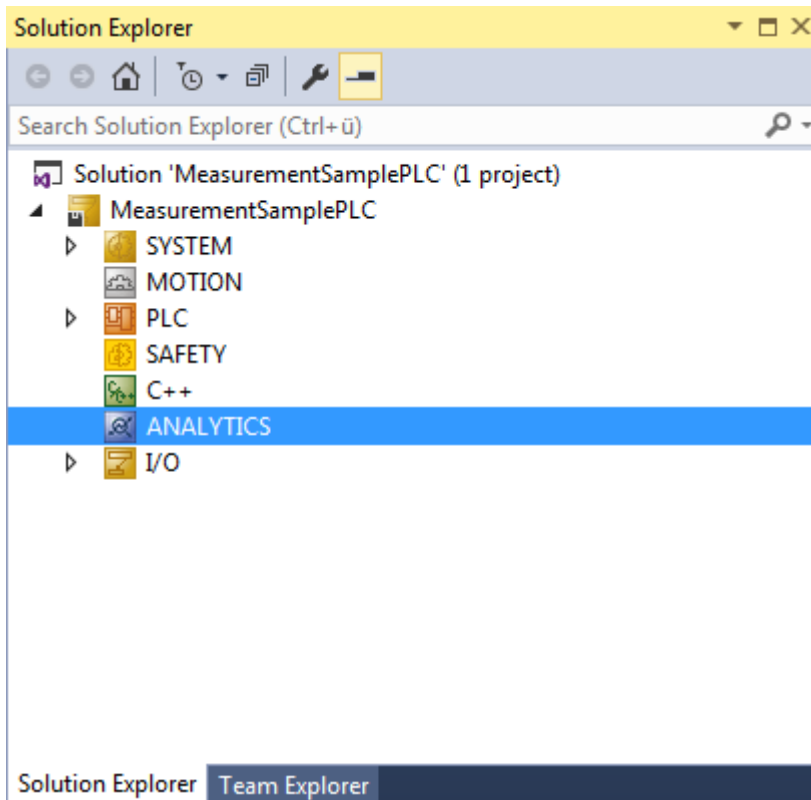
5.3 Data Compression

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.

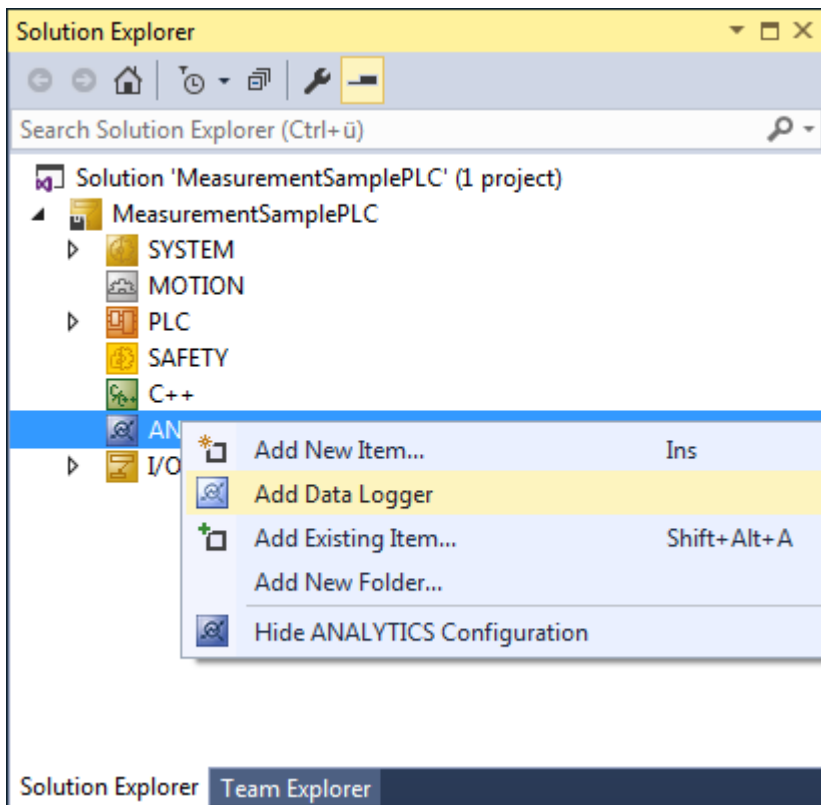
6 Configuration

6.1 Basic settings

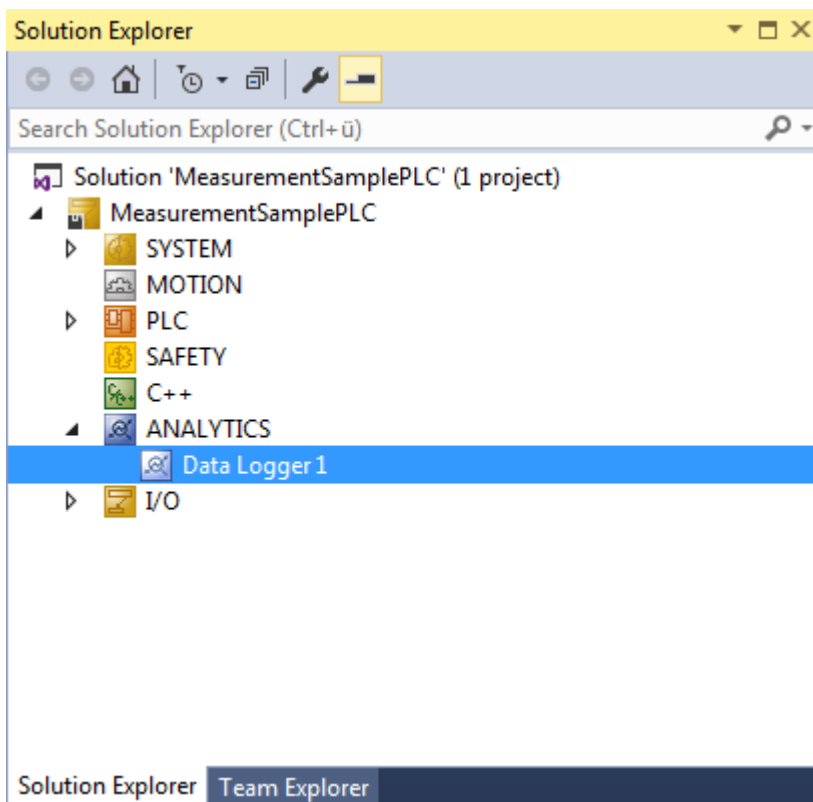
To configure the TwinCAT Analytics Logger the user is provided with a dedicated Analytics configuration inside of a XAE project.



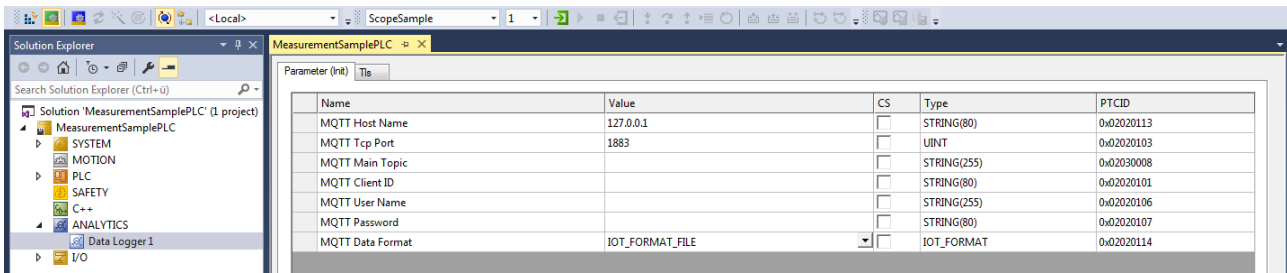
To add a data logger choose the respective item in the context menu of this configuration node.



This may either result in the additional Data Logger node alone or subordinate stream nodes in case there already are variables that can be logged.



By double clicking the new Data Logger node the editor window will be open. In the Parameter tab you can make your specific Analytics Logger settings.



- **Data Format:** Here the user can choose between IOT_FORMAT_FILE and IOT_FORMAT_BINARY. By using the FILE format the Analytics Logger stores the data in local binary files. The files will be stored in C:\TwinCAT\3.1\Boot\Analytics. By using MQTT_BINARY the data will send to the configured MQTT Message Broker.
- **Data Compression:** Data compression can be switched on and off here.
- **Max. Compression Compare Width:** Sets the compression mode.
- **MQTT Host Name:** Provide here the host name or IP address of your native MQTT Message Broker.
- **MQTT Tcp Port:** Set the Tcp port for the communication here. Default MQTT port: 1883
- **MQTT Main Topic:** It is possible to provide an own and individual main topic. Sample: Beckhoff/Verl/Production/Drives/Machine5 – the Analytics Logger will add automatically his own specific subtopics: Beckhoff/Verl/Production/Drives/Machine5/Bin/Tx/Data
- **MQTT Client ID:** The client identifier is an identifier of each MQTT client connecting to a native MQTT Message Broker. It should be unique per Broker.
- **MQTT User Name:** MQTT allows to send a username for authenticating the client.
- **MQTT Password:** MQTT also allows to send a password for authenticating the client and authorization.

It is possible to have multiple Logger in one TwinCAT project to communicate data to different MQTT Message Broker or to have partly a storage in a local binary file.

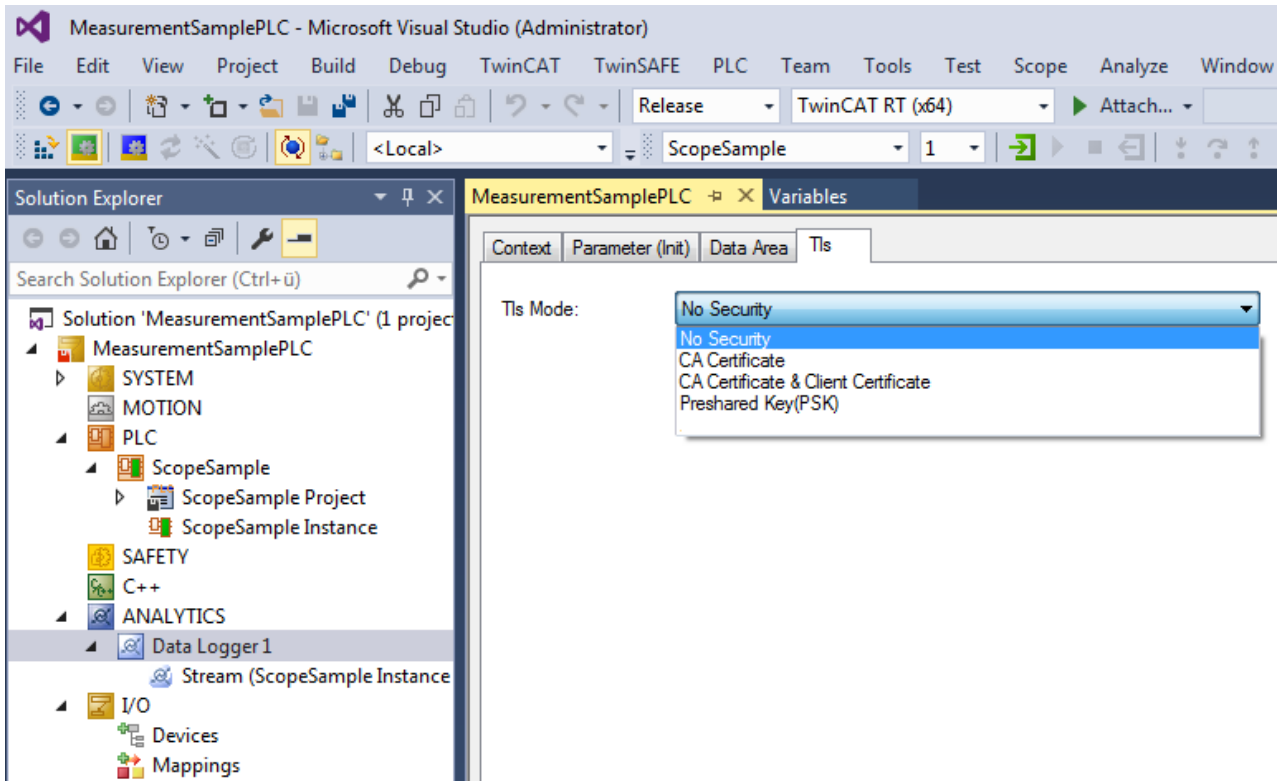
6.1.1 TLS

TLS (Transport Layer Security) provide a secure communication channel between a client and a server. At its core, TLS is cryptographic protocols which use a handshake mechanism to negotiate various parameters to create a secure connection between the client and the server. The TwinCAT Analytics Logger is supporting TLS version 1.2.

MQTT communication with TLS

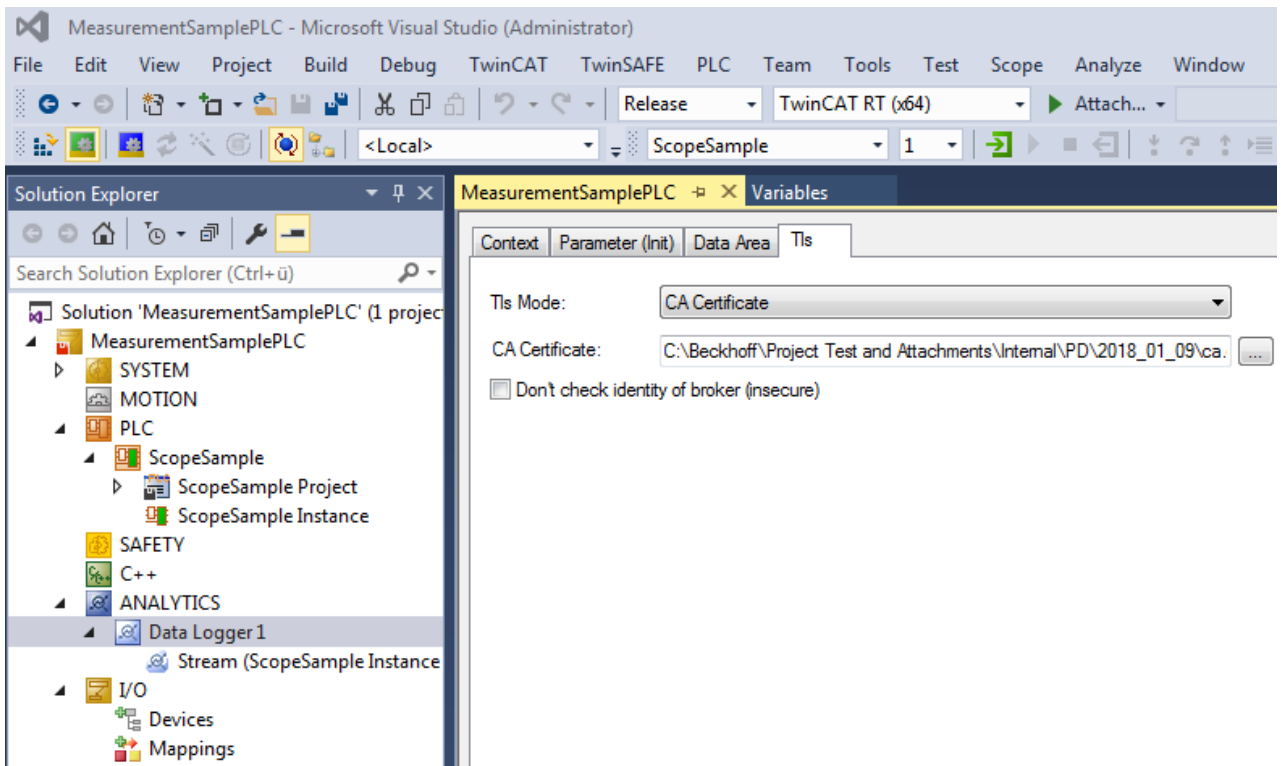
i By the usage of certificates the TCP port 8883 is exclusively reserved for MQTT over TLS!

On the TLS tab of the Data Logger your first choice is the TLS Mode in a drop down box. Depending on the Message Broker it is possible to use different TLS mechanism/modes. The Analytics Logger is supporting the modes CA Certificates, CA Certificates & Client Certificate and the Preshared Key (PSK) mode.



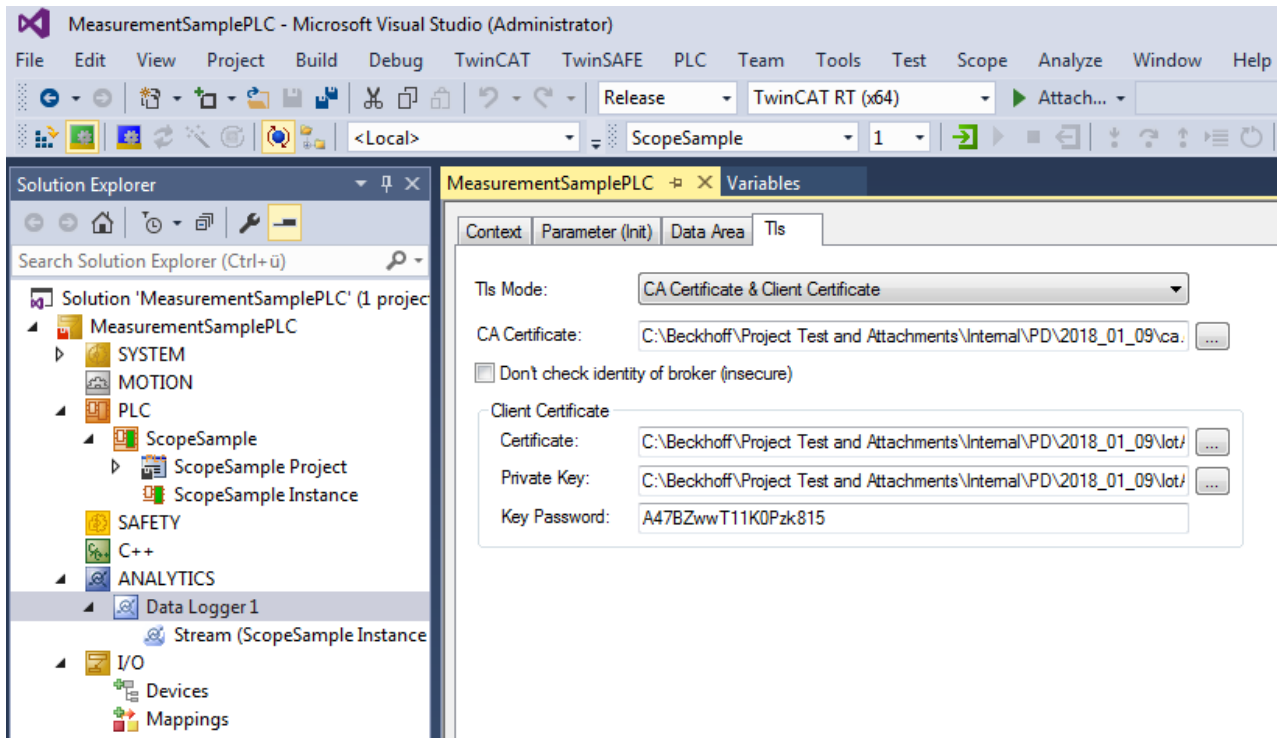
CA Certificate

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



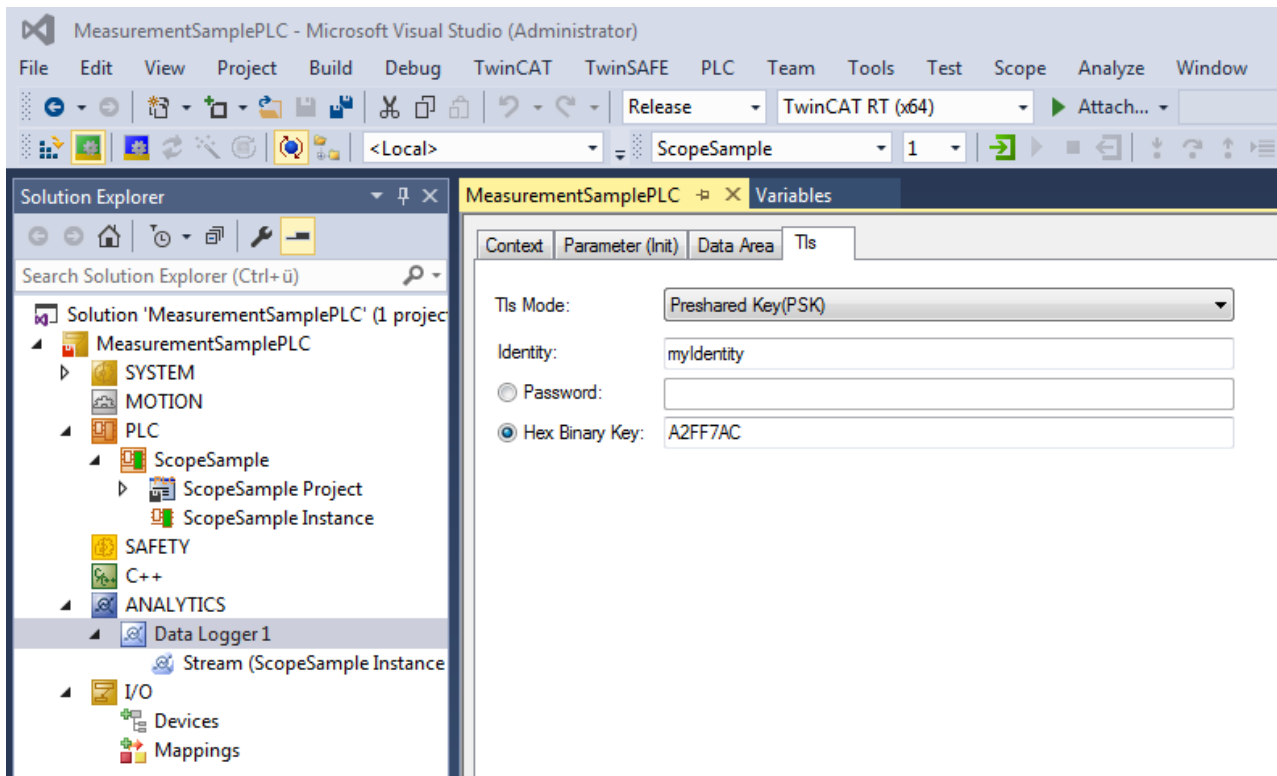
CA Certificate & Client Certificate

Encryption and authentication via TLS can also be accomplished through a certificate authority (CA). The CA provides a signature via 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.



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 packages.

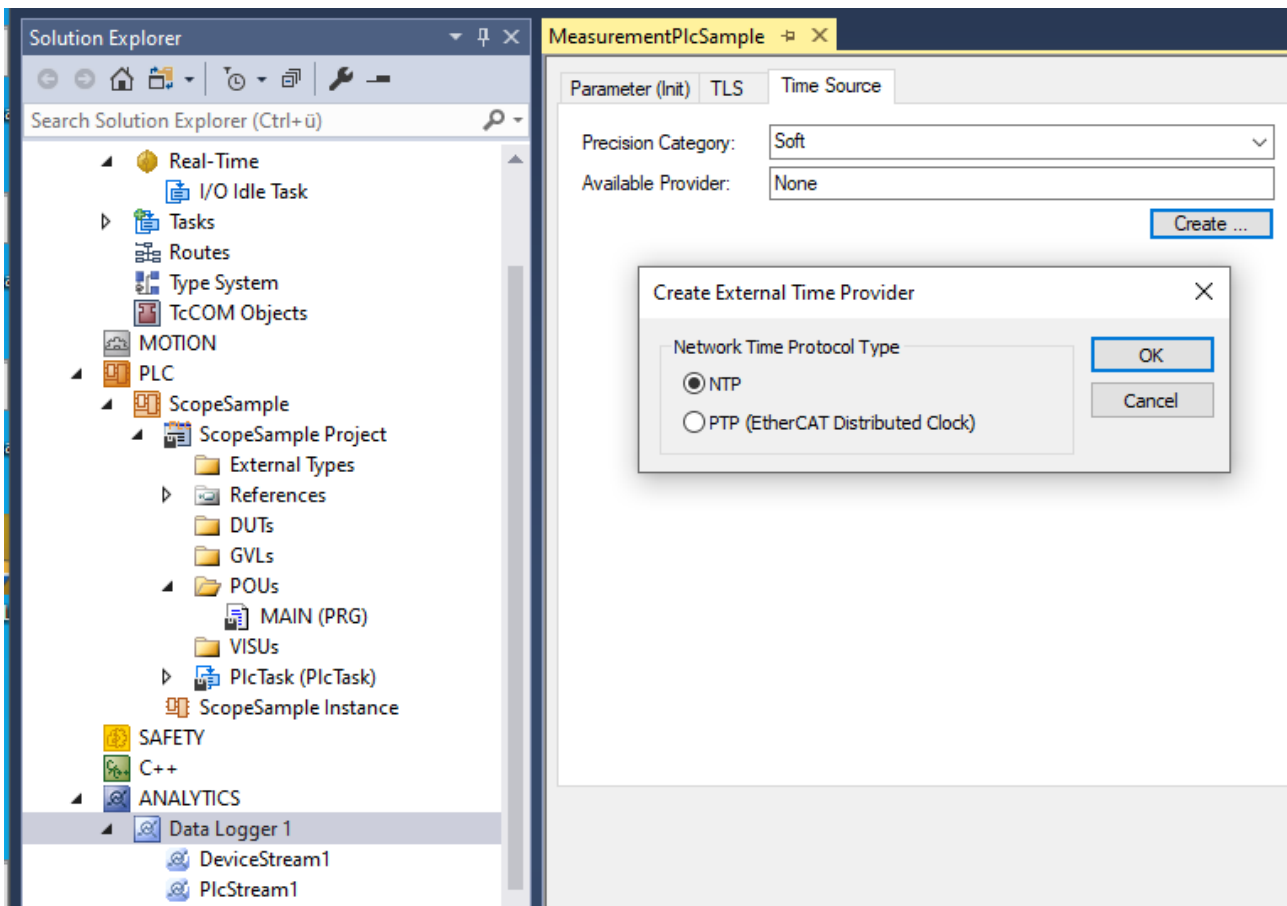


6.1.2 Timestamp correction

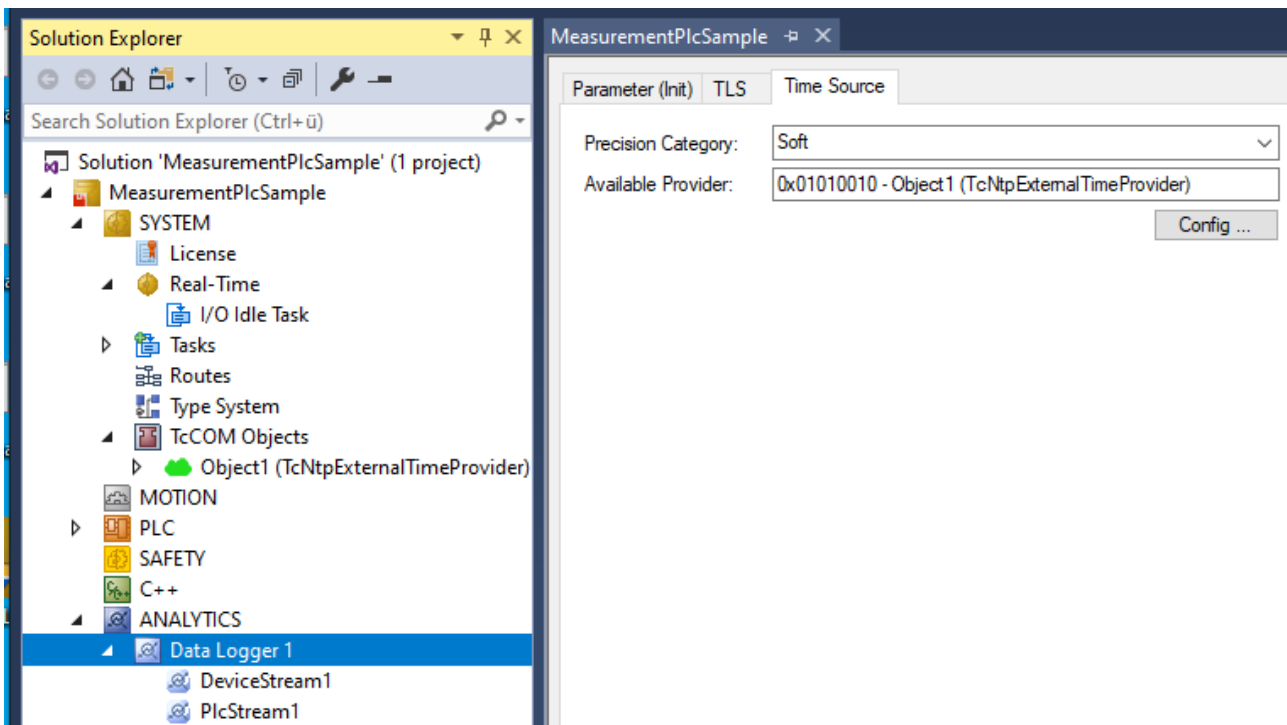
The TwinCAT time, which is used by default for timestamps in the Data Logger, deviates more and more from the actual system time the longer a controller is in run mode. This is due to the fact that different hardware counters are used as clock generators. With the concept of timestamp correction, which TwinCAT offers, it is possible to append corrected timestamps to the recorded samples. The correction using the External Time Provider, which TwinCAT provides, 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).

The settings for timestamp correction can be found under the **Data Logger** project node in the tab **Time Source**.

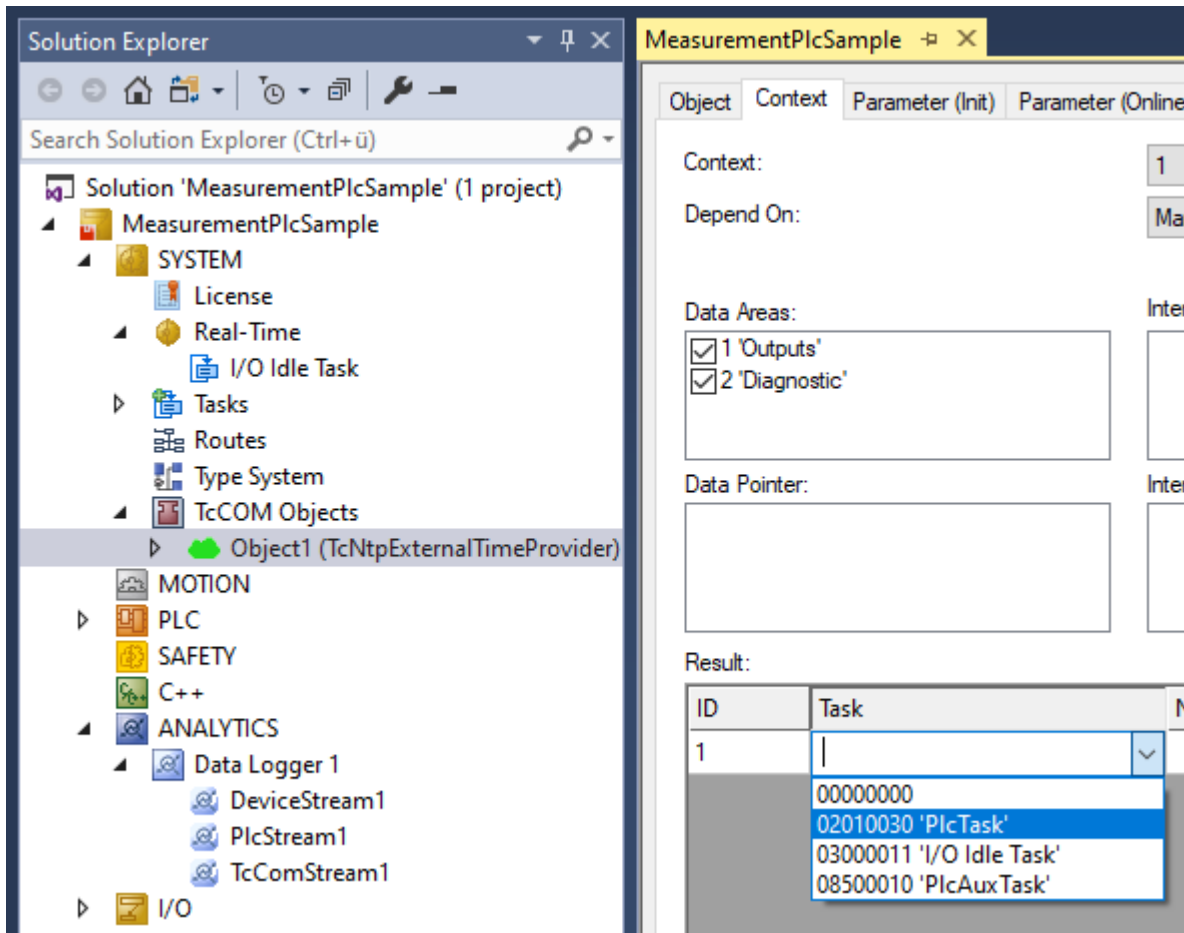
If the desired Precision Category is selected, any External Time Provider is displayed. If none exists yet, a provider can be created via the button **Create**.



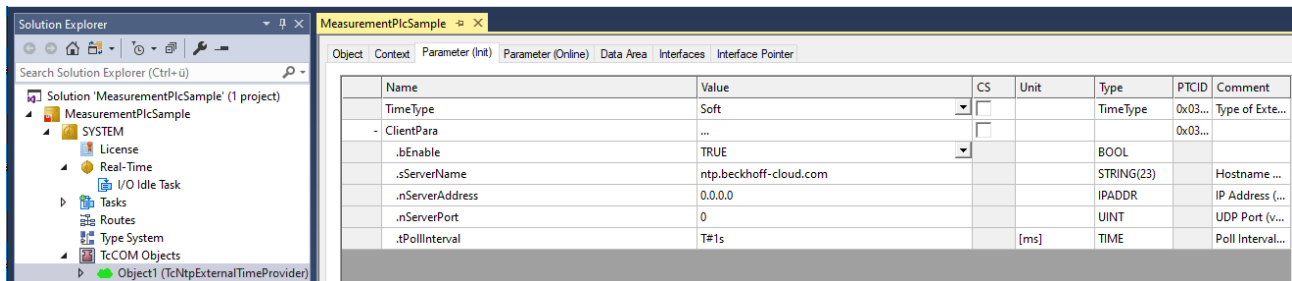
This then appears in the project under the TcCom-Objects node, which you can access via the button **Config**.



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



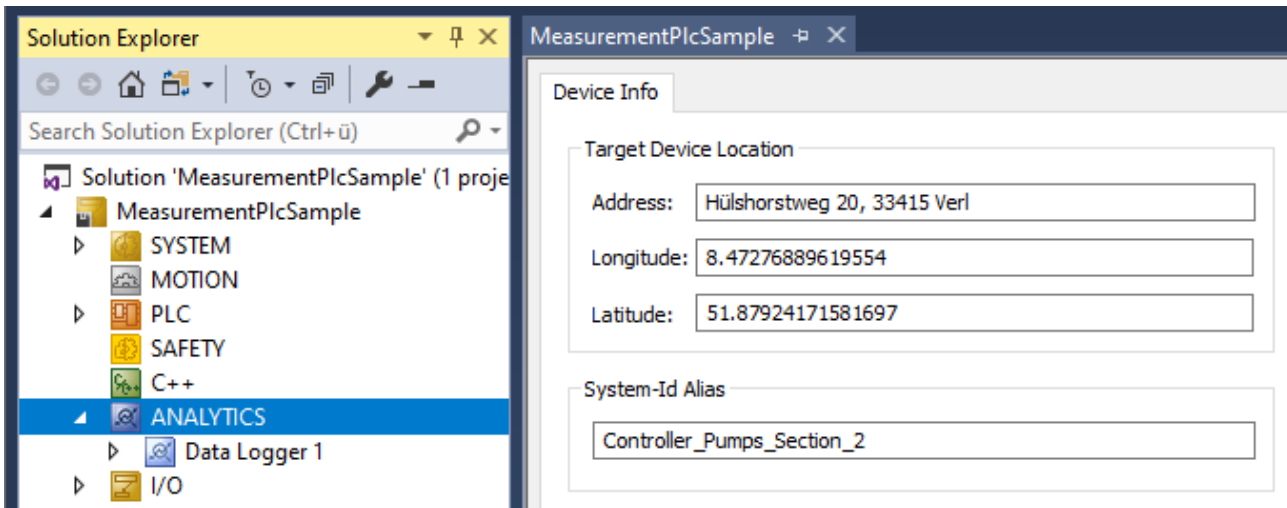
In addition, the cycle time of the provider's synchronization with the NTP server and the server host name can be set via the **Init parameter**.



This creates timestamp corrections that the Logger uses to correct an offset resulting from the difference between the TwinCAT time and the NTP-synchronous time.

6.1.3 Device-specific information

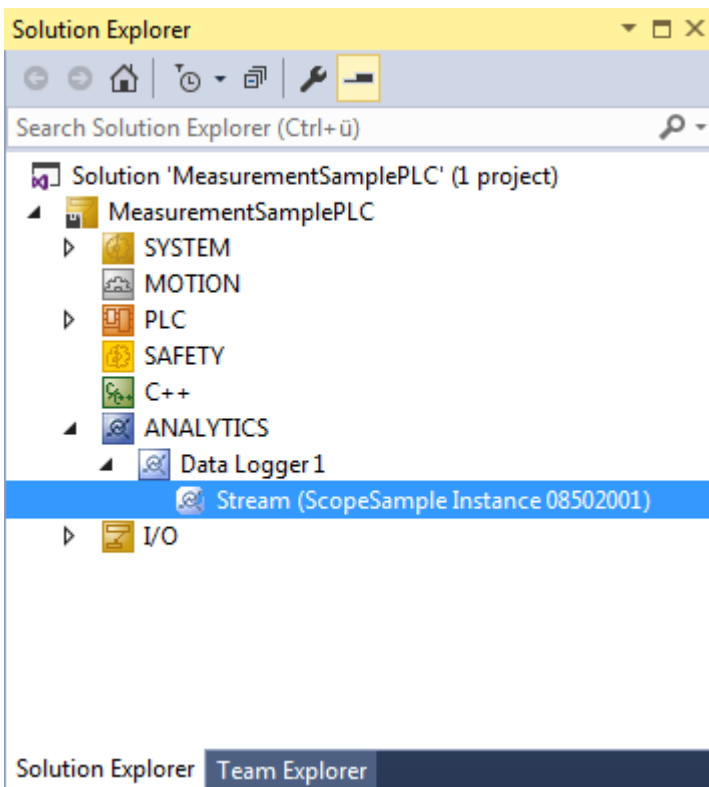
You can enter information about the control device under the Analytics project node in the tab **Device Info**. This includes the address, coordinates and a system-ID alias, which makes it easier for the user to identify the device in the analytics workflow.



This information is sent as part of the Stream Description on the corresponding MQTT topic and is the same for all Data Loggers and Streams.

6.2 Data Streams

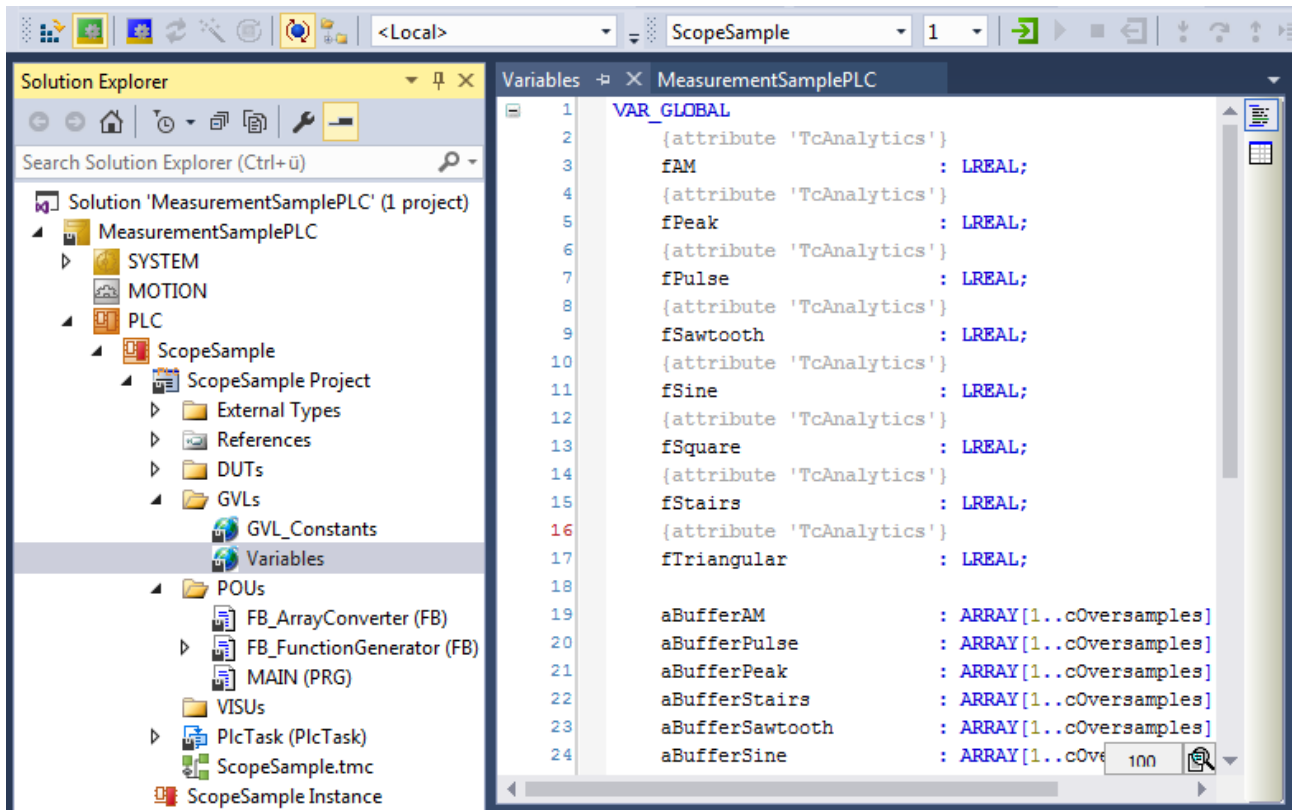
If variables are available for recording they will be shown automatically as a so called Stream. At the moment it is possible to record data directly from the EtherCAT process image or from the PLC application.



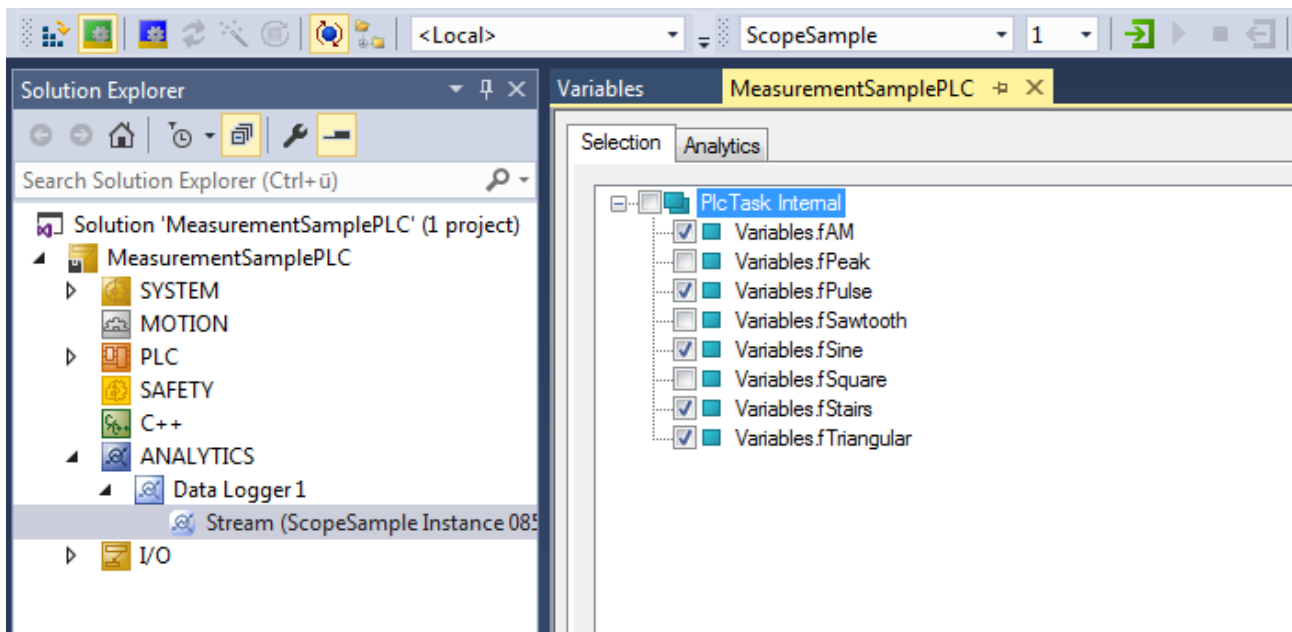
PLC Application:

If variables from PLC should be recorded by the Analytics Logger, the user must set an attribute in front of each variable in the variable declaration.

The attribute syntax is: {attribute 'TcAnalytics'}

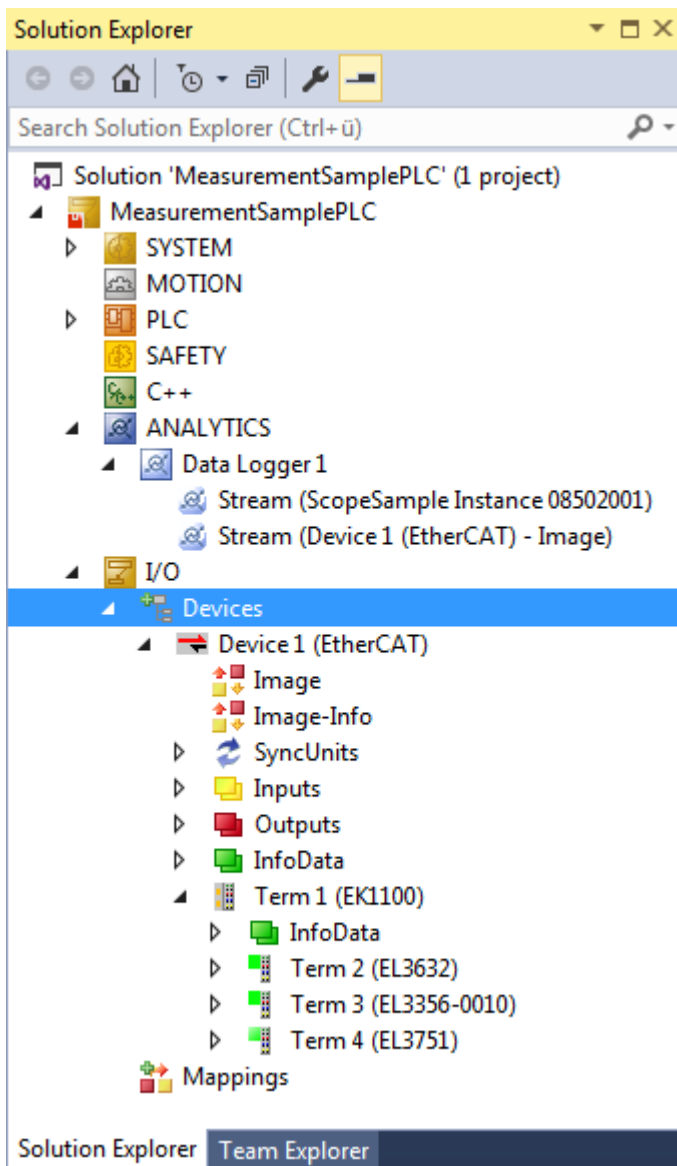


As already described a new stream is automatically add to the Analytics Logger configuration **after rebuild** of the PLC project. In the stream a *Selection* tab is available to choose finally the variables by checkboxes who should be recorded.

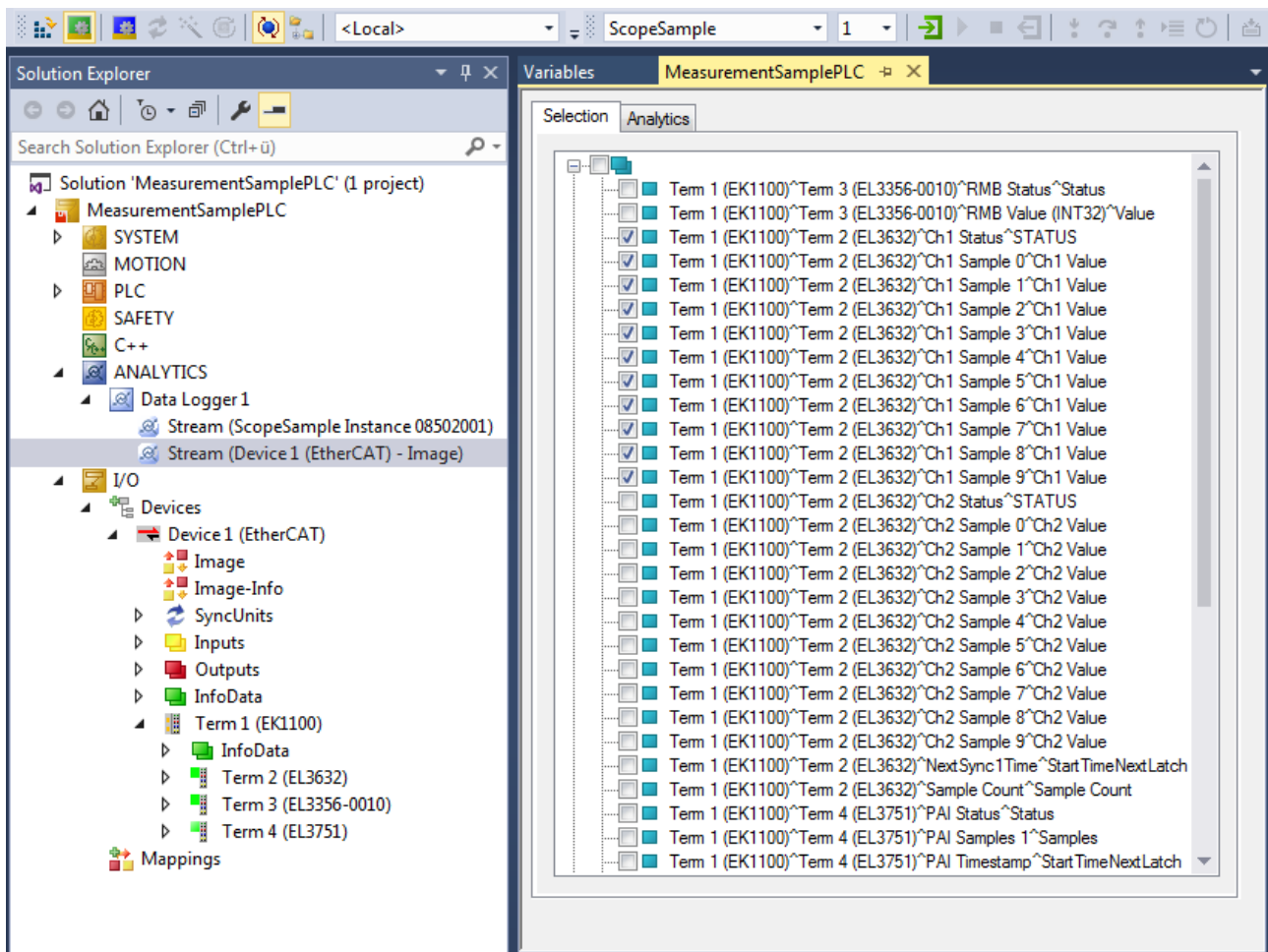


Process Image:

If an EtherCAT process image is available in the given configuration, an additional stream will be shown under the Data Logger tree node.



On the *Selection* tab the user can choose again the values who should be recorded by the Analytics Logger.

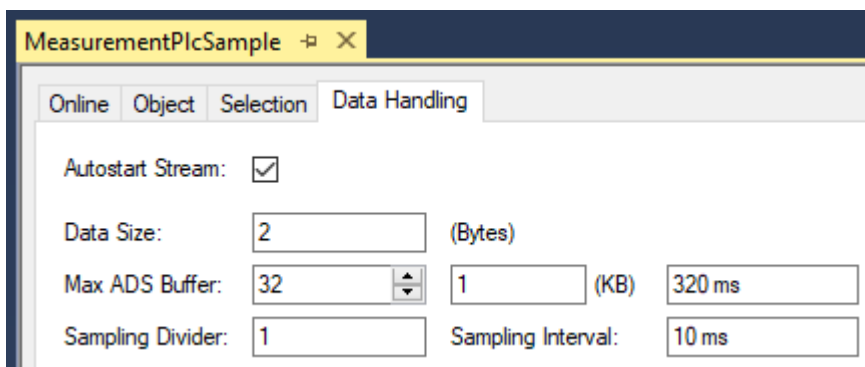


Start Record

With the activation of the TwinCAT configuration the Analytics Logger starts logging. Depending on the basic settings to a MQTT Message Broker or into a local binary file. By the given format there are different settings possibilities on the Data Handling tab of each Stream. See therefore the following chapter.

6.2.1 Data Handling

In the Data Handling tab you can make general settings for the package size of the recorded data. Depending on the given data format it gives additional settings.



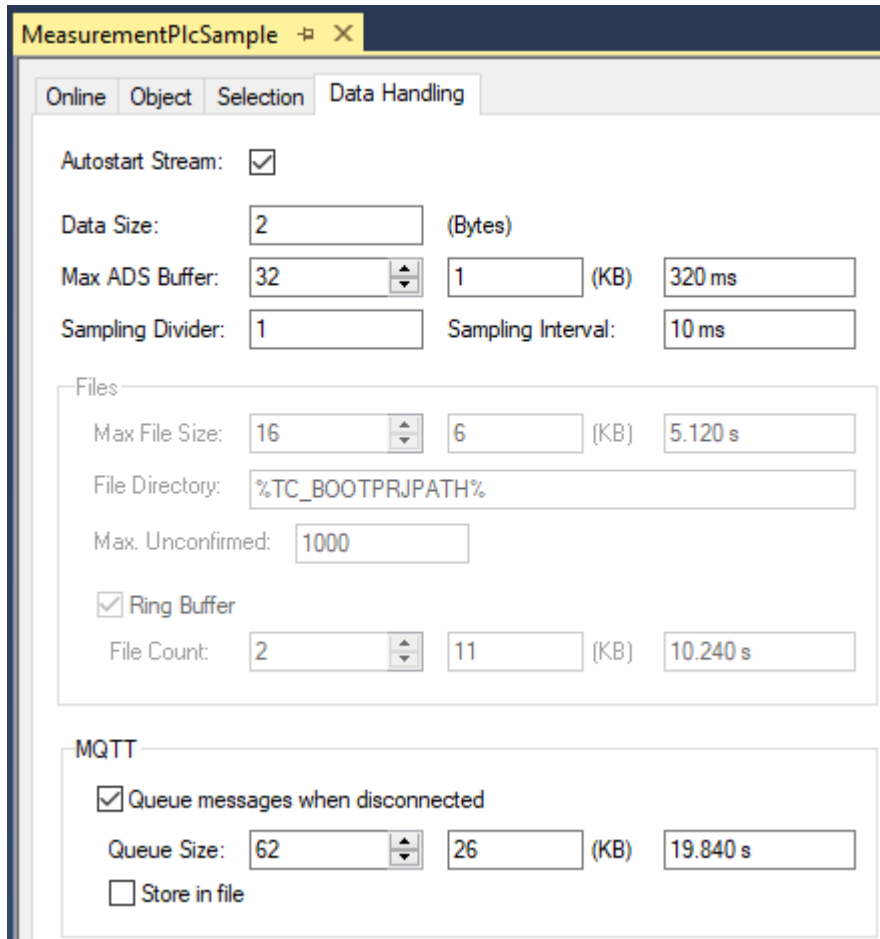
Data Size: This is a read-only value and provide the size of the given variable selection for this stream.

Max ADS Buffer: Here you can set the number of buffered cycles before write to file or send to Message Broker. By 1ms cycle time and 32 buffers the Analytics Logger buffered 32ms before operated. It is an individual setting depending on the system resources.

Max File Size: Only available by *IOT_FORMAT_FILE*. You can give the maximum file size before starting with a new file. The files will be stored under C:\TwinCAT\3.1\Boot\Analytics.

Data format: MQTT

If the data format is *IOT_FORMAT_BINARY* it is possible to activate an additional checkbox for queuing messages by disconnect.



File Store: If this option is activated the queued messages will stored in a temporary file at the hard disk. Otherwise the data will be stored in the RAM when connection to the Message Broker is disconnected.

Queue Size: Is the number of configured ADS Buffer which should be stored in the case of a lost connection.

Data format: File

If the data format is *IOT_FORMAT_FILE* it is possible to activate an additional checkbox for a ring buffer.

The screenshot shows the 'MeasurementPlcSample' configuration window with the 'Data Handling' tab selected. The window has a title bar with a close button and a tab icon. Below the title bar are four tabs: 'Online', 'Object', 'Selection', and 'Data Handling'. The 'Data Handling' tab contains the following settings:

- Autostart Stream:
- Data Size: (Bytes)
- Max ADS Buffer: (KB)
- Sampling Divider: Sampling Interval:
- Files section:
 - Max File Size: (KB)
 - File Directory:
 - Max. Unconfirmed:
 - Ring Buffer
 - File Count: (KB)

File Count: You can set up a ring buffer system for the binary files. With the File Count parameter you can give a number of files who should be part of the ring buffer. The ring buffer time depends on the given Max File Size.

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 TwinCAT is set 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, nSamplesIssued 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 TwinCAT is set 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 TwinCAT is set 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.
AddlotSymbol	BOOL	Add a symbol of type I_ALYC_lotSymbol to the internal symbol list whose values are to be received. TRUE if successful.
ReleaselotSymbol	BOOL	Remove a symbol of type I_ALYC_lotSymbol from the internal symbol list. TRUE if successful.
ReleaseAlllotSymbols	BOOL	TRUE if successful.
ContainslotSymbol	BOOL	TRUE if symbol of type I_ALYC_lotSymbol 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>
nNumlotSymbolsRegistered	UDINT	get/set	Number of lot 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 TwinCAT is set 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 [► 81]	
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.
AddlotSymbol	BOOL	Add a symbol of type I_ALYC_lotSymbol to the internal symbol list whose values are to be received. TRUE if successful.
ReleaselotSymbol	BOOL	Remove a symbol of type I_ALYC_lotSymbol from the internal symbol list. TRUE if successful.
ReleaseAlllotSymbols	BOOL	TRUE if successful.
ContainslotSymbol	BOOL	TRUE if symbol of type I_ALYC_lotSymbol is in the internal symbol list.

Properties

Name	Type	Direction	Description
nNumlotSymbolsRegistered	UDINT	get/set	Number of lot 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](#) [► 59], [FB_ALYC_FileStream](#) [► 60] and [FB_ALYC_MqttStreamHelper](#) [► 62] 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](#) [► 63] 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.

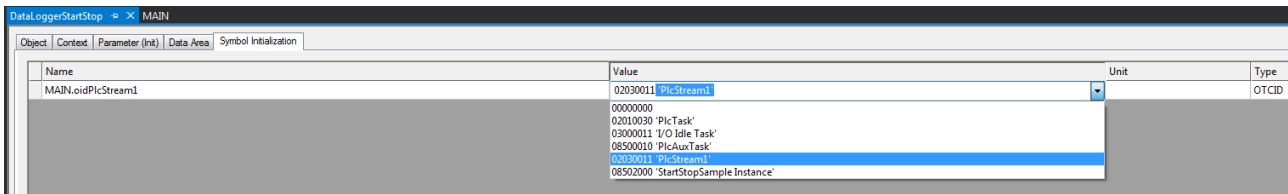


Fig. 1:

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 further questions, please contact our support team support@beckhoff.com.

Should I always use TLS with MQTT? [▶ 85]

Is it possible to have multiple connections? [▶ 85]

Is it possible to control the Analytics Logger by a PLC function block? [▶ 85]

?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?

!There is no special function block to control the Analytics Logger. But you can use an interface of the Analytics Logger to control him with easy commands like start and stop from the PLC. How to do see [this \[▶ 82\]](#) cheapter.

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

