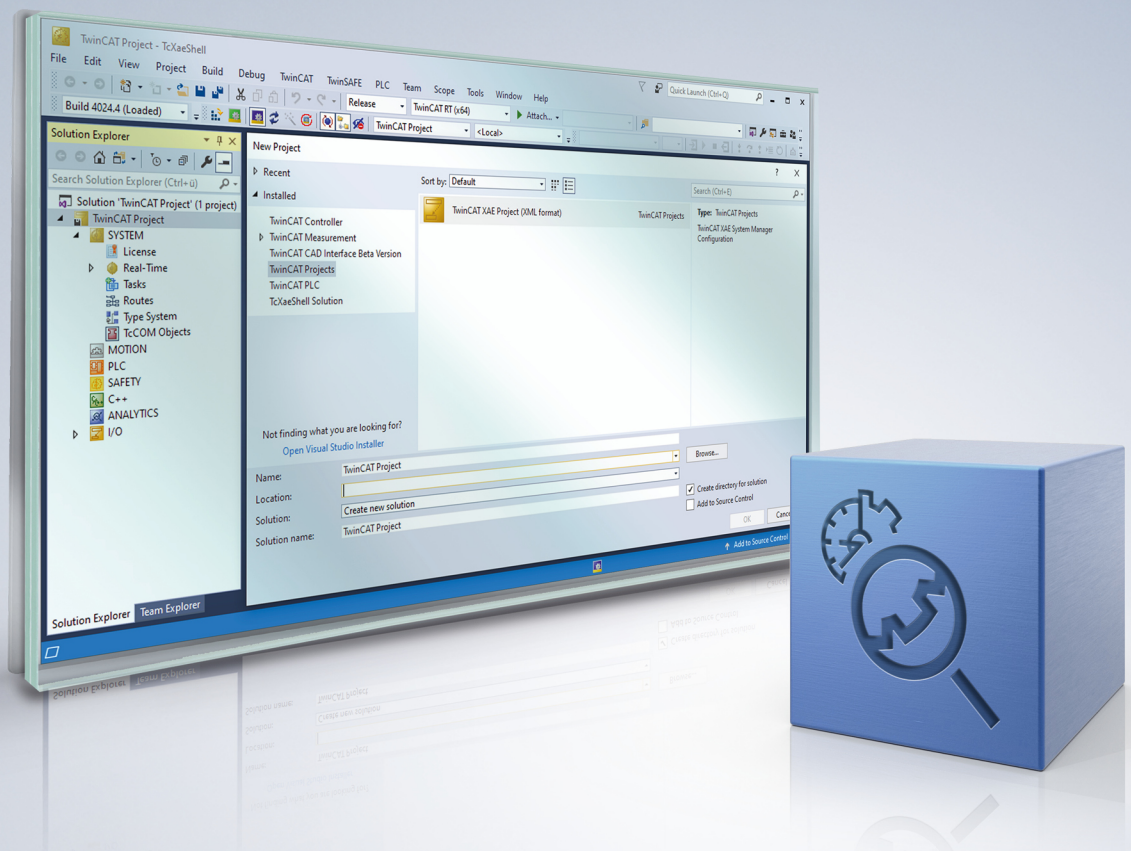
BECKHOFF New Automation Technology

Manual | EN

TF355x

TwinCAT 3 | Analytics Runtime



2024-08-29 | Version: 1.2.1

Table of contents

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

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 Runtime ensures that a continuous data analysis runs parallel to the actual machine applications. The PLC code that is downloaded to the Analytics Runtime can be automatically generated with the help of the engineering product TE3500 Analytics Workbench. Optionally, a complete Analytics Dashboard can be generated based on the HTML5-based TwinCAT HMI.

If you already have your own visualizations that you would like to integrate into continuous data analysis via OPC UA or ADS, we offer two different runtime products. TF3550 includes the HMI Server and the HMI Client Pack 3, while TF3551 does not include the HMI as a Runtime Base.

TF3550 components

- PLC Runtime (TC1200)
- Analytics PLC Library (TF3510)
- IoT connectivity with MQTT and HTTPS/Rest (TF6701 and TF6760)
- Analytics Storage Provider Library
- HMI Server (TF2000)
- HMI Client Pack 3 (TF2020)
- HMI Scope (TF2300)
- Scope Server (TF3300)

TF3551 components

- PLC Runtime (TC1200)
- Analytics PLC Library (TF3510)
- IoT connectivity with MQTT and HTTPS/Rest (TF6701 and TF6760)
- Scope Server (TF3300)
- Analytics Storage Provider Library

Since TwinCAT Build 4024.57, the number of sources to be analyzed (Analytics Logger, Analytics Data Exchange API, IoT Data Agent and EK9160) is no longer limited, no additional Controller Packs need to be licensed.

3 Installation

3.1 System requirements

The Analytics Runtime setup is an all-around setup. It includes a current version of TwinCAT 3.1 XAR and of the TwinCAT HMI Server.

Technical data	TF3550 TwinCAT 3 Analytics Runtime
Target System	Windows 7/8/10

3.2 Installation

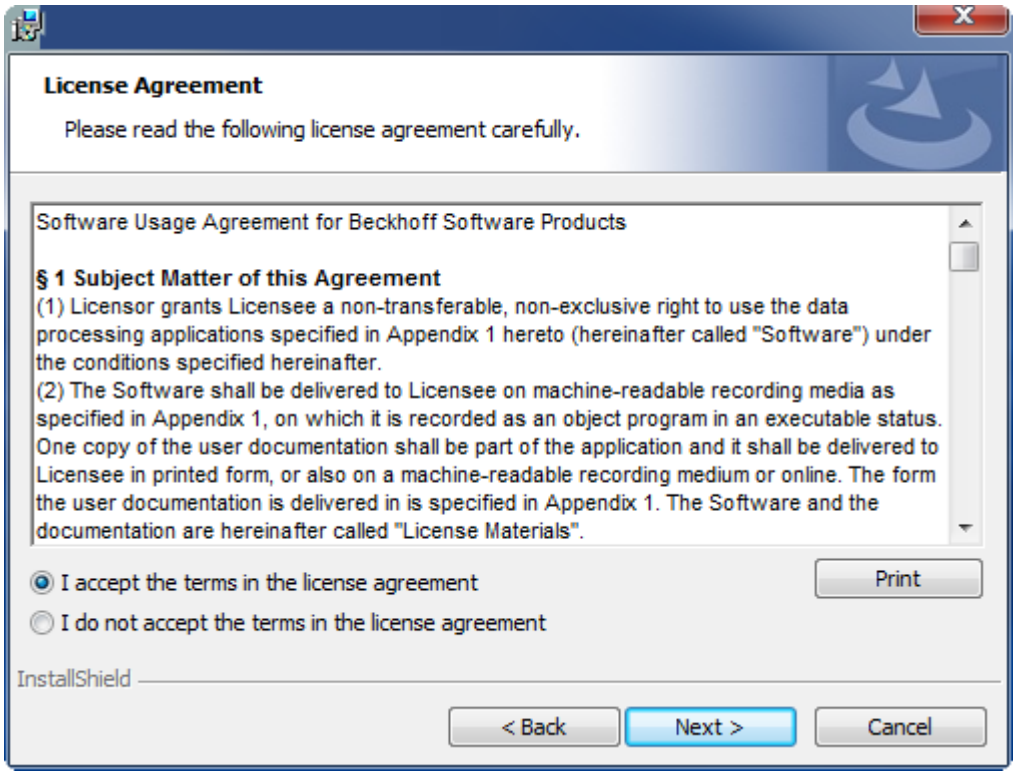
The following section describes how to install the TwinCAT 3 Function for Windows-based operating systems.

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

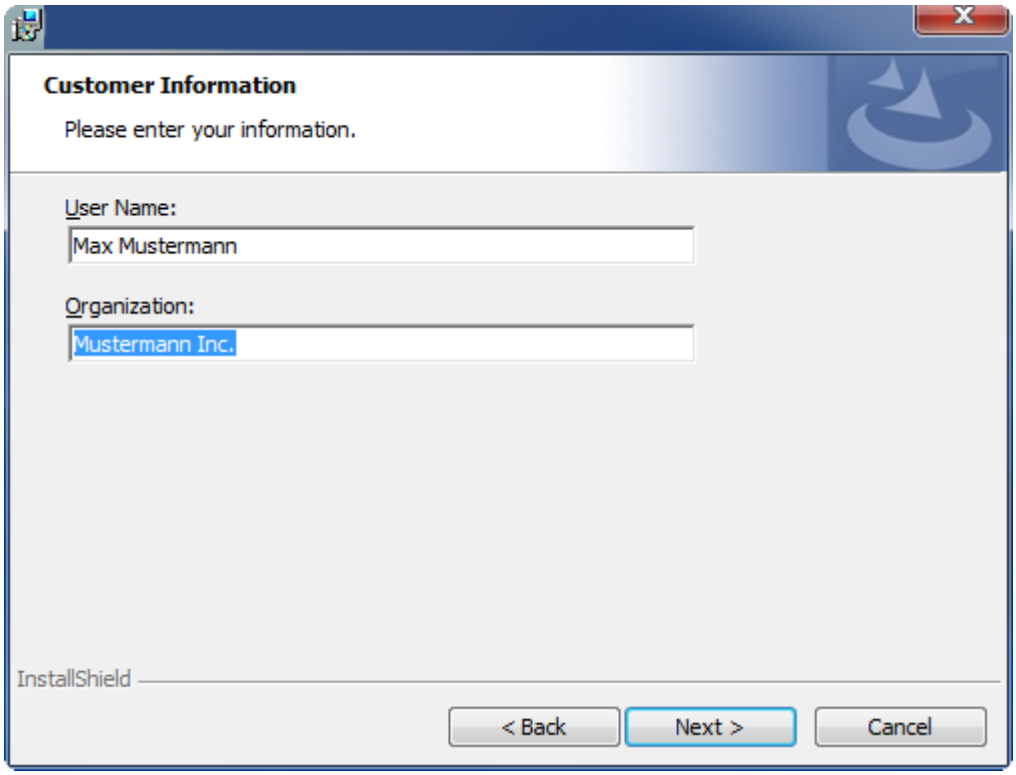
1. Run the setup file as administrator. To do this, select the command **Run as administrator** in the context menu of the file.

   ⇨ The installation dialog opens.

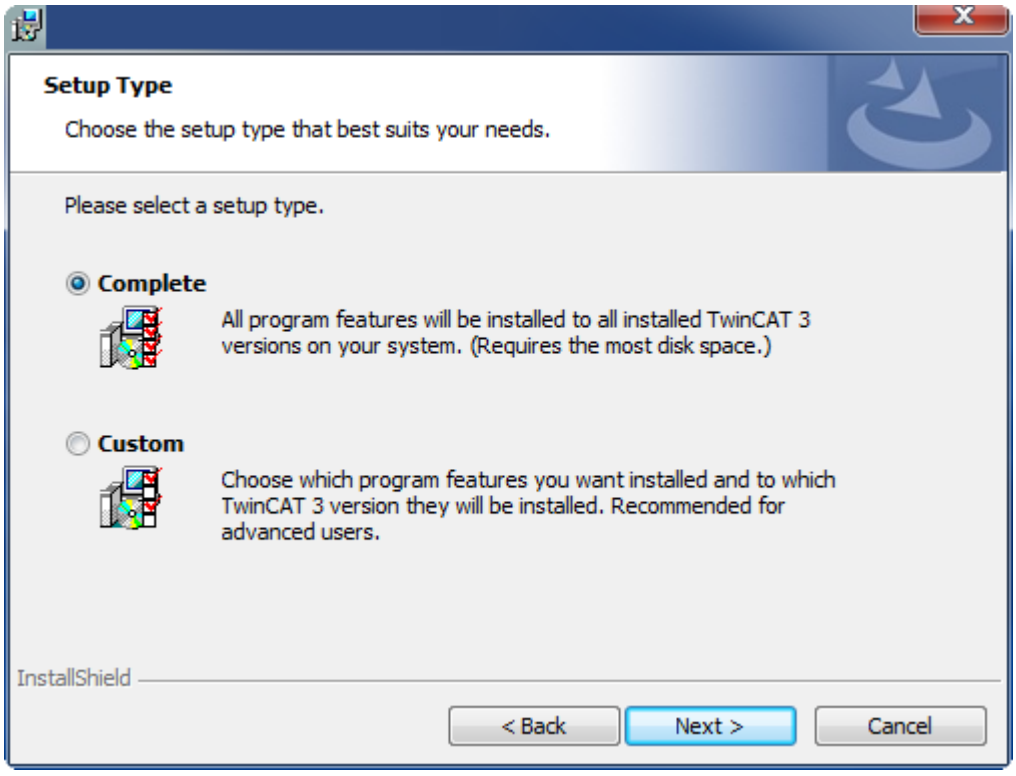
2. Accept the end user licensing agreement and click **Next**.



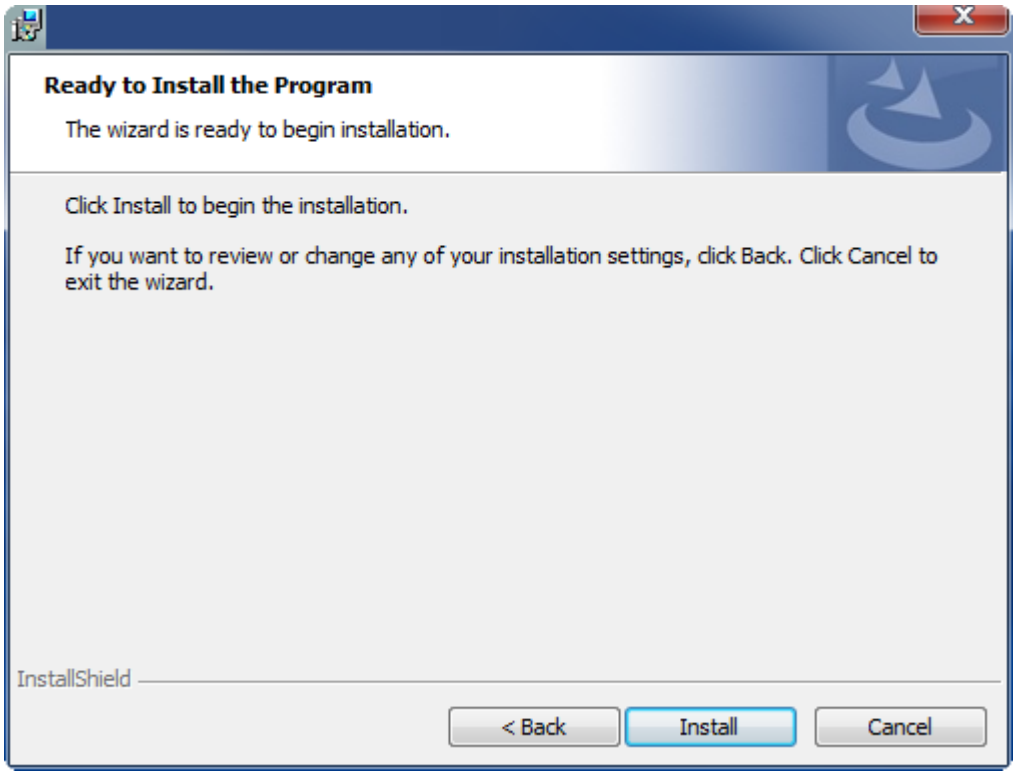
3. Enter your user data.



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

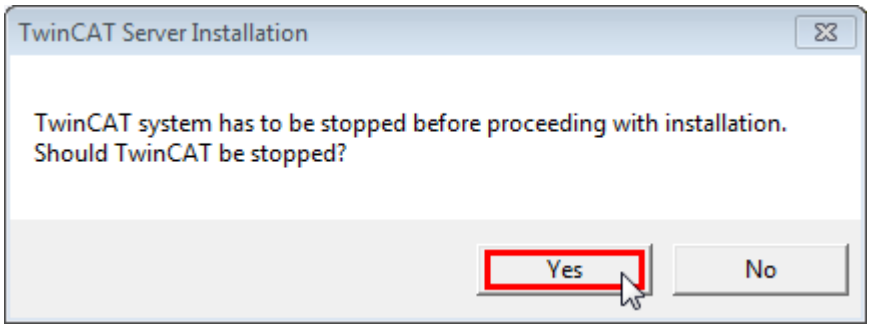


5. Select **Next**, then **Install** to start the installation.

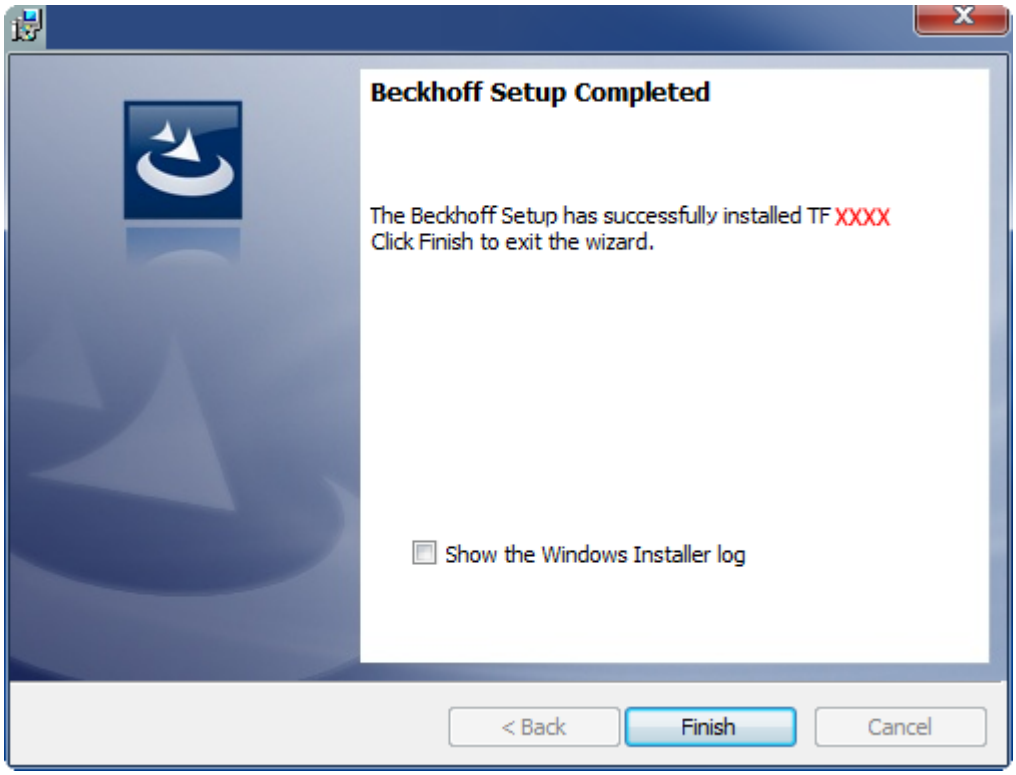


⇨ A dialog box informs you that the TwinCAT system must be stopped to proceed with the installation.

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



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



⇨ The TwinCAT 3 Function has been successfully installed.

3.3 Licensing

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

Licensing the full version of a TwinCAT 3 Function

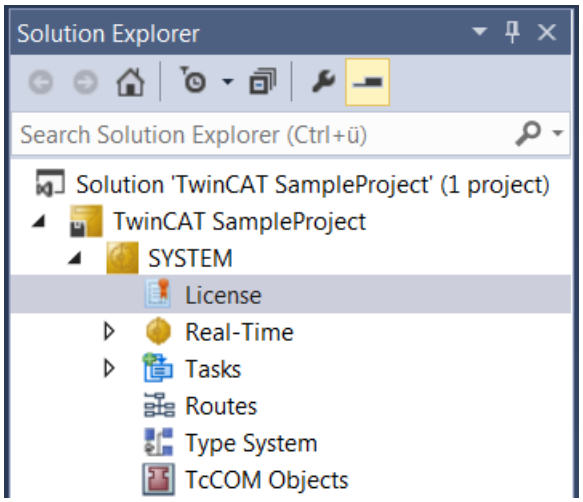
A description of the procedure to license a full version can be found in the Beckhoff Information System in the documentation "TwinCAT 3 Licensing".

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

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

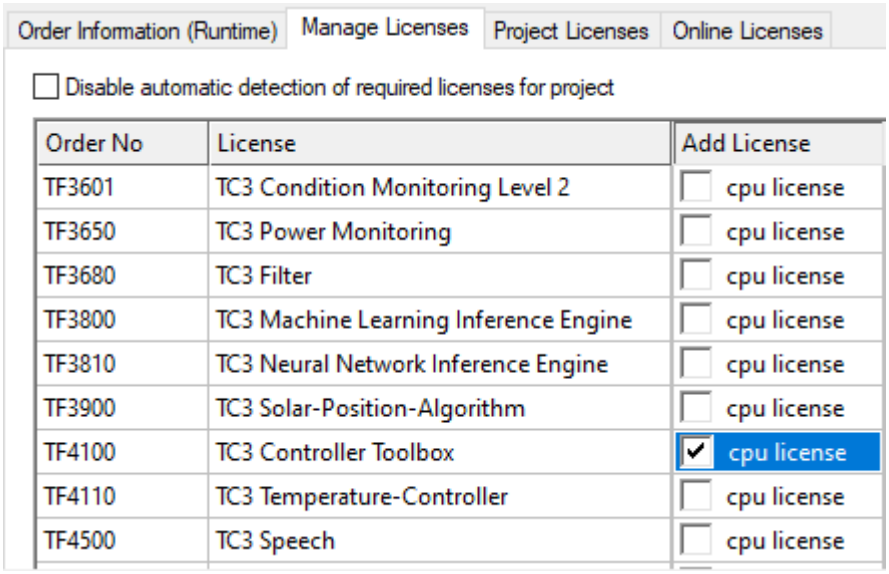
1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the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

4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.



⇨ The TwinCAT 3 license manager opens.

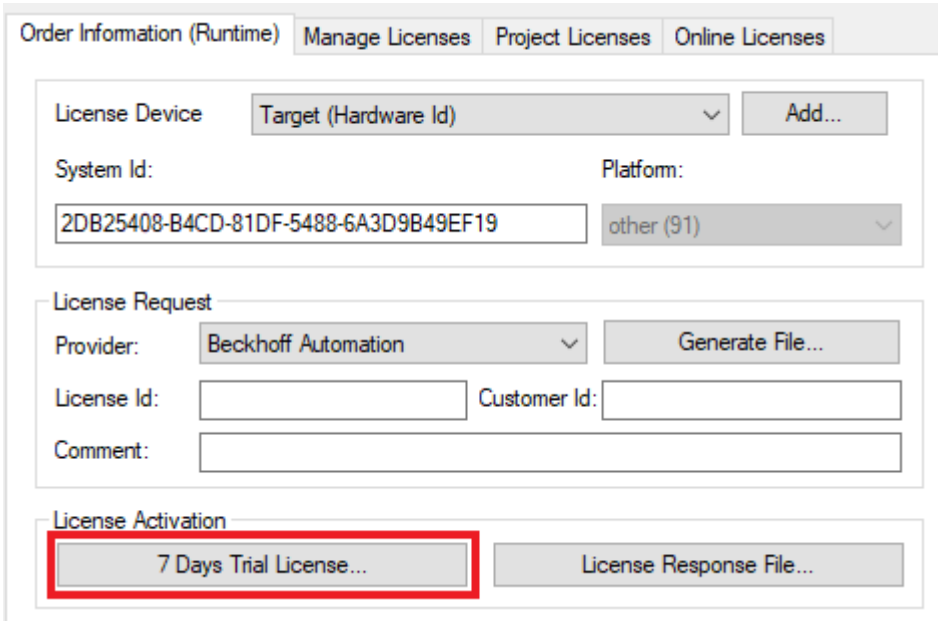
5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. "TF4100 TC3 Controller Toolbox").



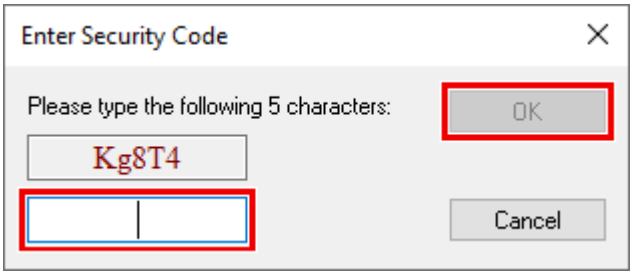
6. Open the **Order Information (Runtime)** tab.

⇨ In the tabular overview of licenses, the previously selected license is displayed with the status “missing”.

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



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



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

⇨ The 7-day trial version is enabled.

4 Analytics Workflow - First Steps

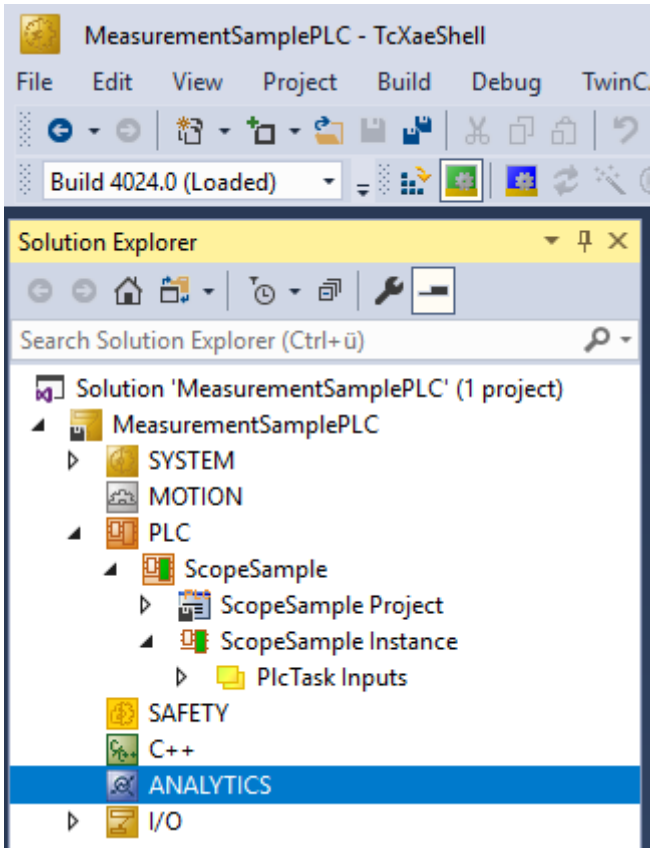
This step by step documentation presents the complete TwinCAT Analytics workflow. From the data acquisition over the communication and historizing up to the evaluation and analysis of the data and to the presentation of the data in web-based dashboard.

4.1 Recording data from the machine

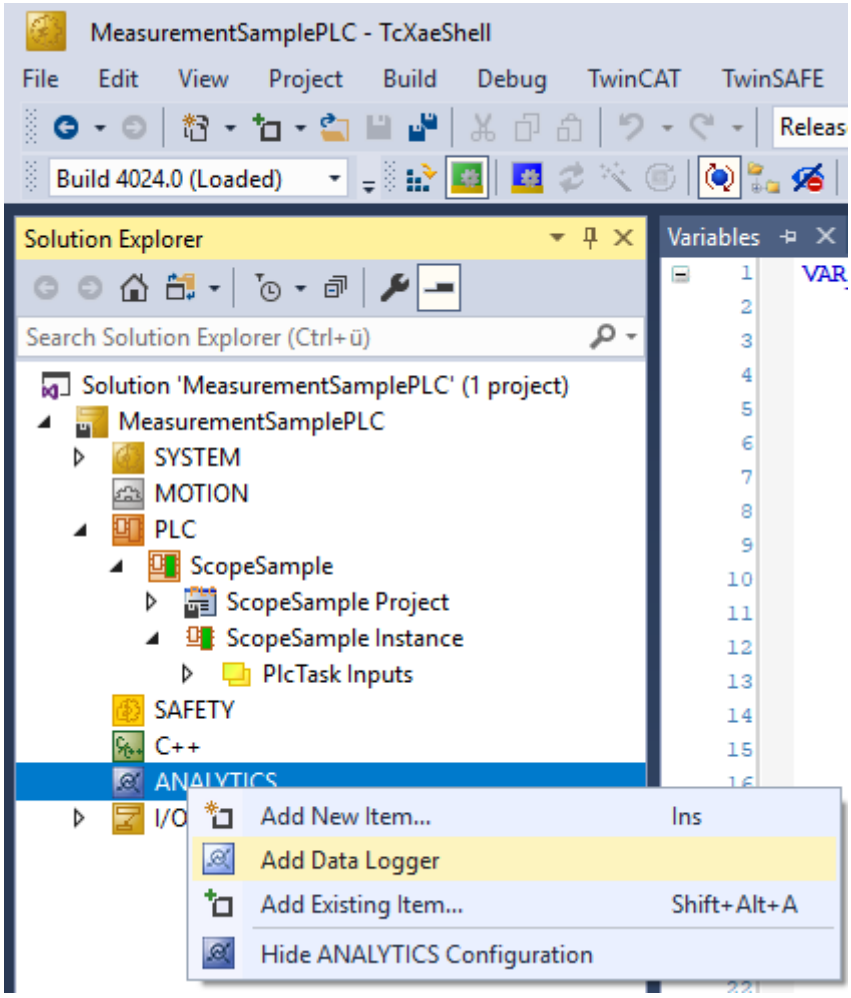
On the machine side is the Analytics Logger the recorder of process data from the machine image, PLC, NC and so on. The Logger is working in the real-time context of TwinCAT.

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

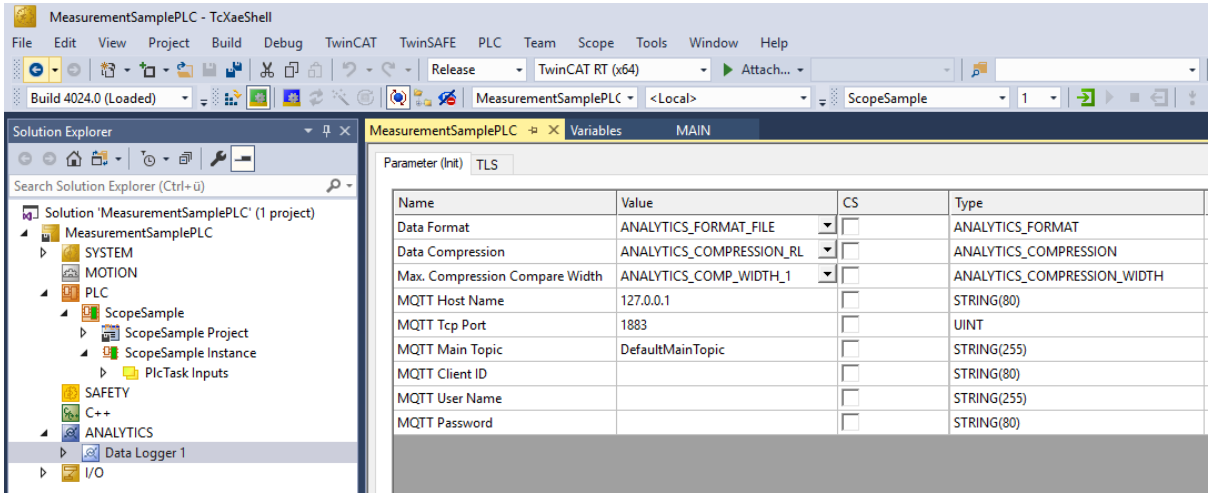
- To configure the Analytics Logger you have to navigate in your existing TwinCAT Project to the Analytics tree node



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



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



You can make your specific Analytics Logger settings

-Data Format: Binary file or MQTT stream

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

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

-Data Compression: on (default) or off

-Max Compression: mode of the compression

-MQTT host name

-MQTT Tcp port

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

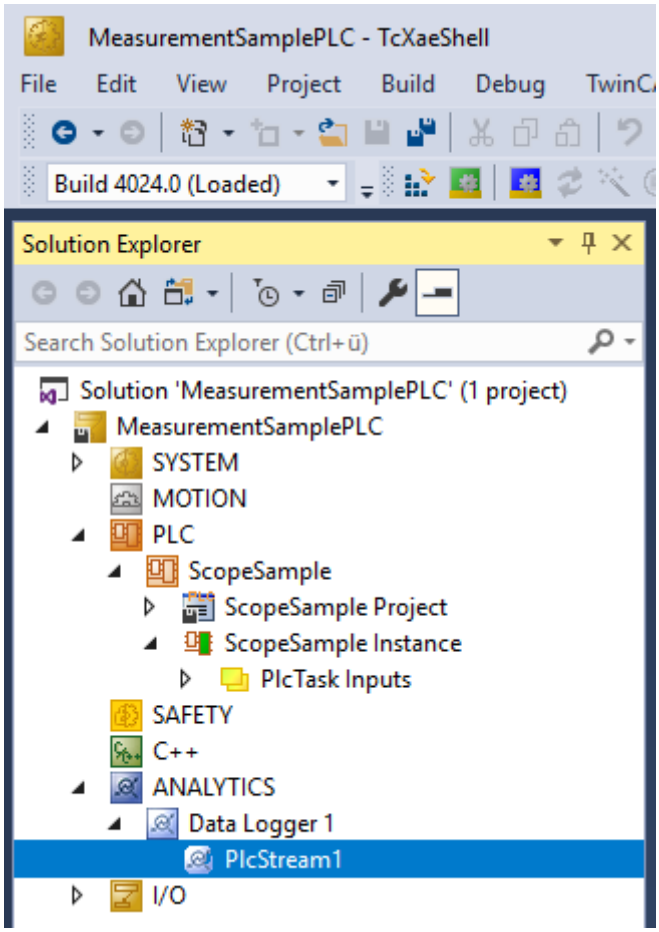
-MQTT Client ID should be unique in the network

-MQTT username

-MQTT password to make authentication at the message broker

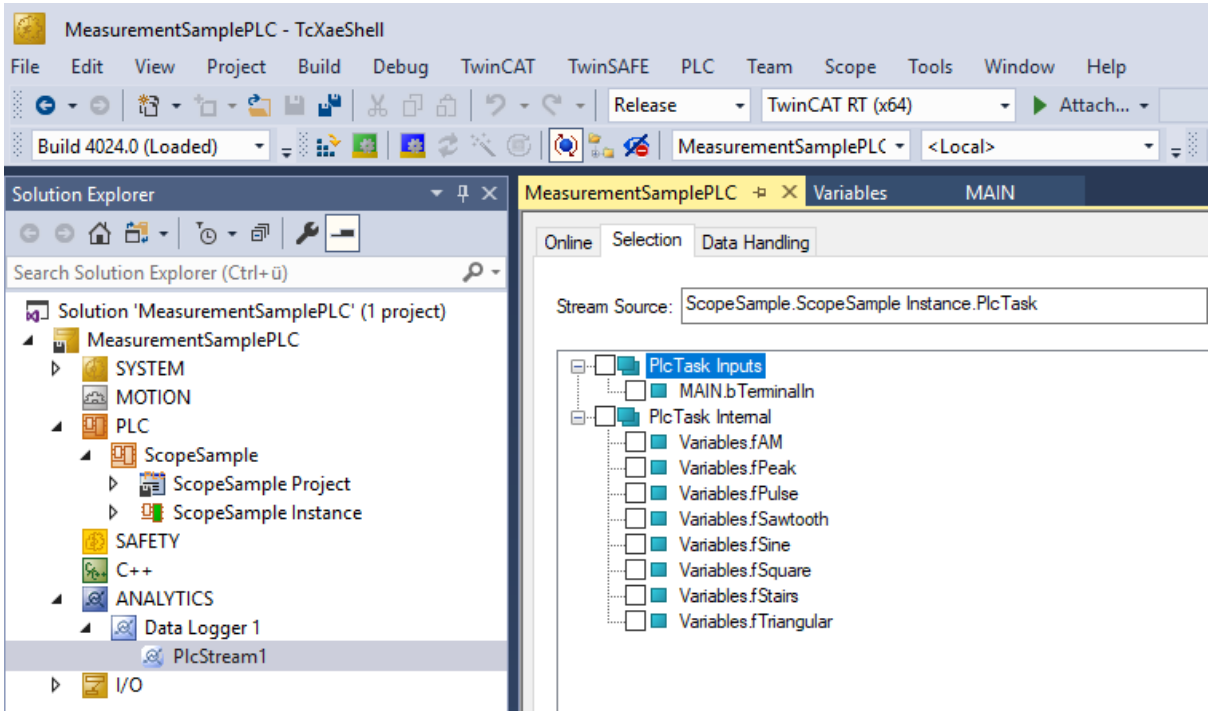
-At the TLS (Transport Layer Security) tab, security settings can be configured. TLS is a secure communication channel between client and server. By the usage of certificates, the TCP port 8883 is exclusively reserved for MQTT over TLS. Analytics Logger is supporting the modes CA Certificates, CA Certificates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

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

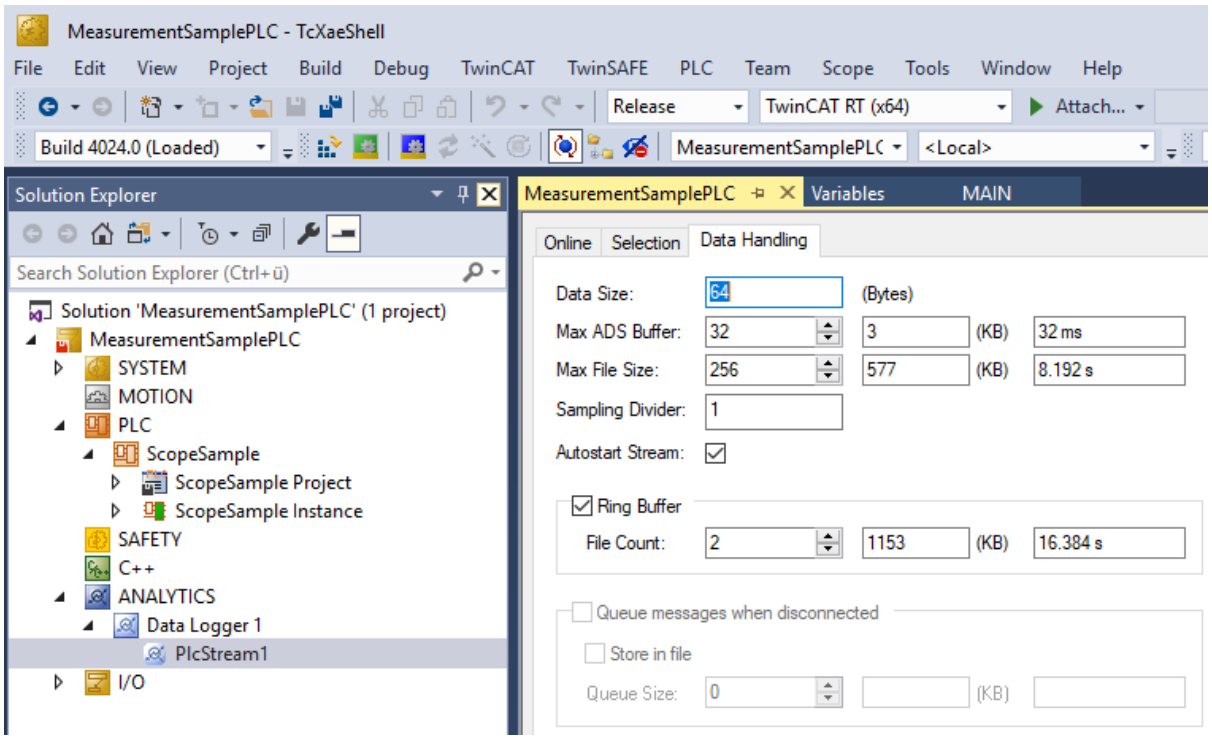


An additional device stream will be shown if your configuration provides an EtherCAT Process Image.

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

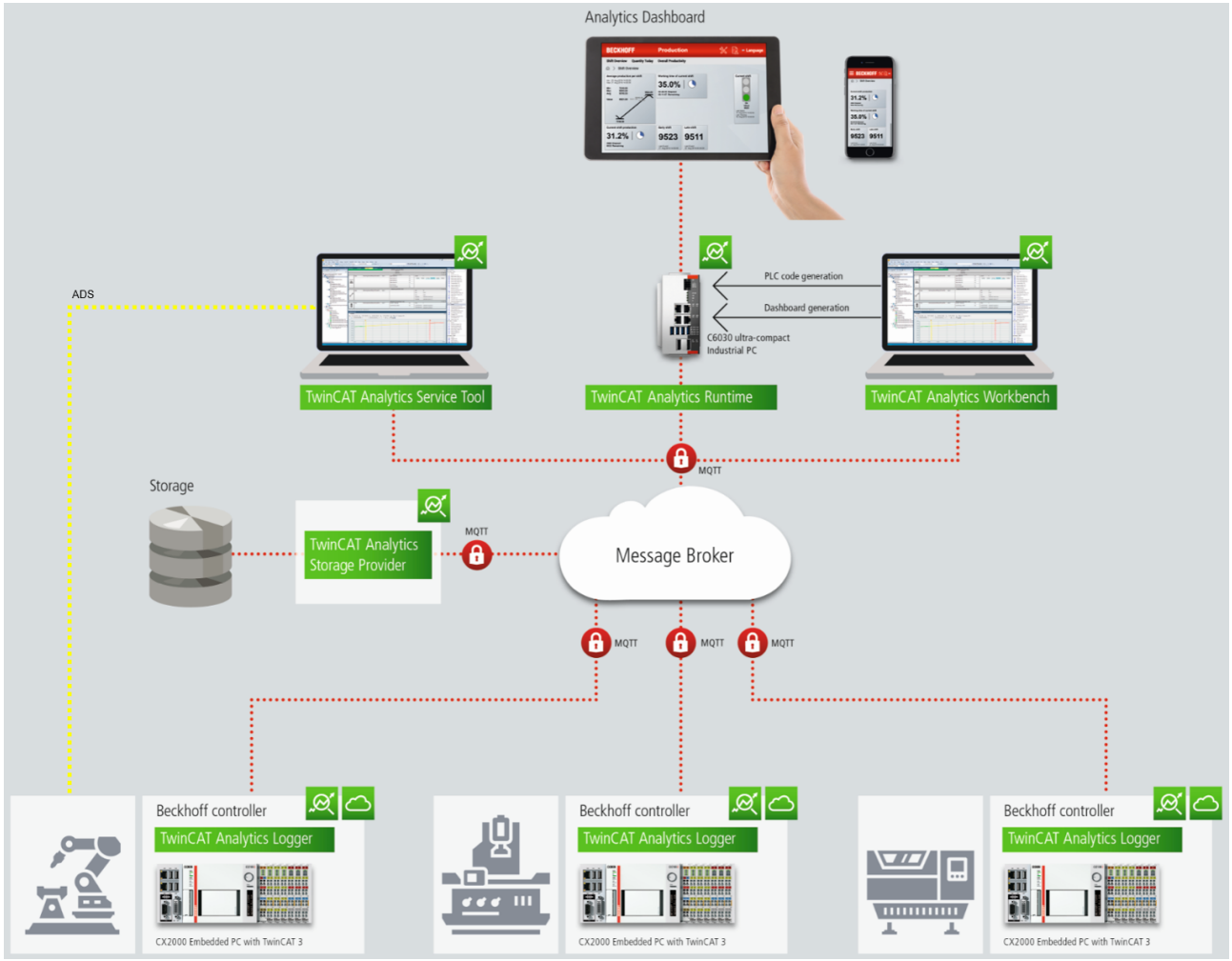


- Finally it is possible to change the package size for the frames or to configure the ring buffer for disconnects and file in the Data Handling tab.



4.2 Communication

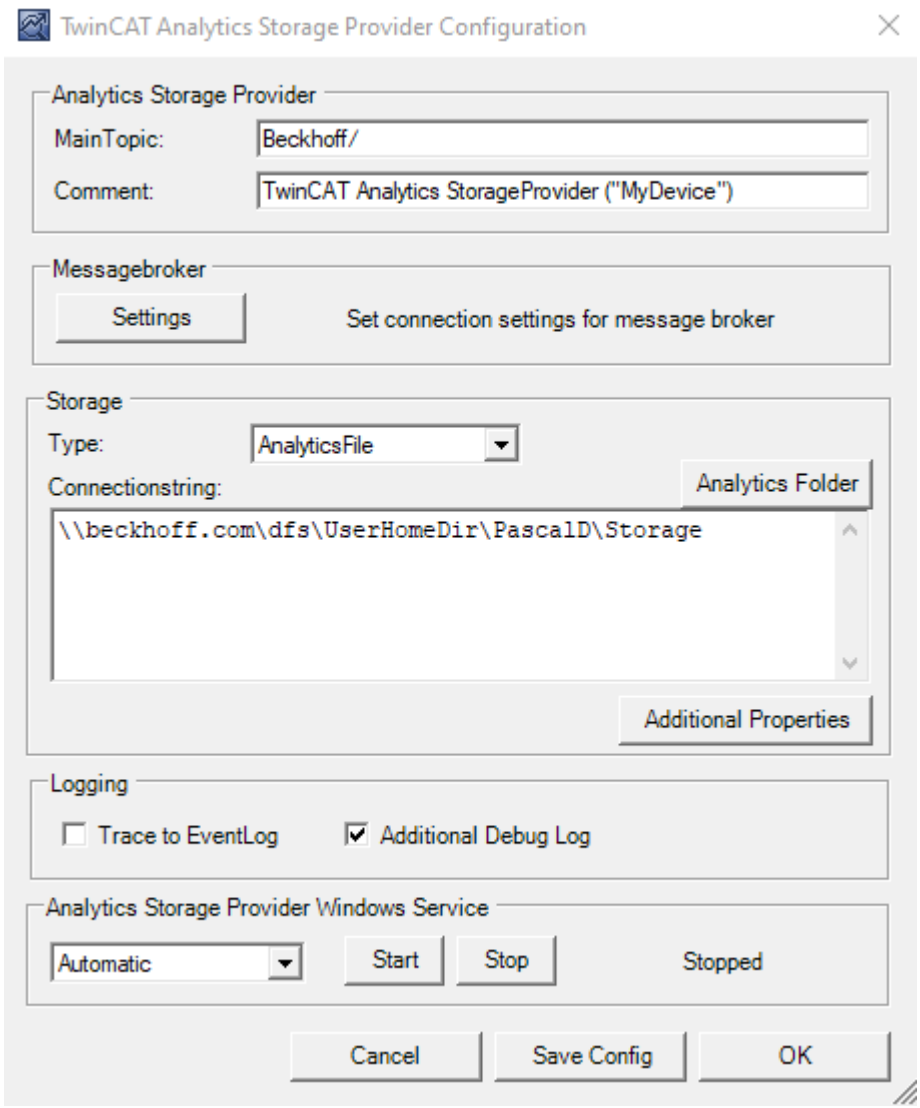
Currently, the Analytics workflow is fully mappable via MQTT. The engineering tools can also access the data of the machines via ADS and carry out analyzes.



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

4.3 Historicize data

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

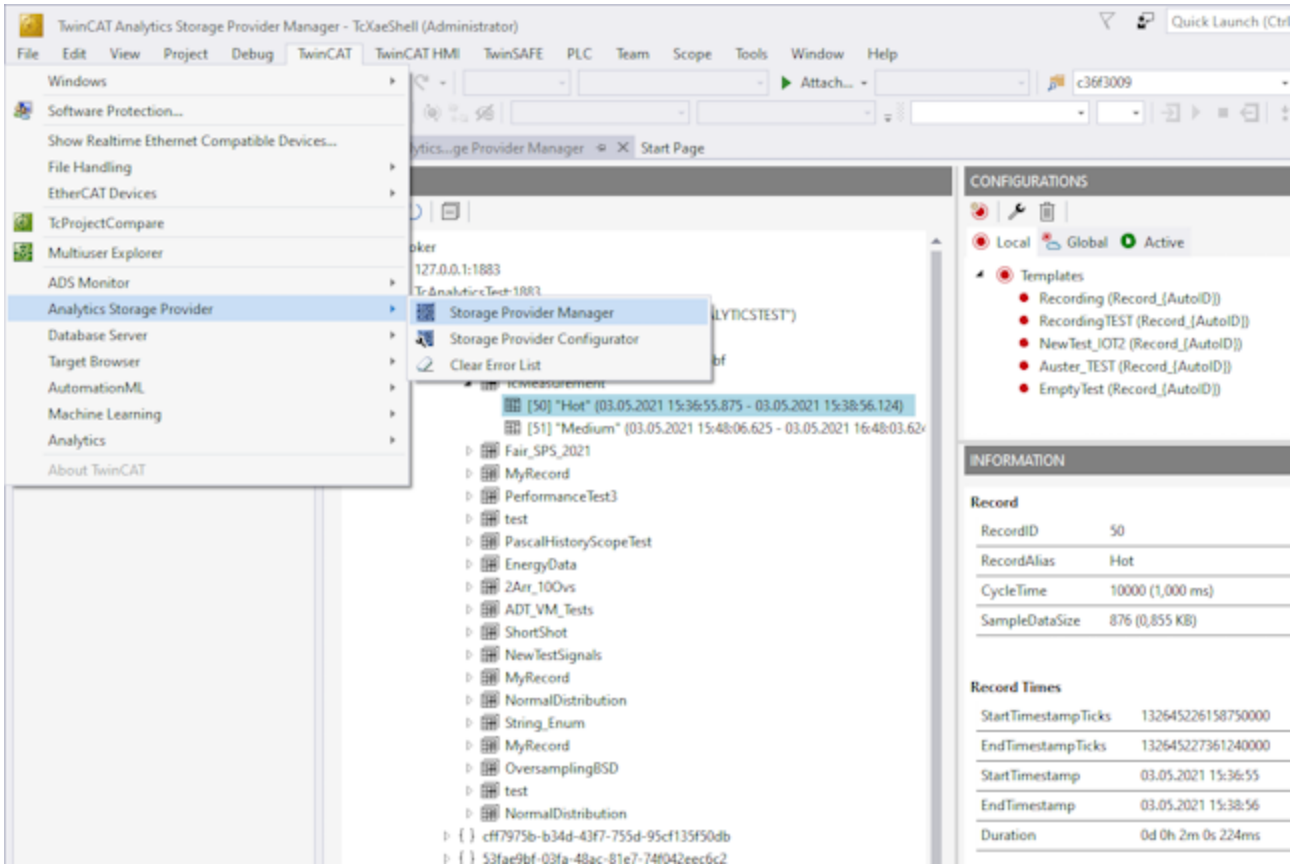


The main part of the topic can be defined in the configuration as well as the comment, which is used for identification if more than one Storage Provider is registered with the message broker.

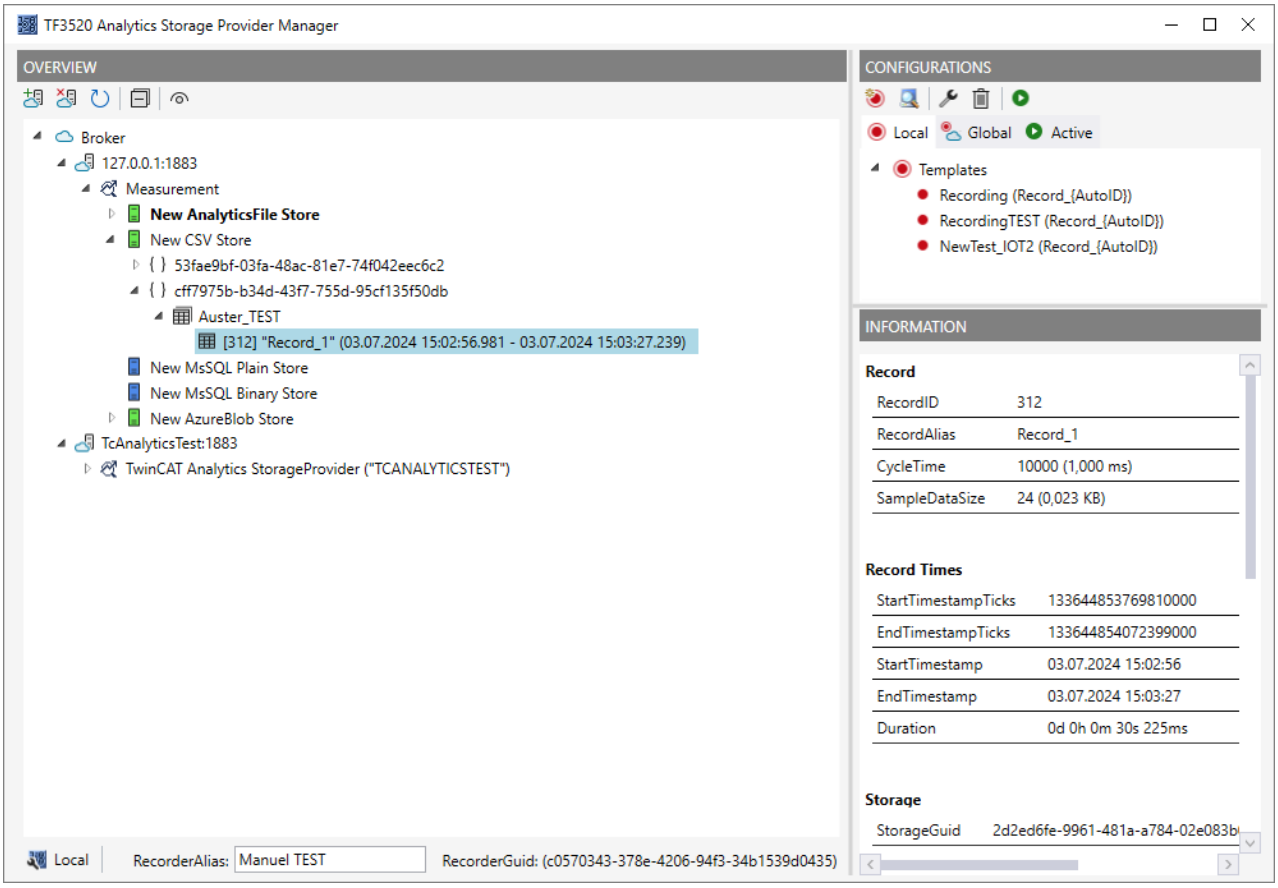
You can make the message broker settings and decide on a storage type:

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

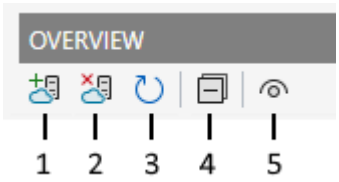
At last you can save the configuration and start the service. The next step is to configure the specific recording. For this you should select the **Storage Provider Manager** in your development environment.



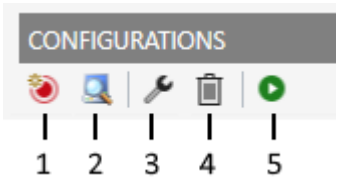
With the Storage Provider Recorder recording definitions can be created, started and managed. In addition, it is possible to manage the data memories of individual Analytics Storage Providers. All important properties of the found Analytics Storage Providers and historized data are clearly displayed.



Toolbars



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

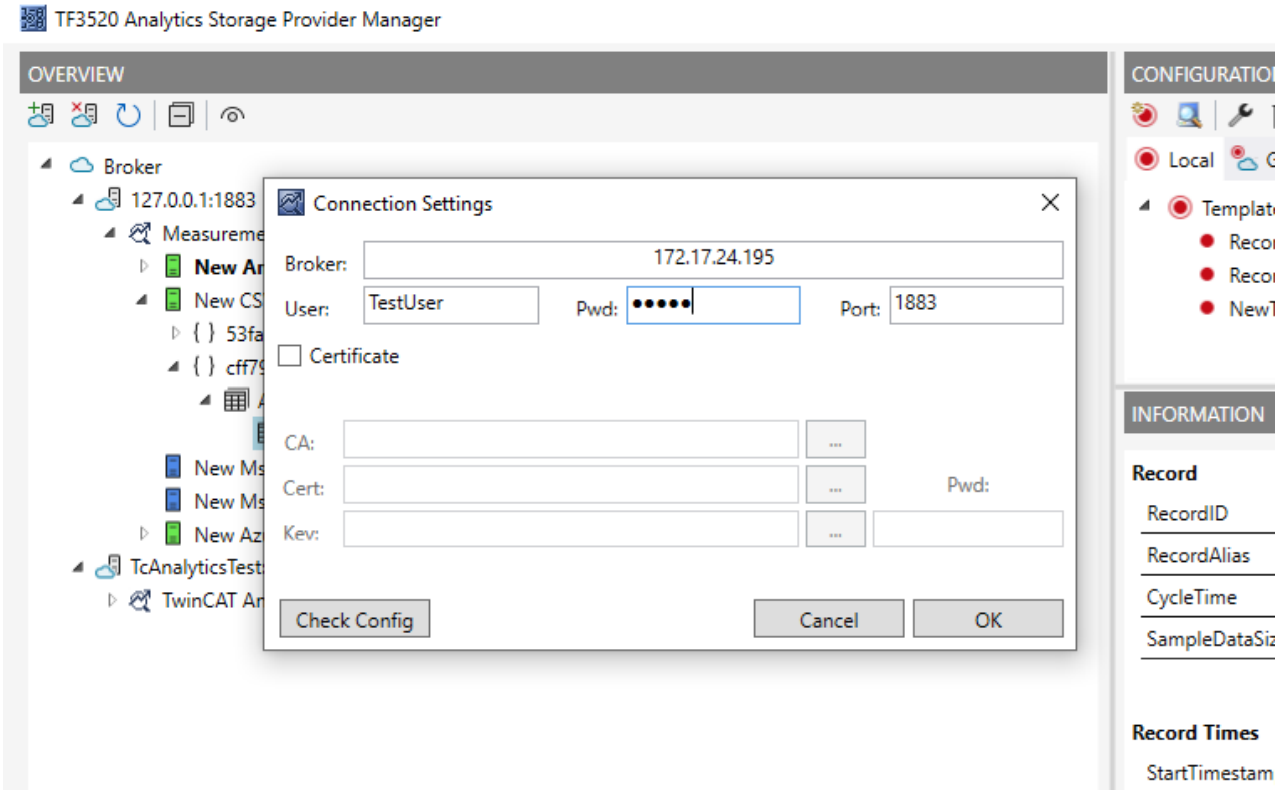


1	Create new recording definition
2	Open Target Browser
3	Edit selected definition
4	Delete selected definition
5	Start selected definition

Set up Manager window ("OVERVIEW")

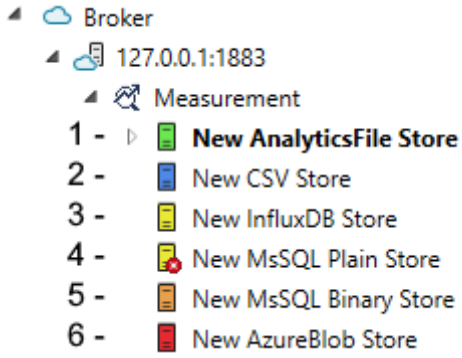
First assign a "RecorderAlias". This helps to group the started recordings and to find its self started ones again.

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



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

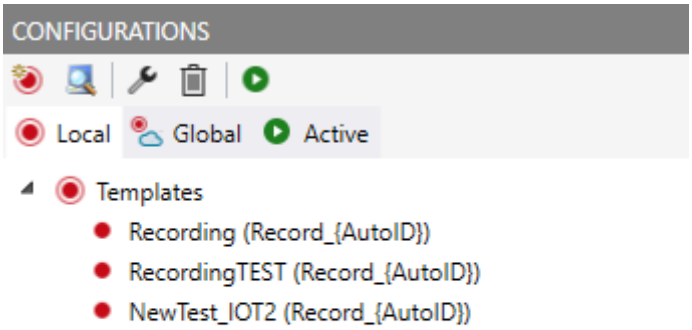
"Storage" status



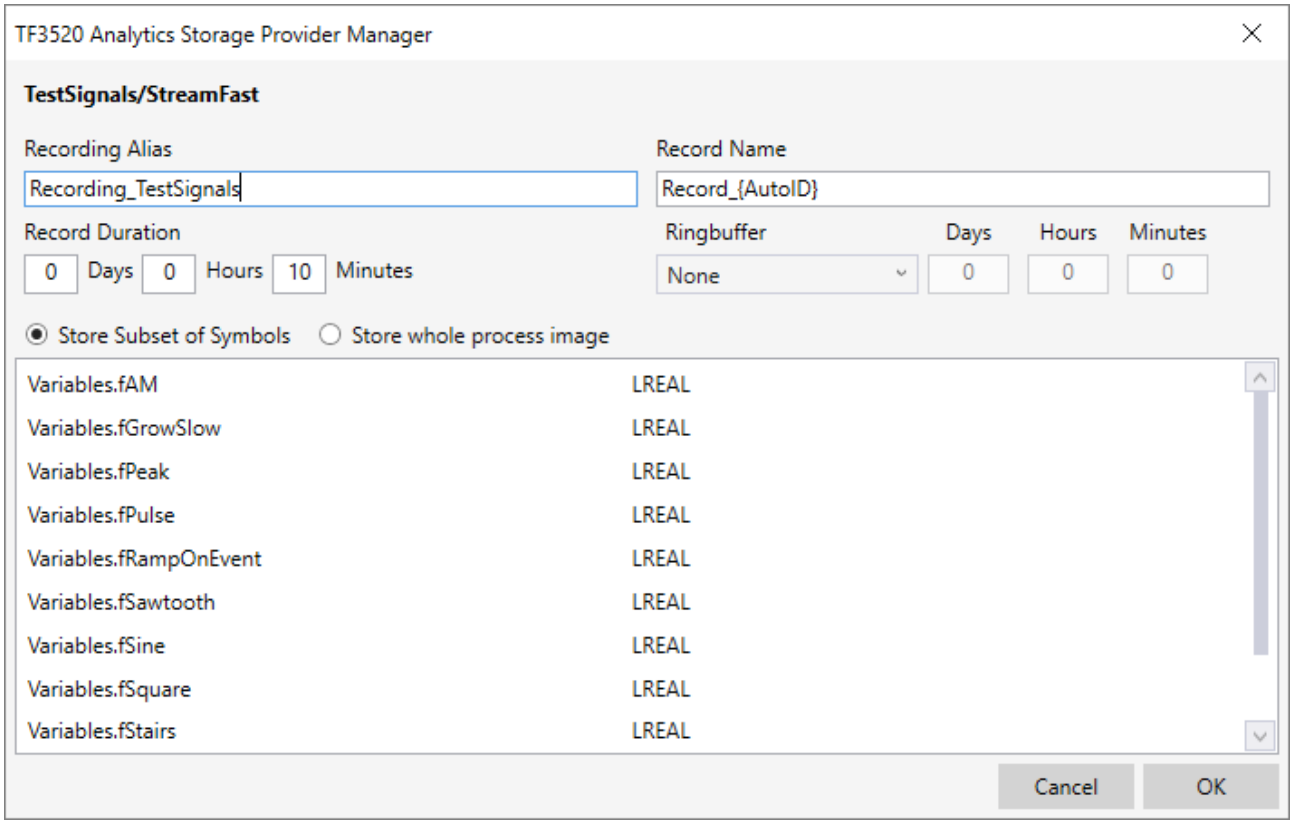
1	Storage Online
2	Storage Offline
3	Storage starts
4	Storage starts with error. Still trying to start it
5	Storage is shut down
6	Storage is in the error state

Recording definitions ("CONFIGURATIONS")

Local:



Recording definitions that are saved locally are listed in the "Local" tab. New definitions can also be created here. To do this, press the "Create new definition" button. The following dialog opens.



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

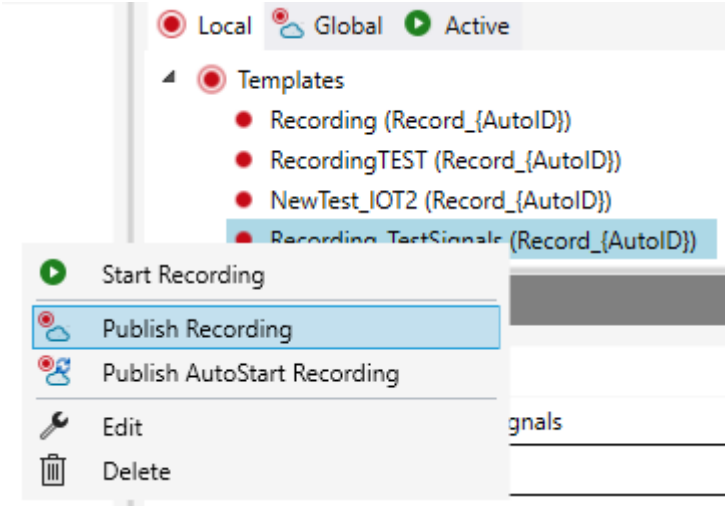
Various placeholders are available for the Record Name:

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

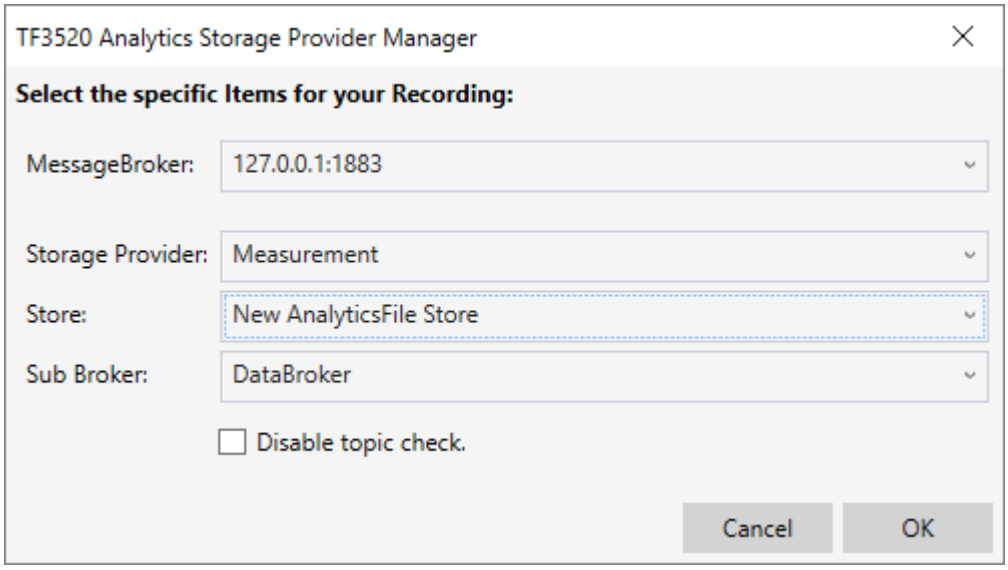
You can also configure recording names and a duration (otherwise the recording will run endlessly until it is stopped manually). A ring buffer can be set by storage or time.

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

It is now possible to start this definition directly via the toolbar or the context menu. However, it is also possible to make the definition globally accessible. This can be done via the context menu with the entry "Publish Recording".

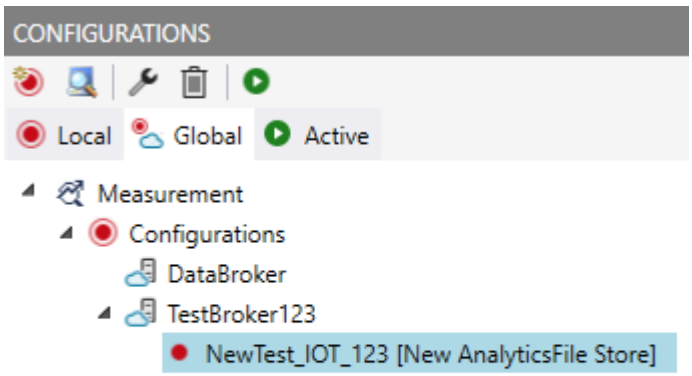


The following dialog then opens:



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

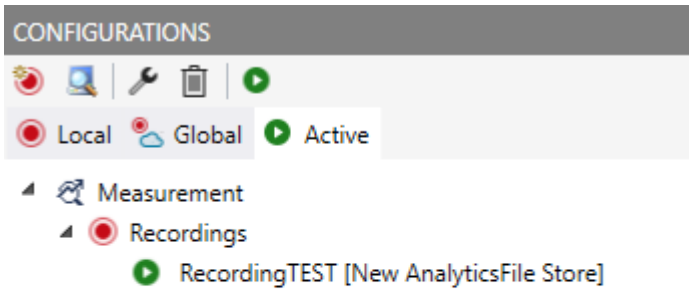
Global:



All global recording definitions of the selected Analytics Storage Provider are listed here. The definitions are assigned to the Sub Brokers. These global definitions can be changed, downloaded and started.

Started recording definitions are listed in the "Active" tab.

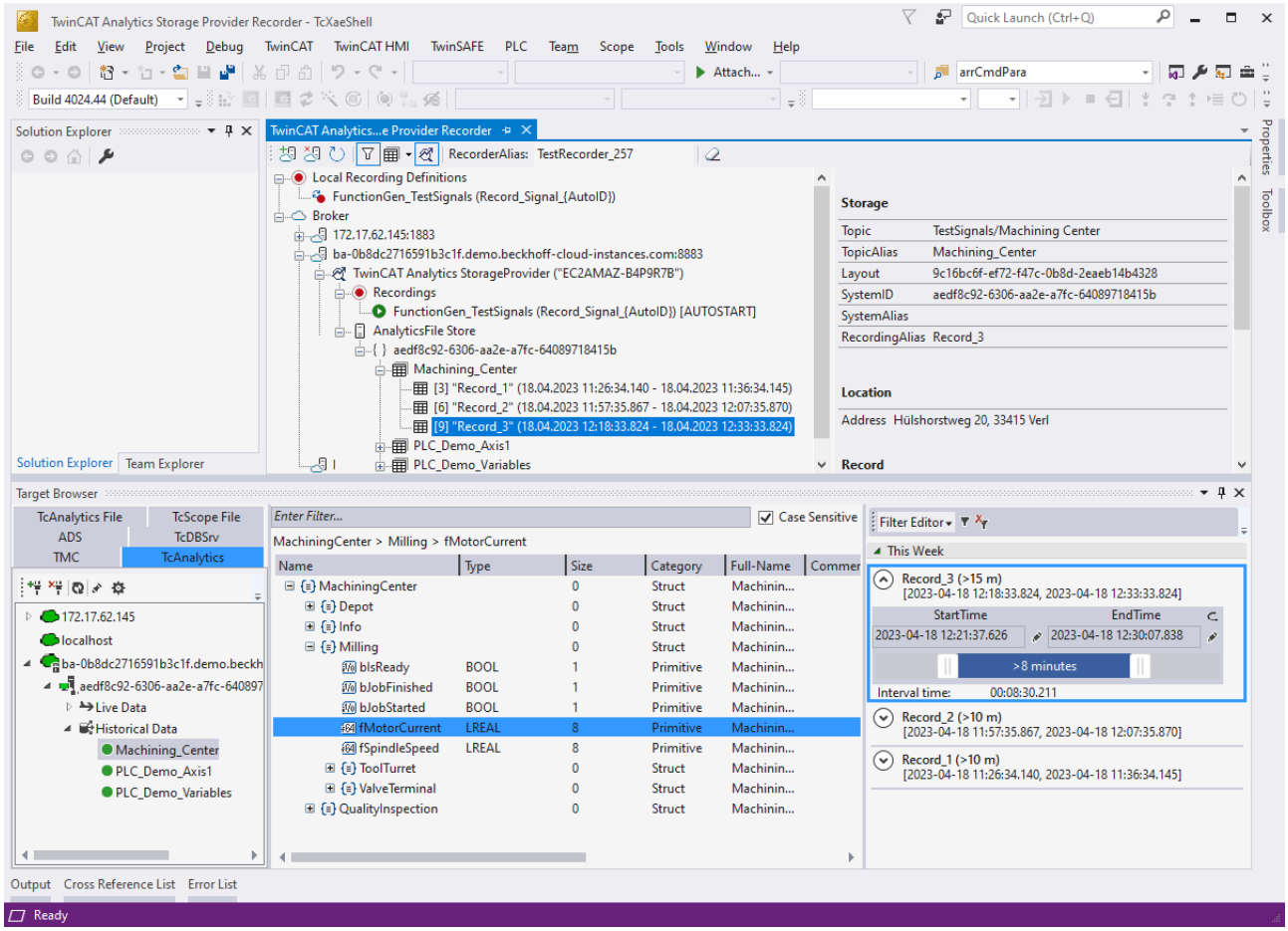
Active:



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

Use historized data

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

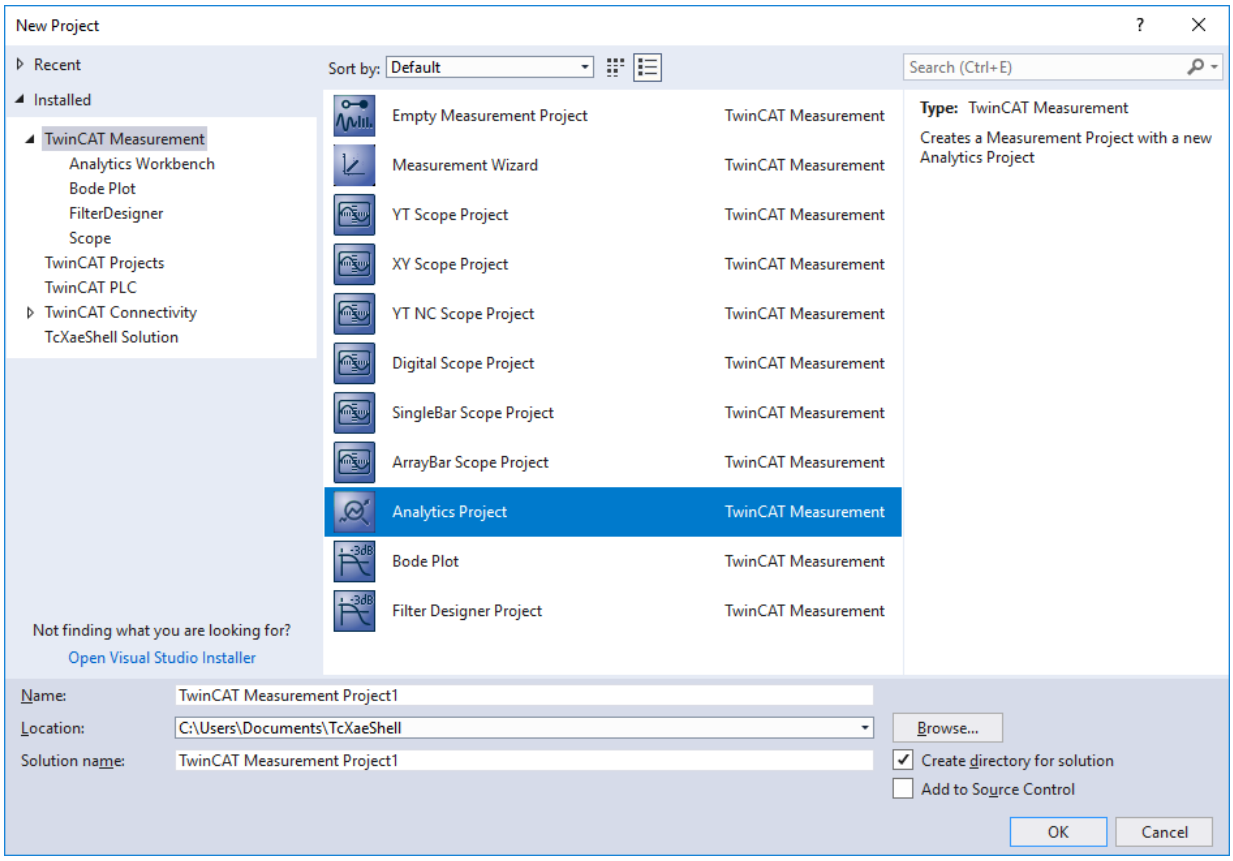


4.4 Analyse data

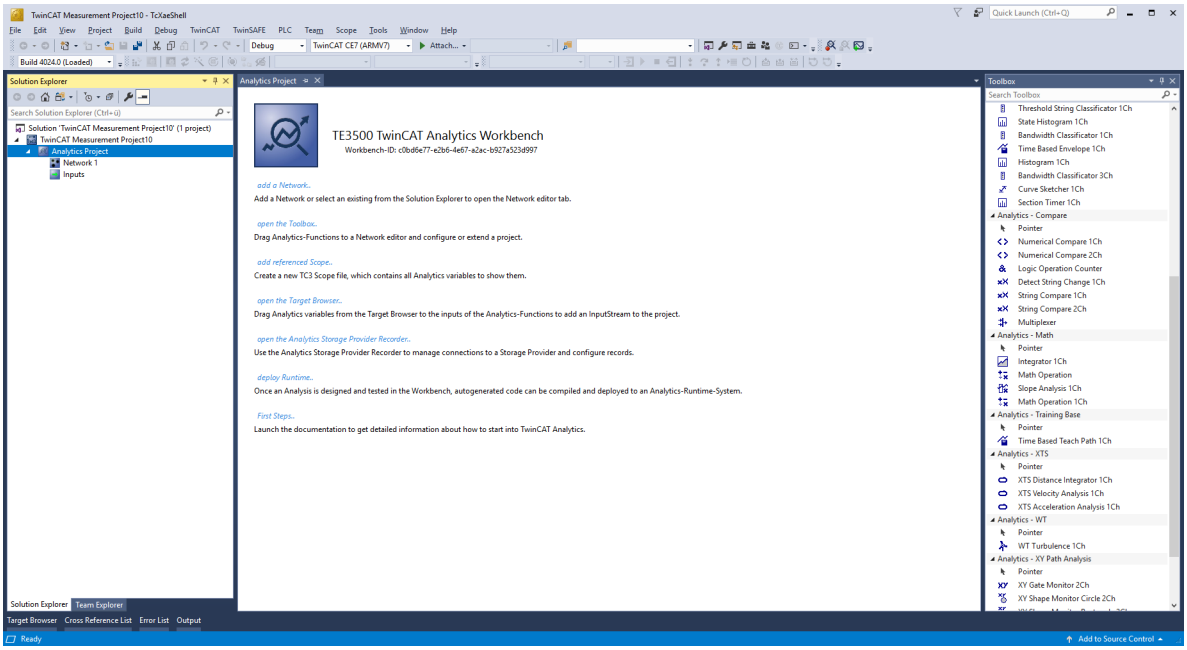
✓ Open your TwinCAT Engineering environment to start the data analysis.

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

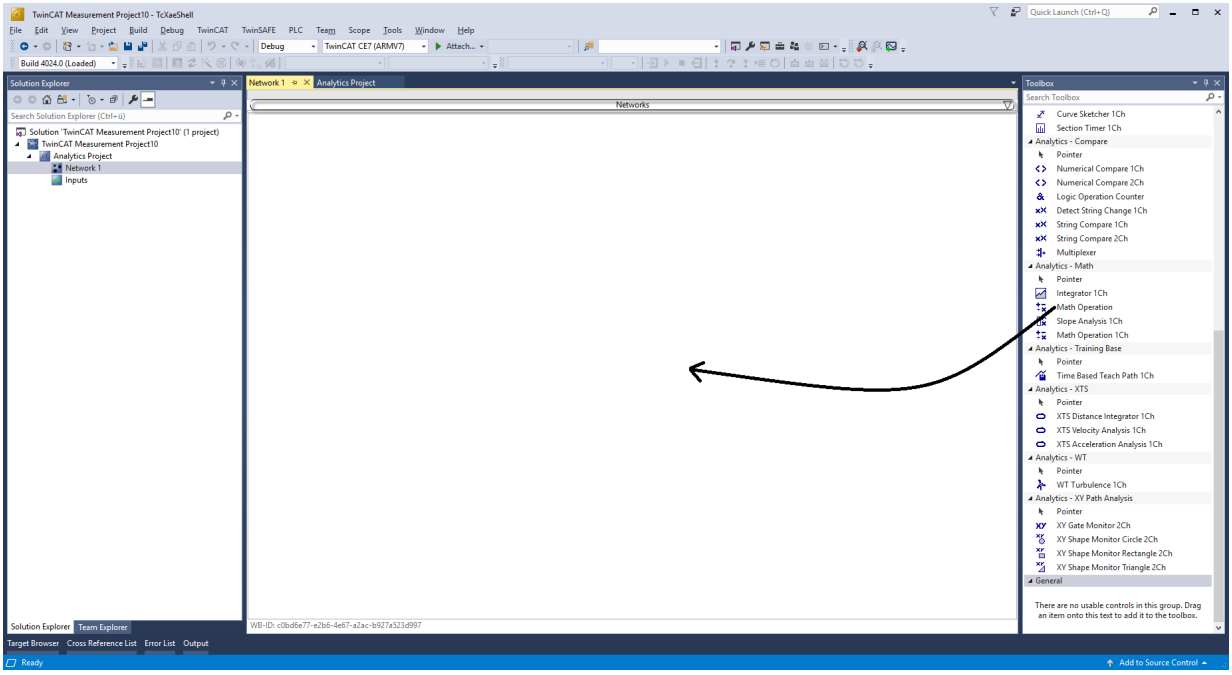
2. Select the **Analytics project template** from **TwinCAT Measurement**.



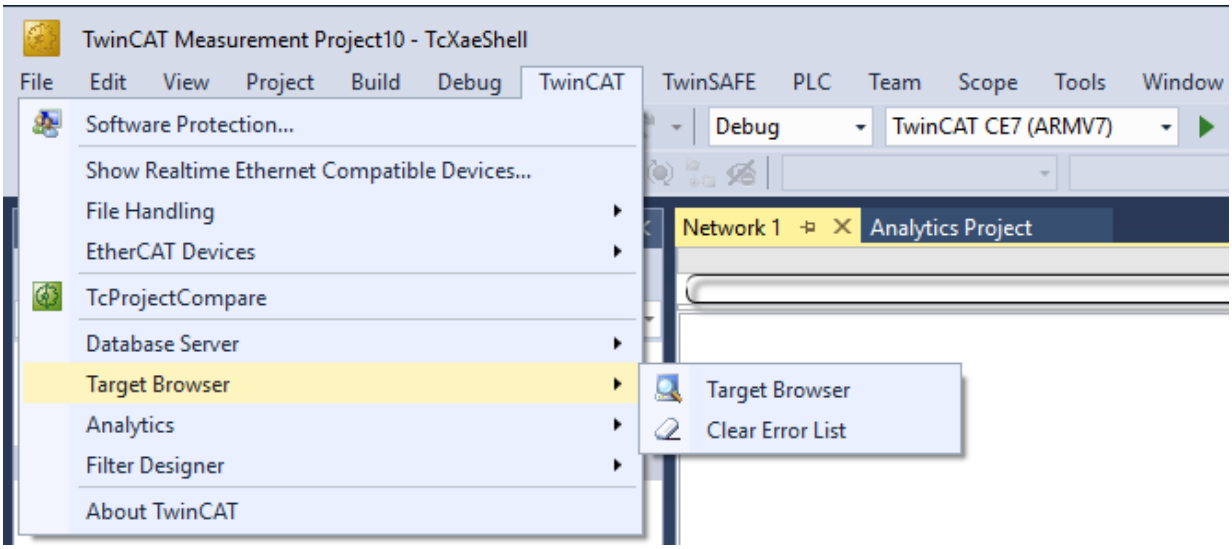
⇨ The new project is displayed in the Solution Explorer. After clicking the **Analytics Project** tree node element a start window opens where you can select your first action. From here you can add a network, open the **Toolbox**, open the **Target Browser** or open the **Analytics Storage Provider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



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

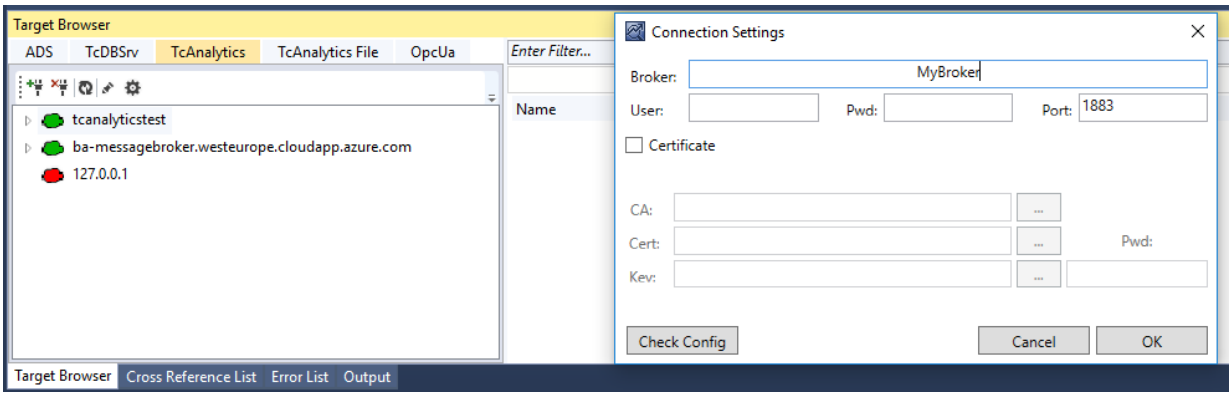


4. When you click on the network, an editor opens. Now you can drag and drop the desired algorithm into the editor interface.
5. After selecting the algorithm, you need to connect input variables to the modules (algorithm). To do this, open the **Target Browser**.
   TwinCAT > Target Browser > Target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

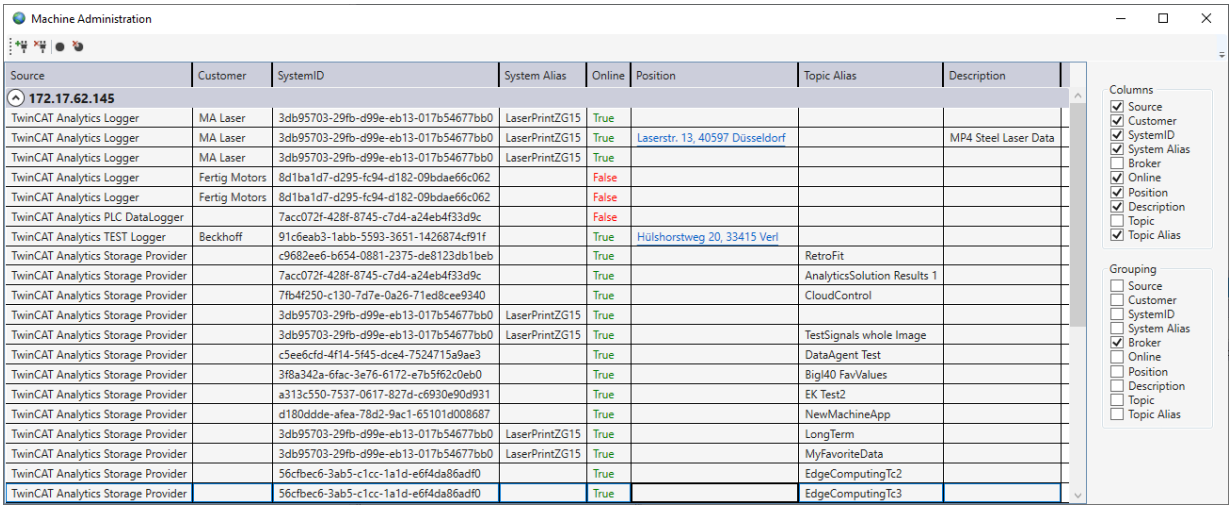


6. Now select the **TcAnalytics** or **TcAnalyticsFile** tab in the Target Browser. Continue with the tab **TcAnalytics** (MQTT).

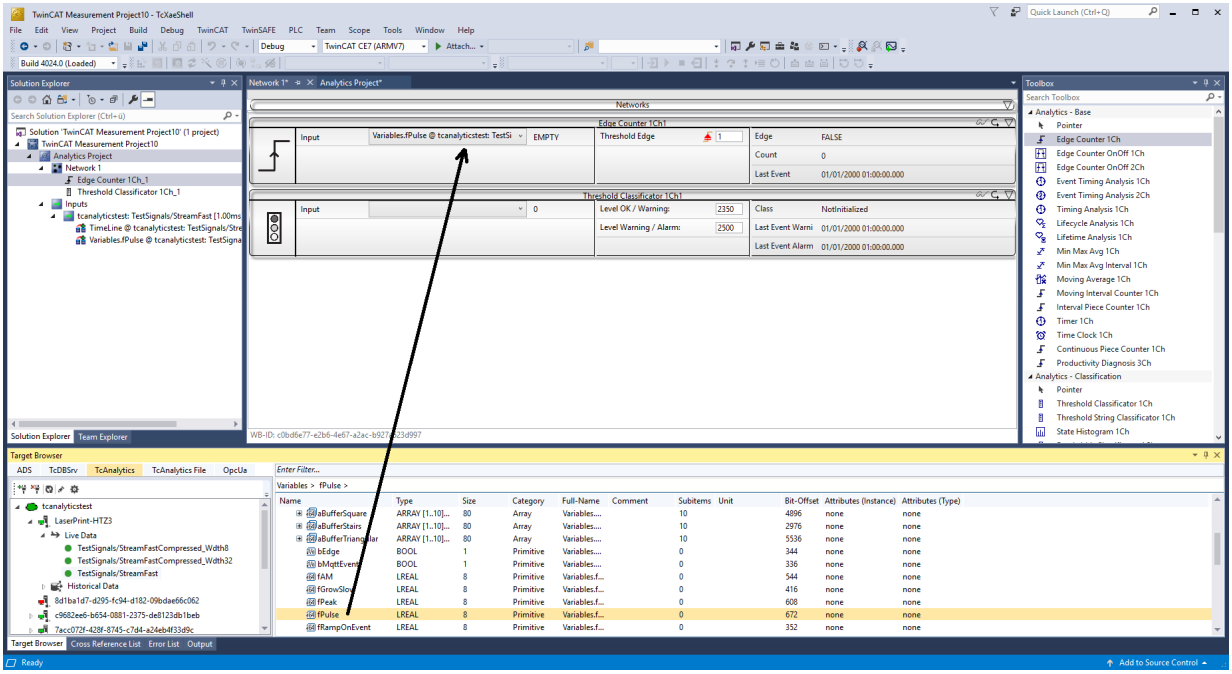
7. Click the icon highlighted in green in the toolbar of this Analytics extension. A window opens in which you can specify the connectivity data of your message broker.



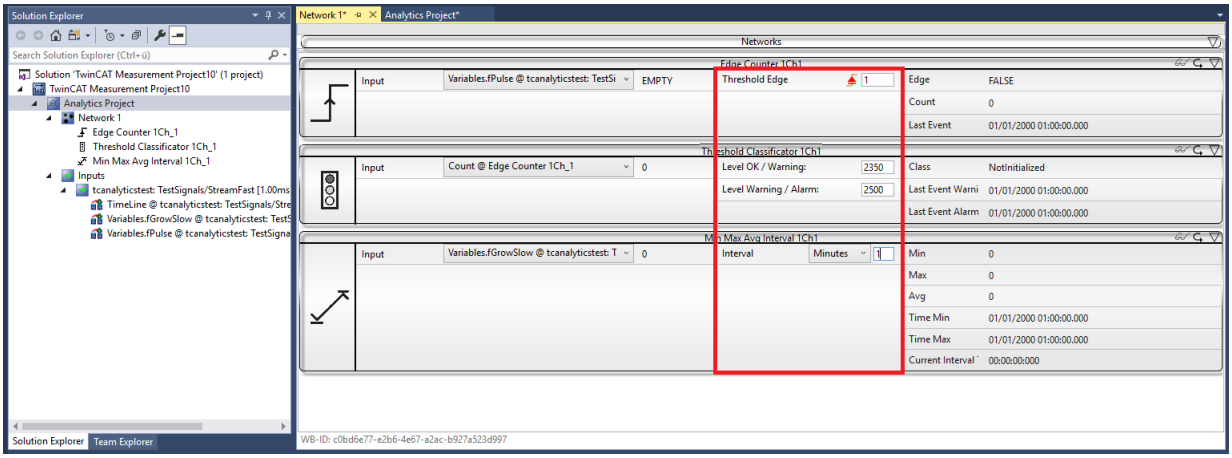
8. Select your MQTT Analytics client (TwinCAT Analytics Logger, TwinCAT IoT Data Agent or Beckhoff EK9160). There is a unique ID for each control. This ID is displayed in the Target Browser.
9. Clicking on the **gear icon**, you will get to the Machine Administration page. Here you can assign a system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



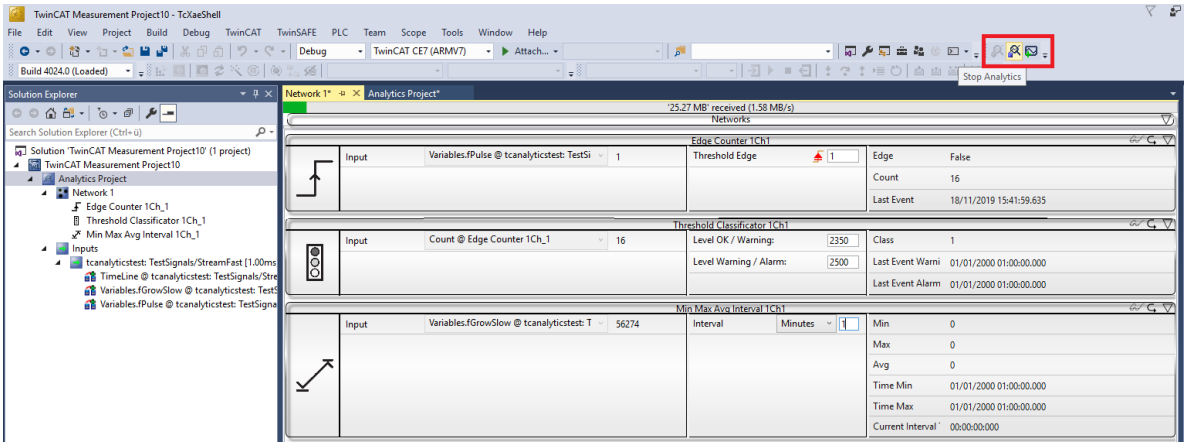
10. In the next step, you can choose between live data and historical data for each MQTT Analytics client. In this case, the historical data is provided by the TwinCAT Analytics Storage Provider.



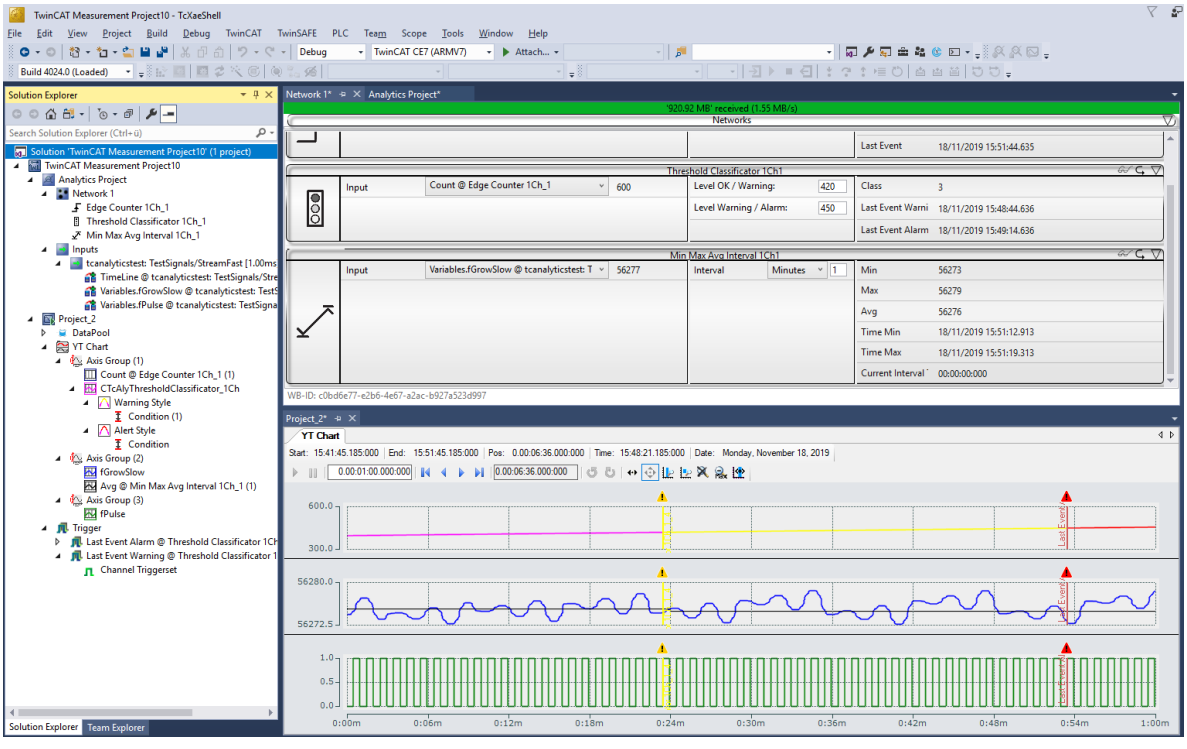
11. You can drag and drop the variables into the inputs of the specific algorithm. In most algorithms, conditions such as thresholds, time intervals, logical operators etc. can be specified. These settings are made in the middle of each module.



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



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

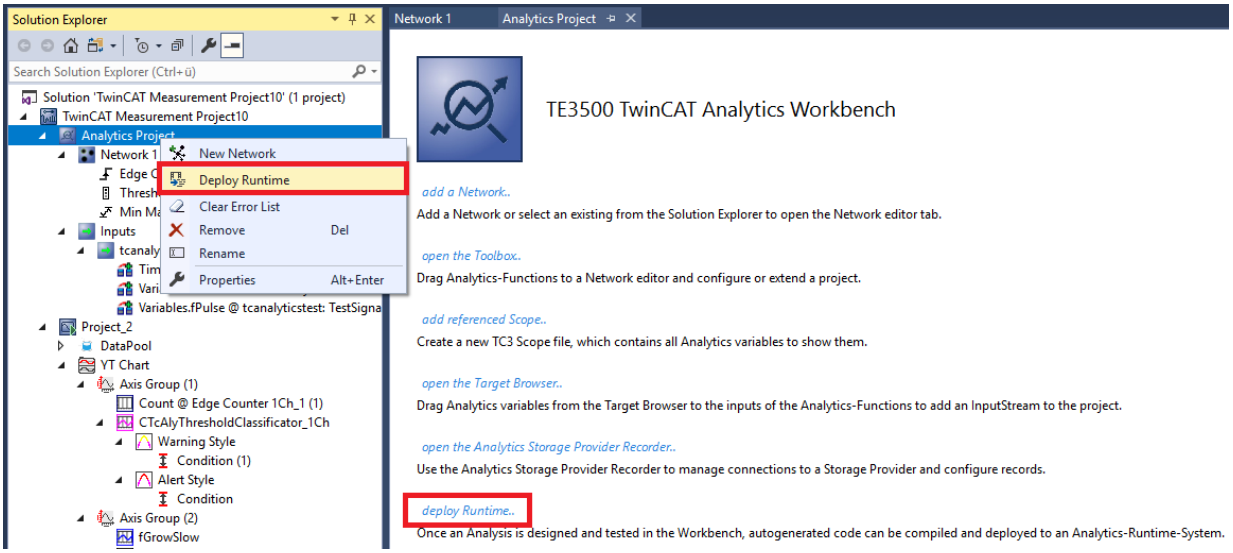


⇨ The analysis results can be displayed in the Scope View graphs using drag-and-drop. For example, a mean value can be displayed as a new channel in the view. Timestamps as markers on the X-axes show significant values.

4.5 24h Analytics application

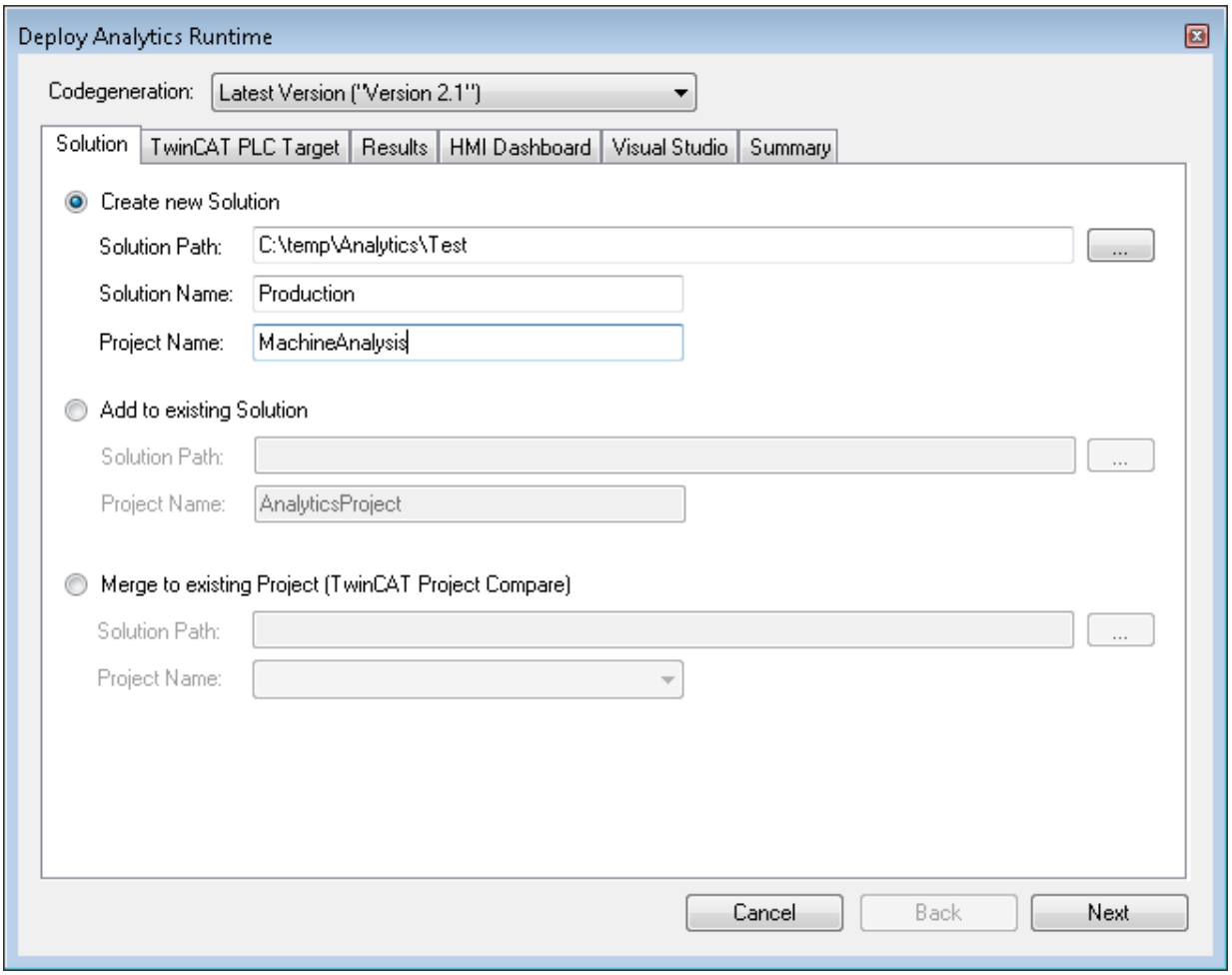
The last major step in the TwinCAT Analytics workflow is the continuous 24-hour machine analysis. It runs in parallel with the machine applications in the field. To make this very easy, the TwinCAT Analytics Workbench can automatically generate PLC code and an HTML5-based dashboard of your Analytics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

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

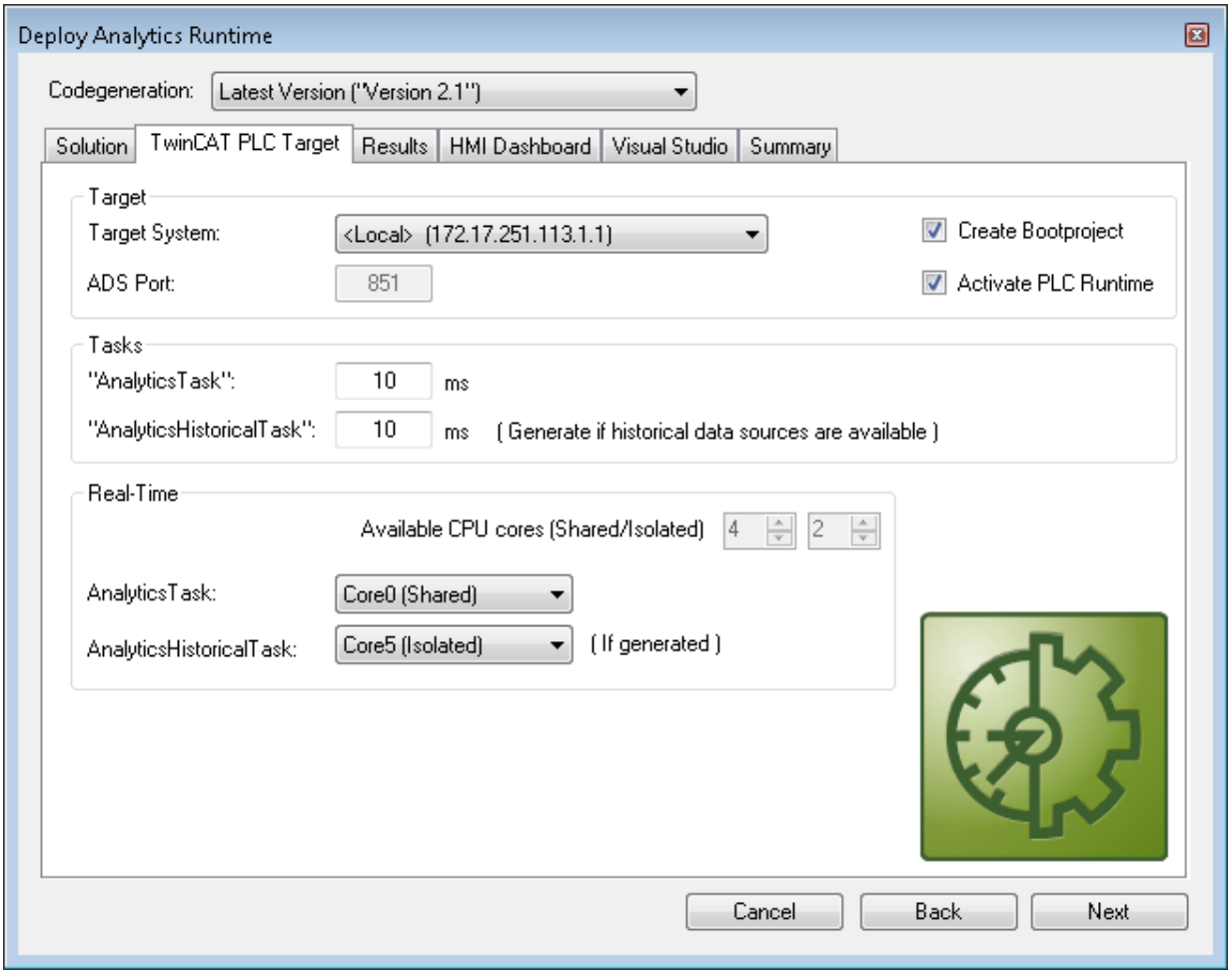


1. When the wizard is open, you can click through some tabs. The first one is called Solution. Here you can decide how your Analytics project should be used in the PLC code: As...
completely new solution.

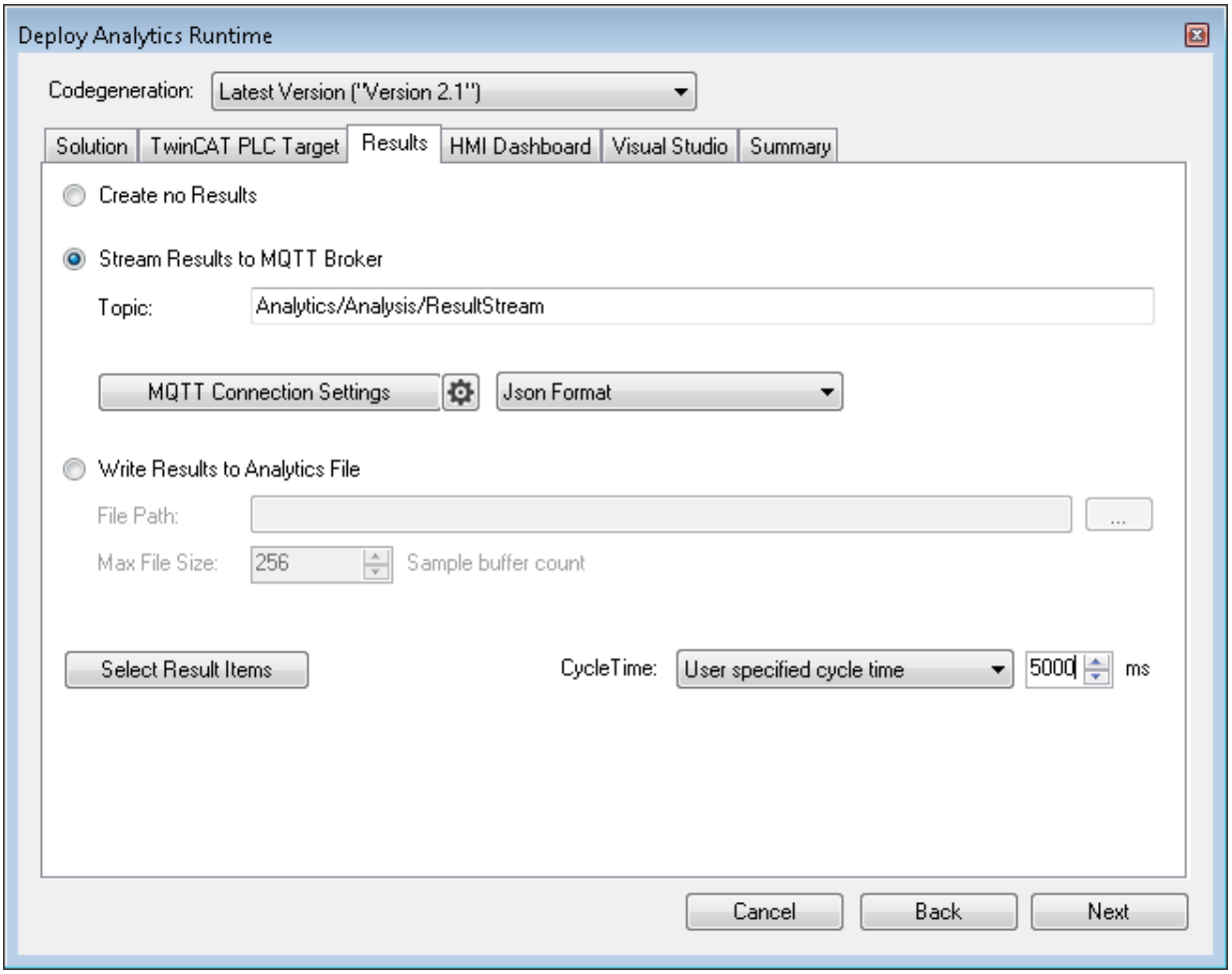
part of an existing solution.
update of an existing Analytics solution.



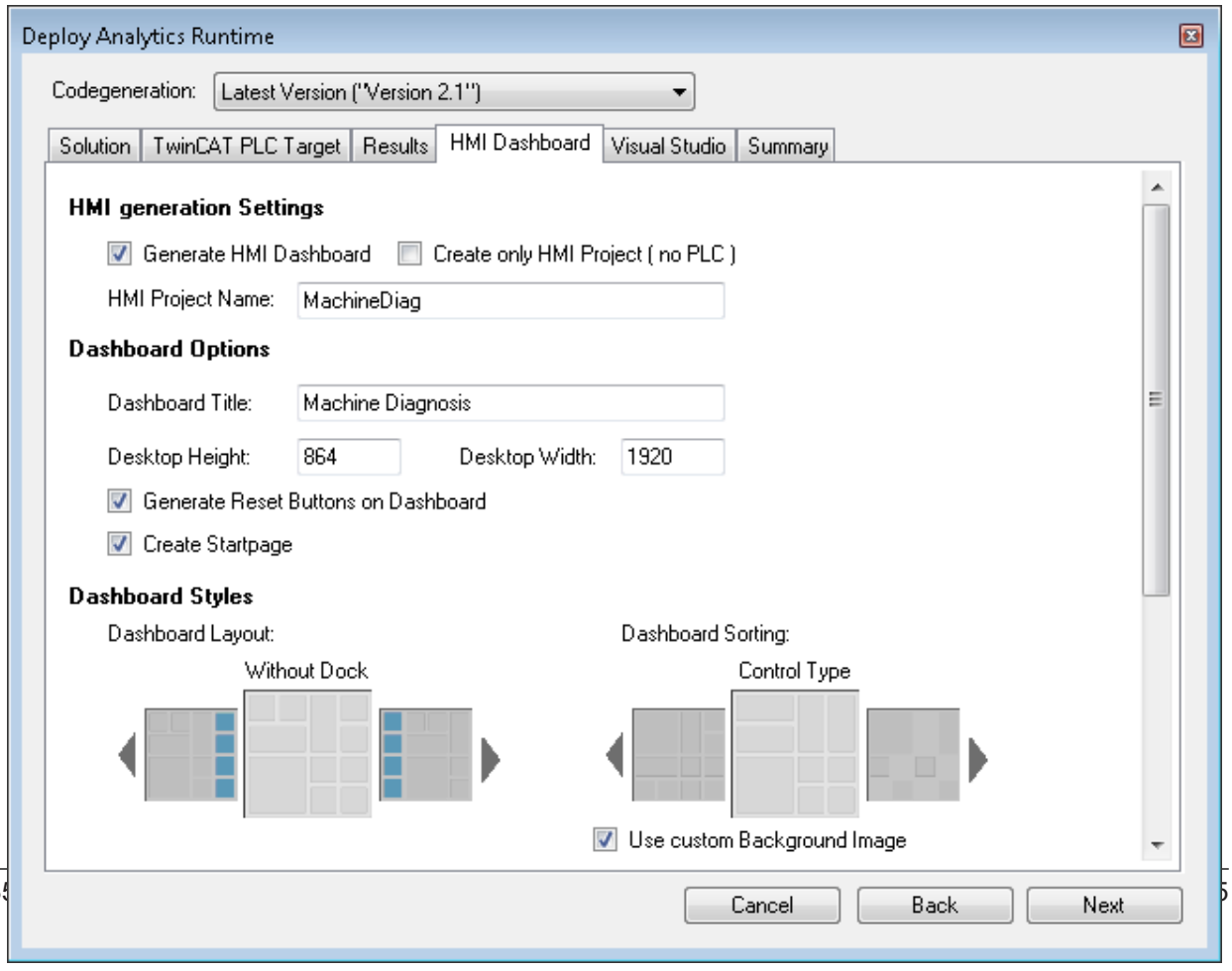
2. In the **TwinCAT PLC Target** tab you can select the ADS target system that runs the TwinCAT Analytics Runtime (TF3550). The created project is immediately executable. For this purpose you can set the Activate PLC Runtime option. In addition, it can be selected that a boot project is created directly.



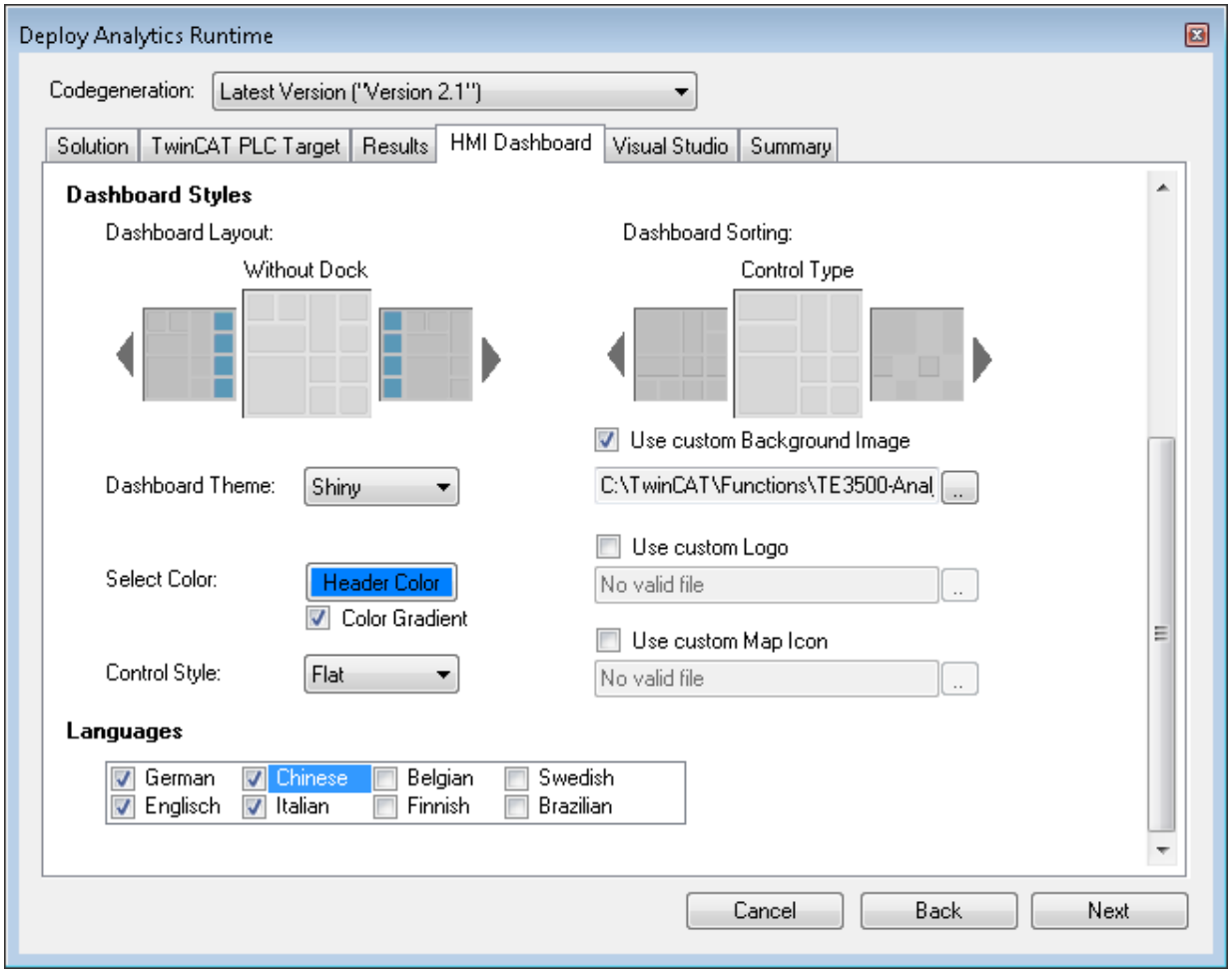
3. Especially for virtual machines, it is important to run the project on isolated cores, which is also an option in this tab. The next tab **Results** is needed only if you have selected the **Stream Results** option in the algorithm properties. If you want to send results, you can decide here in which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



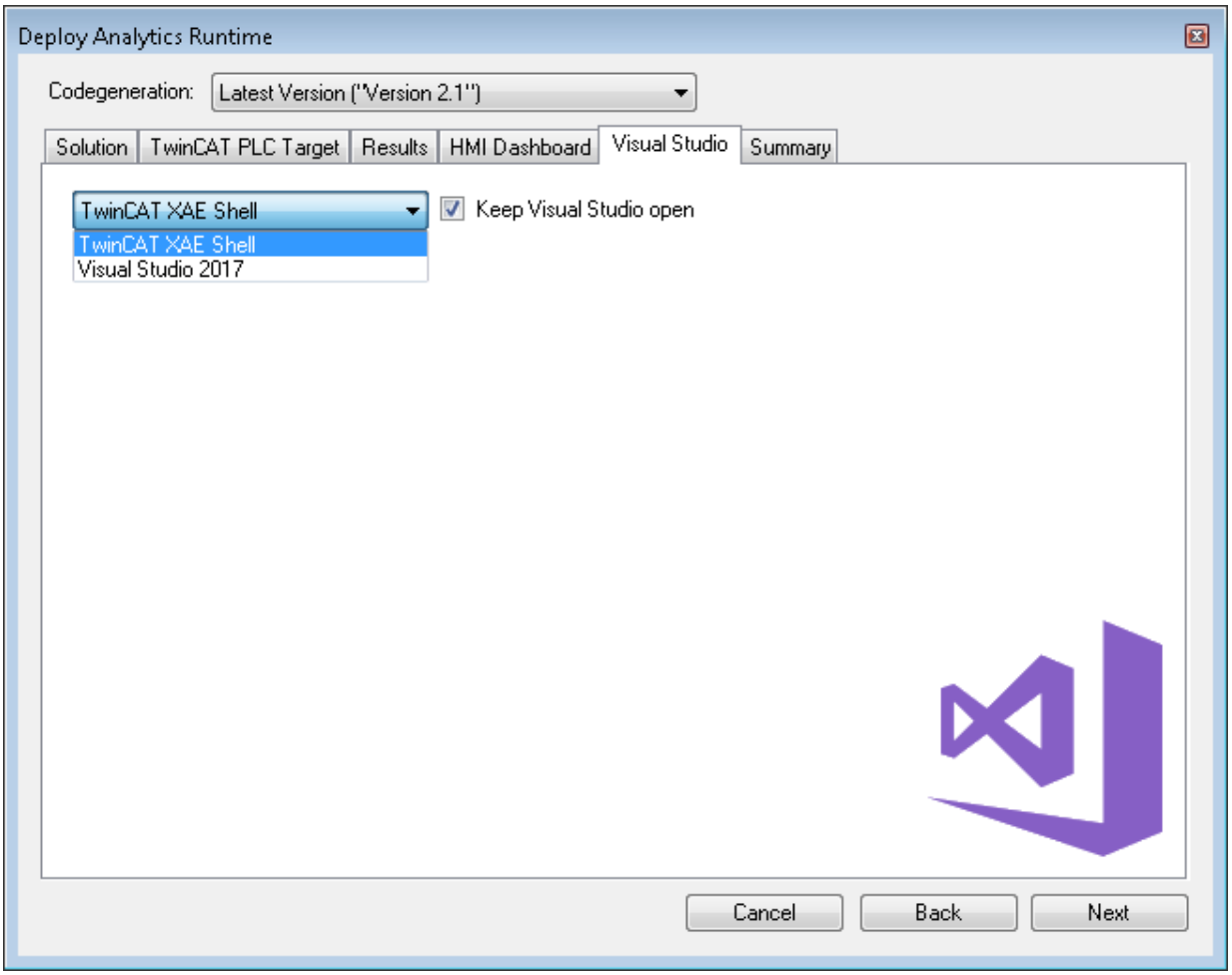
Downsampling of the results is possible by specifying a cycle time. The next tab is for the **HMI Dashboard**. A prerequisite for the automatic generation of the dashboard is the selection of HMI Controls for the corresponding algorithms whose results are to be displayed in the dashboard.



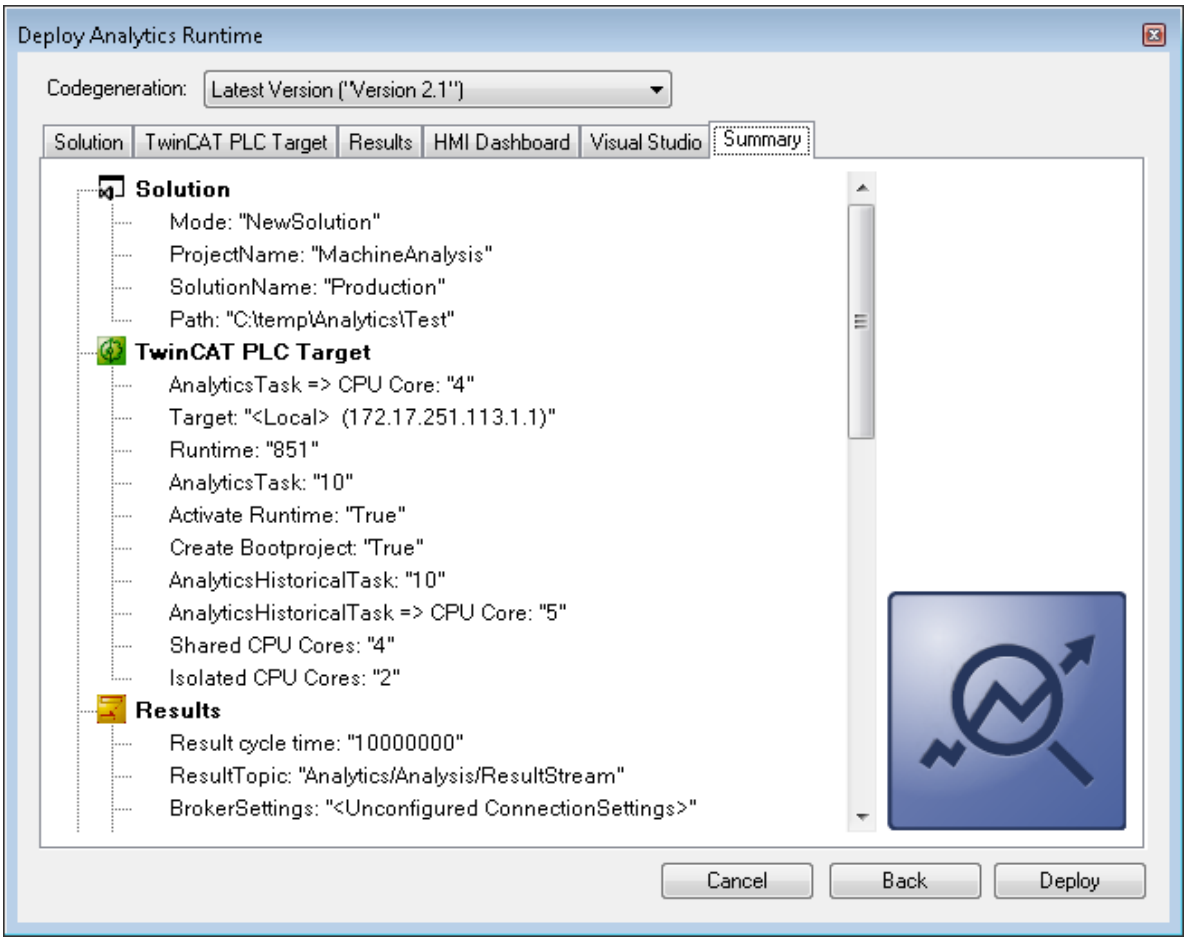
4. You can choose different options for your Analytics Dashboard, such as a start page with a map, layouts, sorting algorithms, custom colors and logos. If you select multiple languages for the Analytics Controls, a language switching menu will also be generated.



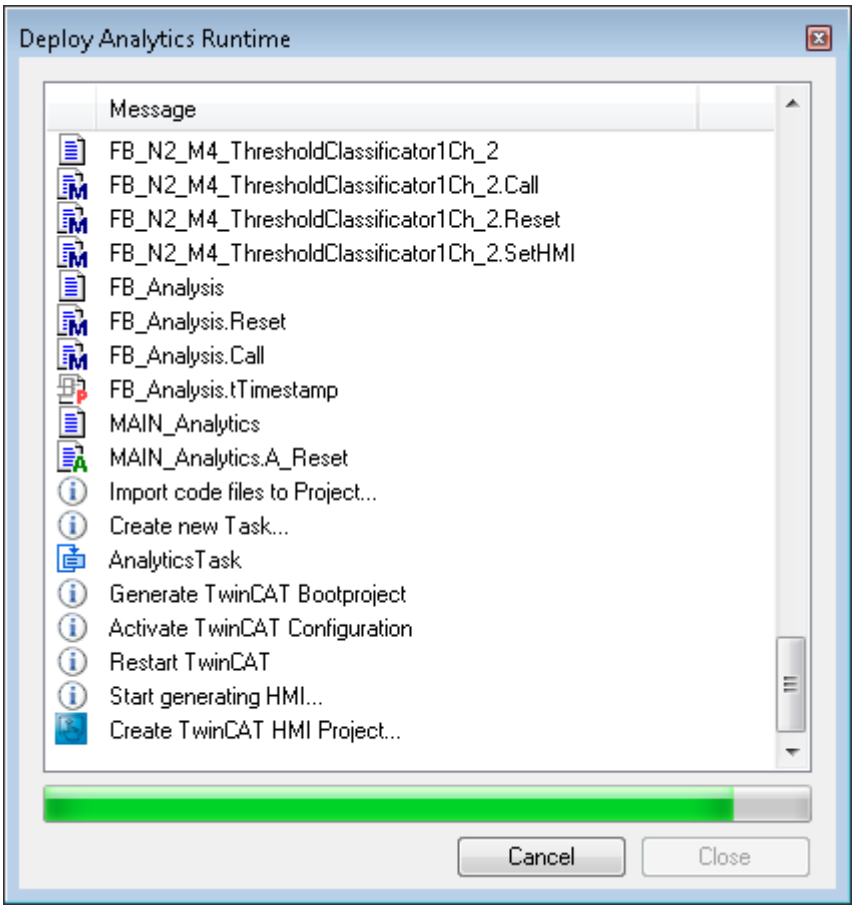
5. Select one of the installed versions of Visual Studio® and, whether the instance should start visibly or just be set up and activated in the background.

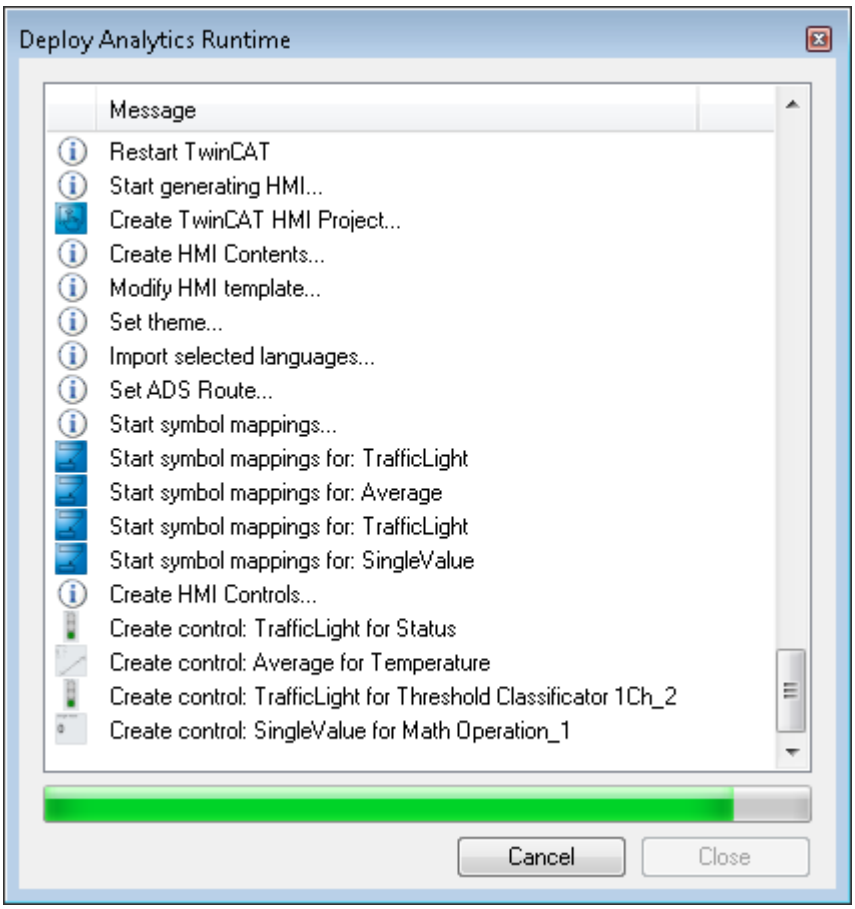


⇨ At last you can find an overview.

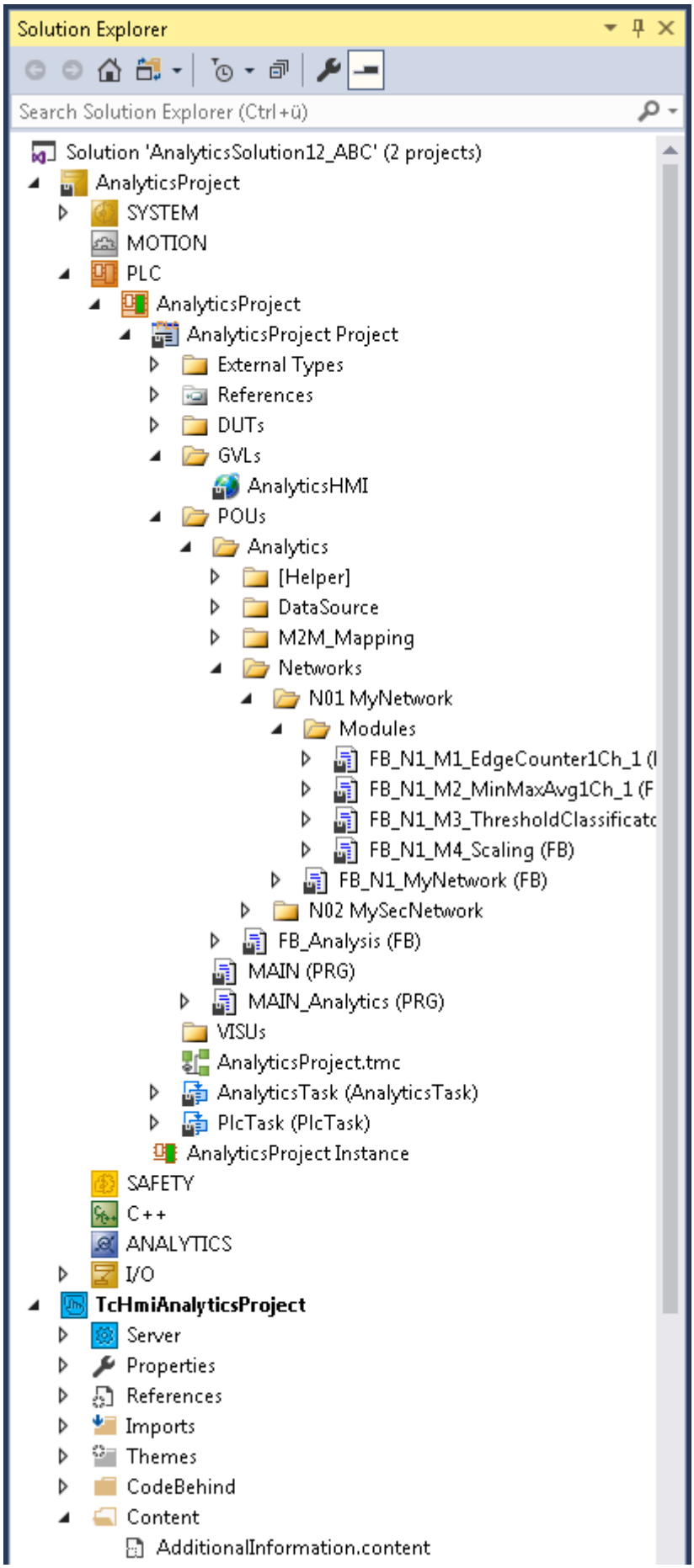


6. Now you can click the **Deploy** button to start the generation process. The PLC project and the HMI dashboard are now generated.





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



5 Technical introduction

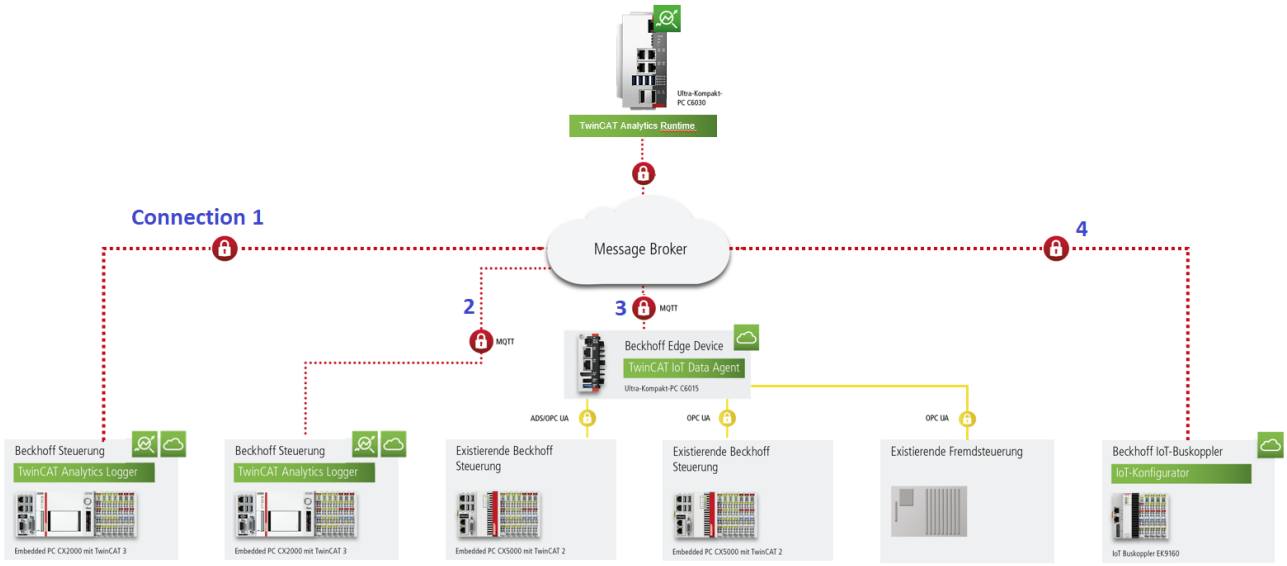
Regarding the TwinCAT Analytics workflow the Analytics Runtime is able to make a continuous analysis of data from different remote controller. The Analytics Runtime based on a standard TwinCAT PLC Runtime. The application code can be generated automatically by the Analytics Workbench. So, the user has no additional engineering effort, but is free to do code changes or to add own analysis code. The usage of other standard Beckhoff PLC libraries like Tc3_Database or Tc3_ConditionMonitoring is possible, too.

Connection handling with remote devices

The number of devices/controller to be analyzed is limited. The Analytics Runtime itself allows four connections to controller without additional packages.

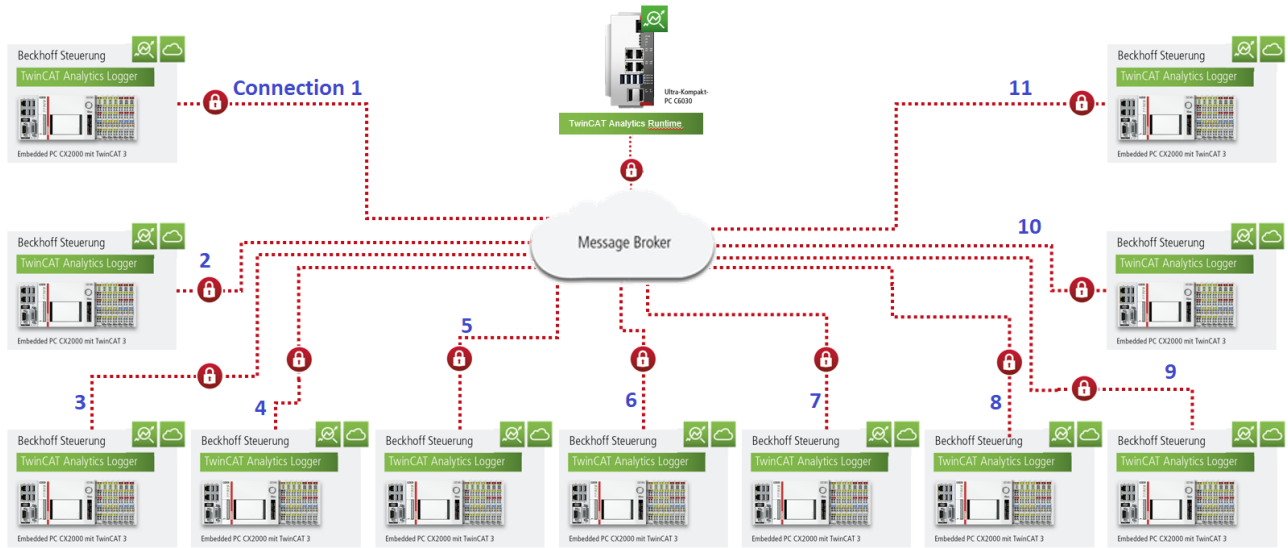
Currently **one** connection is counted for the following controllers:

- Controller with TF3500 Analytics Logger
- Controller with TF6720 IoT Data Agent (can have many underlaid controllers)
- EK9160 IoT Coupler



If you need more than four connections in your application, you can add Controller Packs. Possible packs are shown here.

The picture below shows an application architecture with 11 connected controller. In this case you have to order a license for Controller Pack 8 on the Analytics Runtime device/VM in addition to the TwinCAT Analytics Runtime license.



TwinCAT HMI

Based on the analysis PLC code the engineer is able to design an own HTML 5 dashboard with TE2000 which is integrated in the setup of TE3500 Analytics Workbench. The Analytics Runtime includes the HMI Server. The HMI Server provides the HTML 5 web pages for at least four clients at the same time. The Server provides one client connection by himself and the Analytics Runtime includes the licence for the Client Pack 3. Which gives you the possibility of 4 connections.

6 Configuration

The configuration of the Analytics Runtime is carried out in the engineering tool TE3500 Analytics Workbench. The necessary steps are explained again here. If you need more information about the configuration in TE3500, please go to the documentation of TE3500.

6.1 Runtime deployment

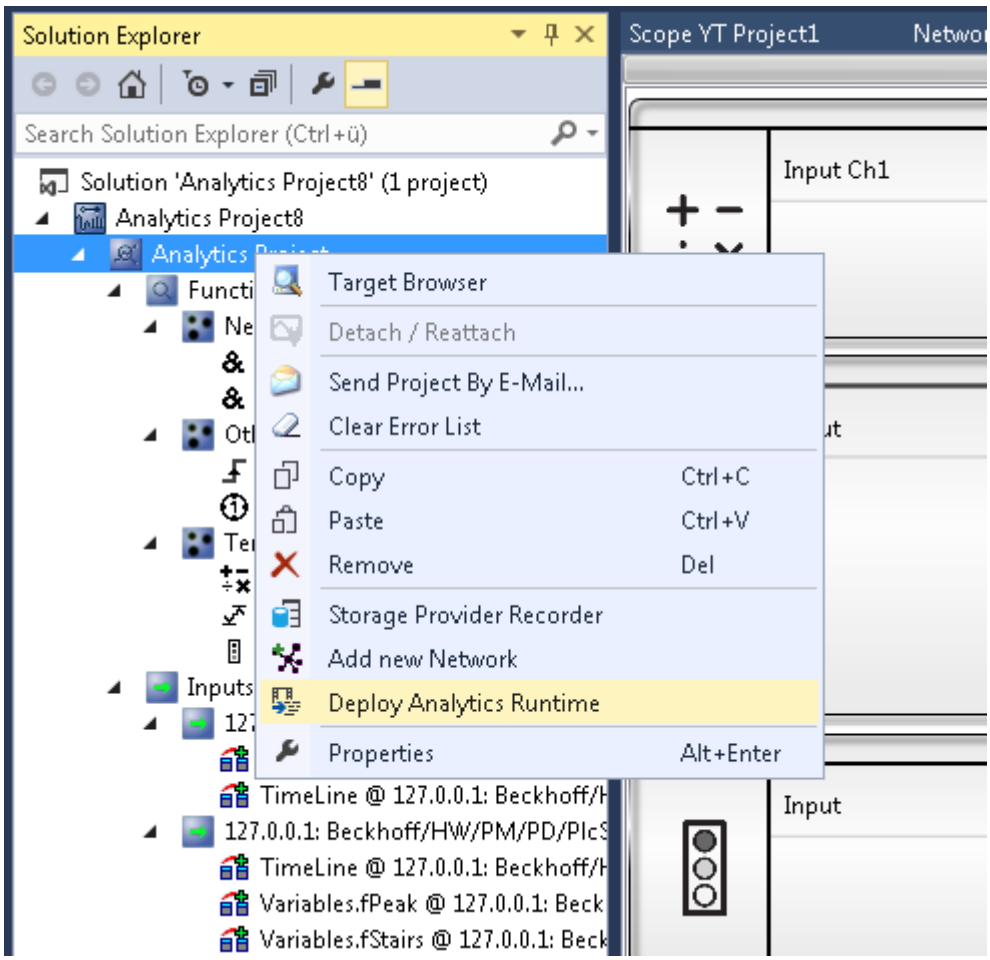
PLC code can be generated with all modules and parameters configured in the TwinCAT Analytics Workbench Configurator. This code can be downloaded into a TwinCAT Analytics runtime to realize 24/7 data analysis.

NOTICE

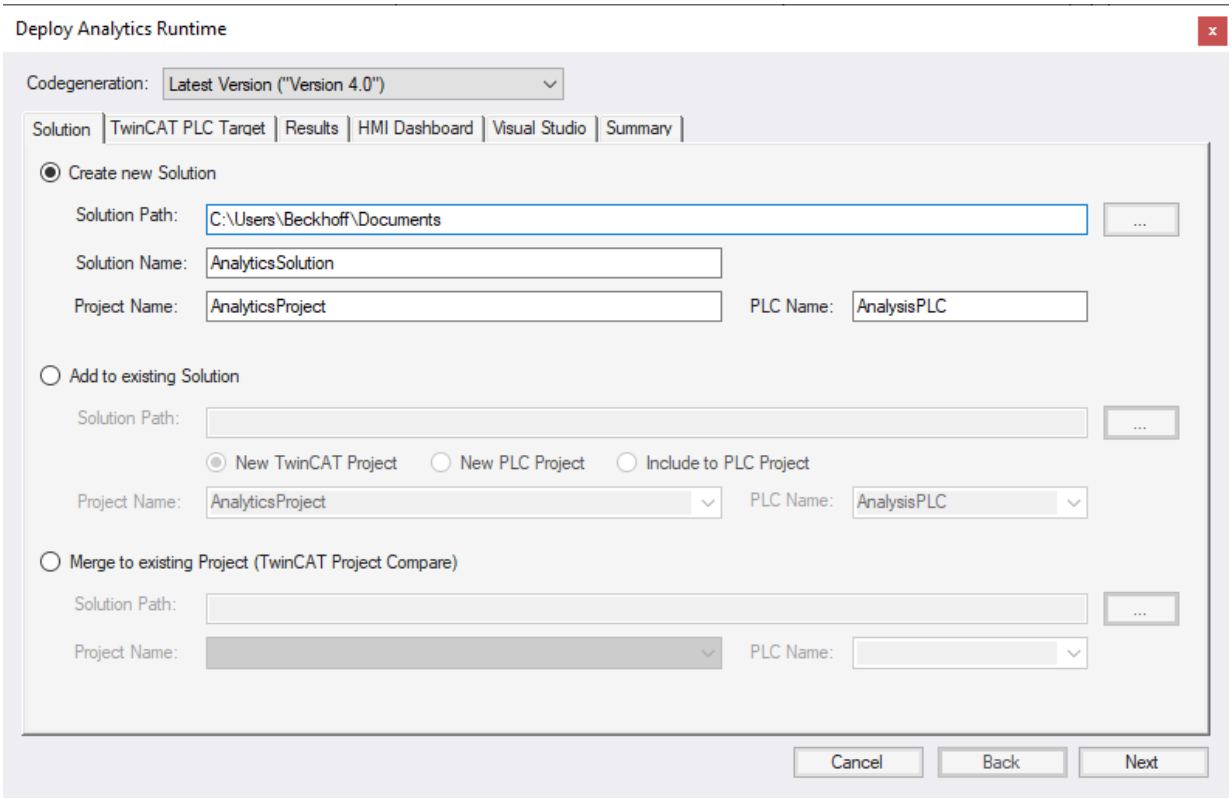
Compatibility of automatically generated PLC code

The automatically generated PLC code is based on the TwinCAT Analytics Library. Interfaces of the base models of the library are compatible with earlier versions after release of the library. The automatically generated code itself is only a sample code! This code generation may change from version to version. As far as possible, this is solved by the code generation version.

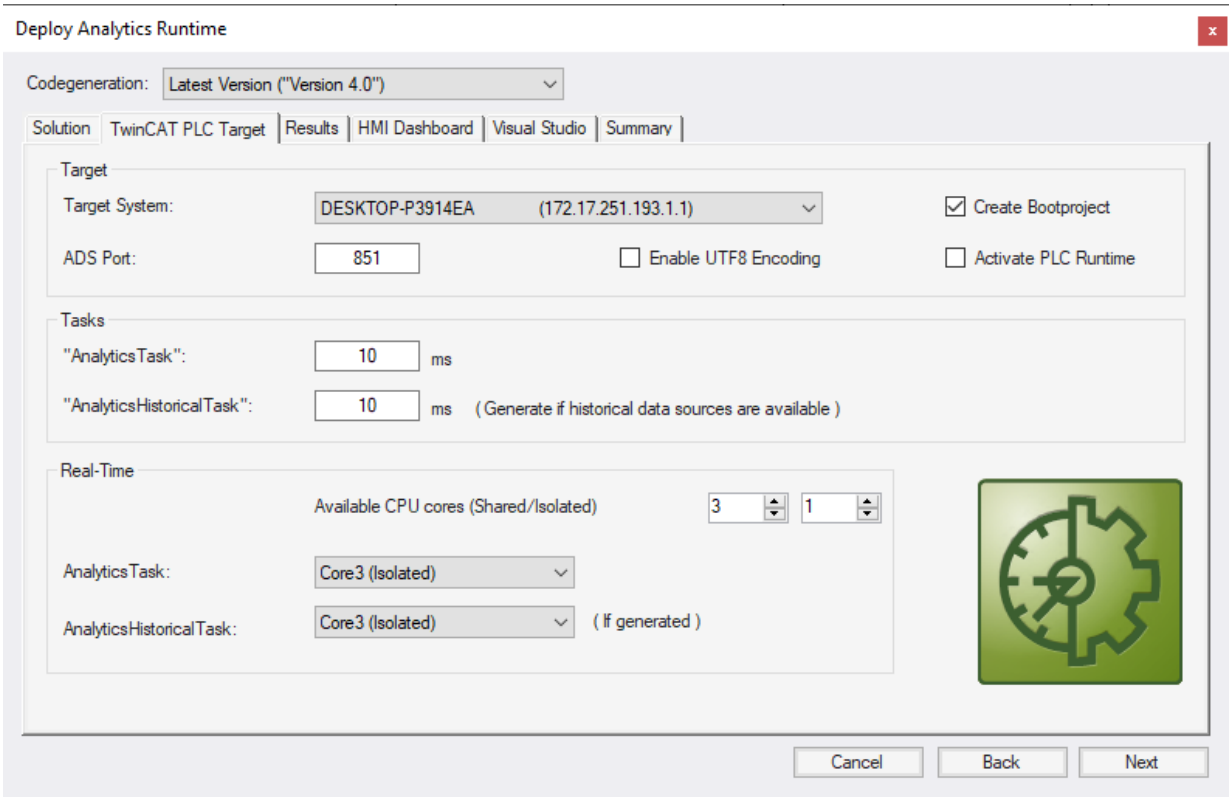
1. After configuration, you can click on the **Deploy Analytics Runtime** command in the context menu, as shown in the image below.



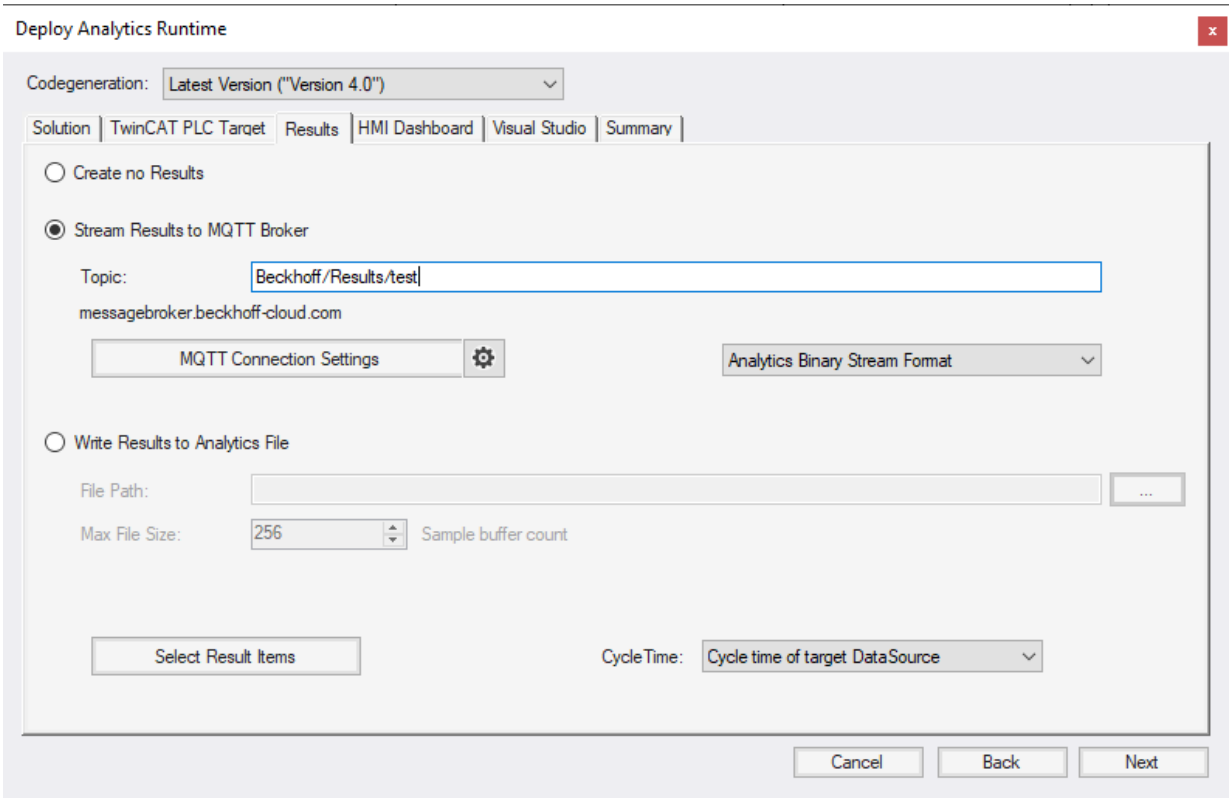
2. The Deploy Wizard starts and it is possible to set up the entire required configuration step by step for use.



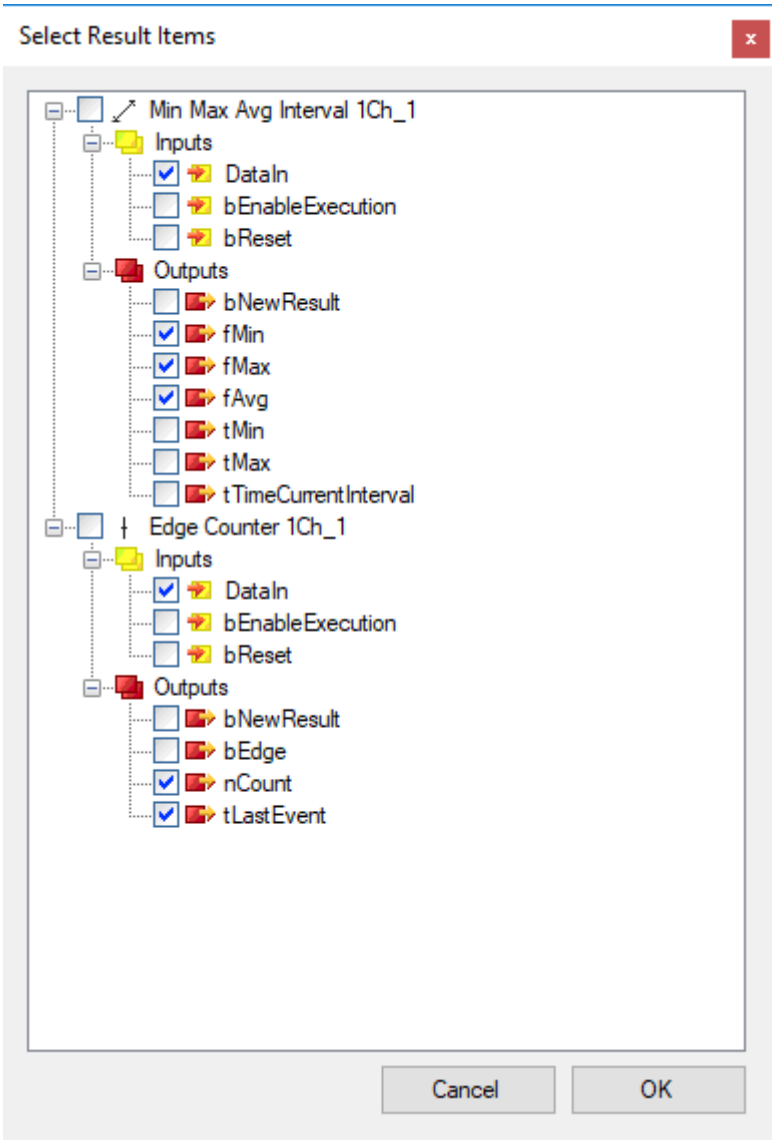
3. On the first tab **Solution** you can select whether a new project folder is created, whether the analysis PLC code is integrated or added to an existing project folder as a new project, new PLC or to an existing PLC, or whether the new project is inserted into an existing project folder using the TwinCAT Project Compare Tool.



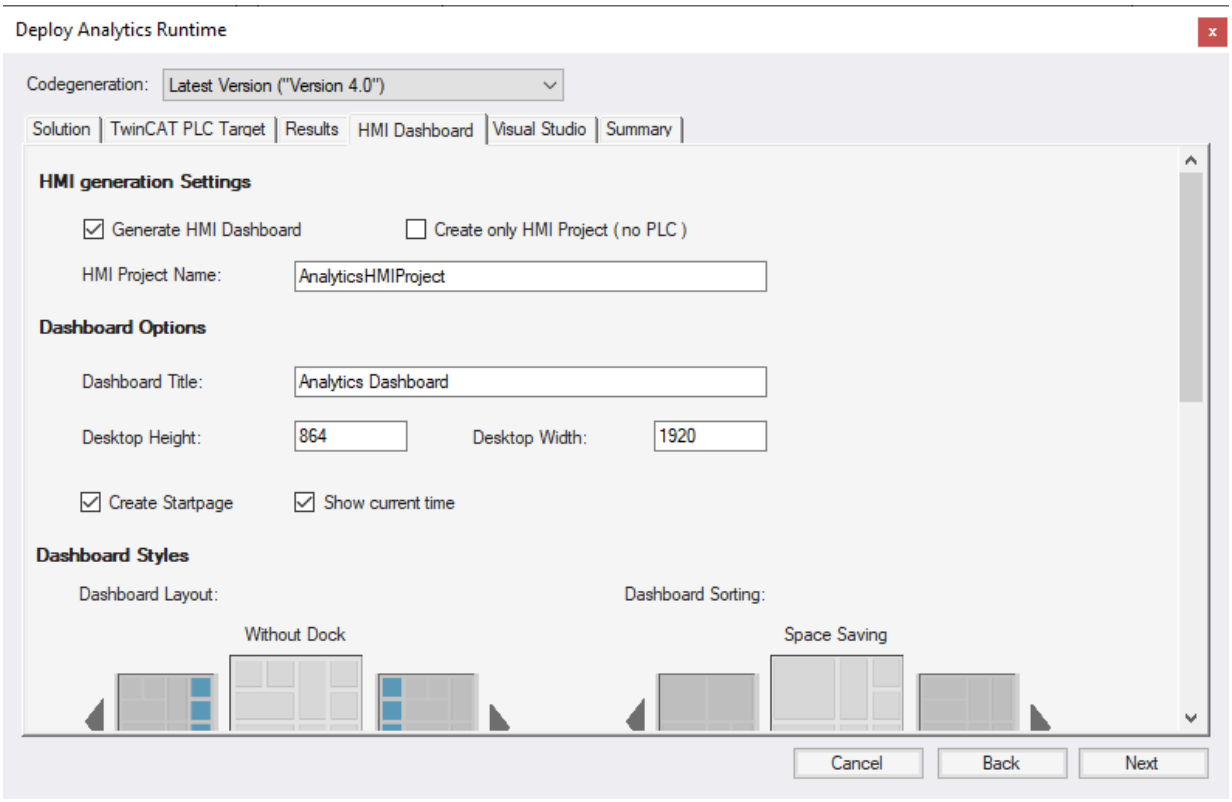
4. On the second tab, **TwinCAT PLC Target**, all PLC-specific parameters such as target system, task cycle time or the task assignment to corresponding CPU cores can be defined.



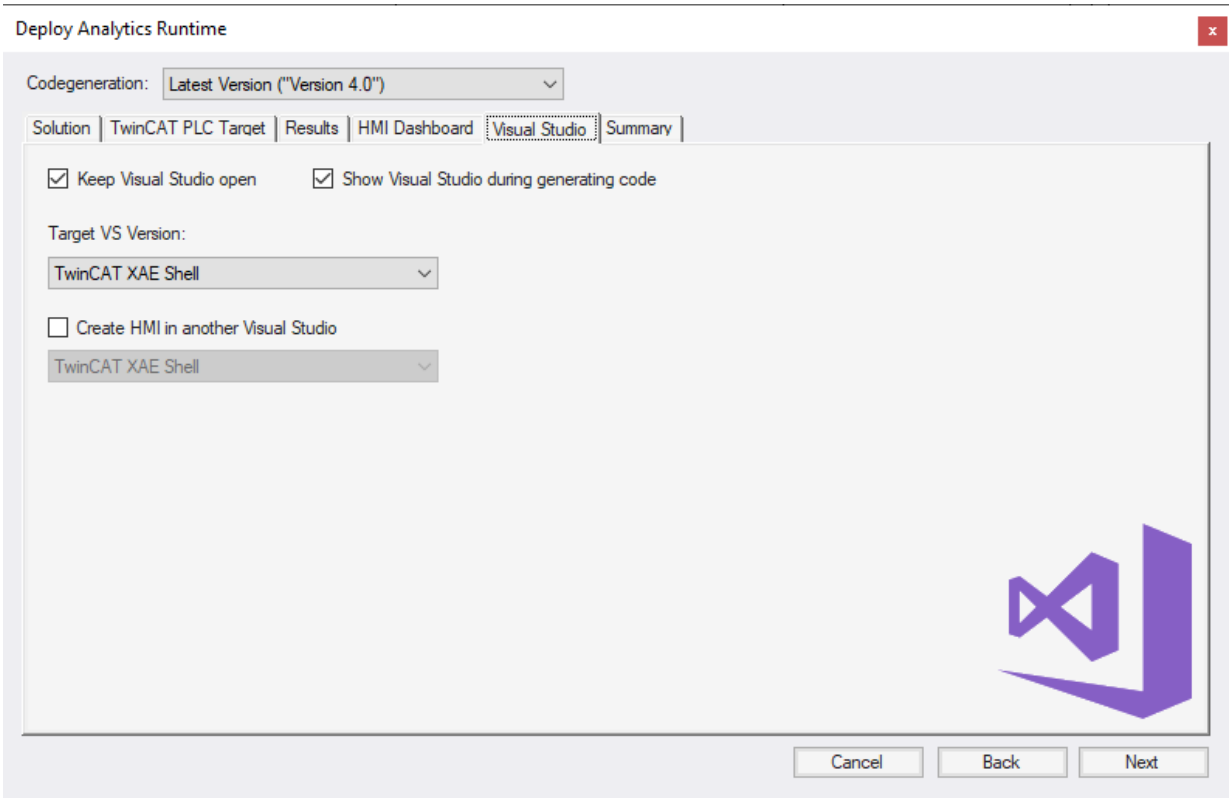
5. If you have set the "PLC Result" property of some functions in the configurator, the **Result** tab is enabled in the Deploy wizard. There you can set up where the results will be streamed or stored.



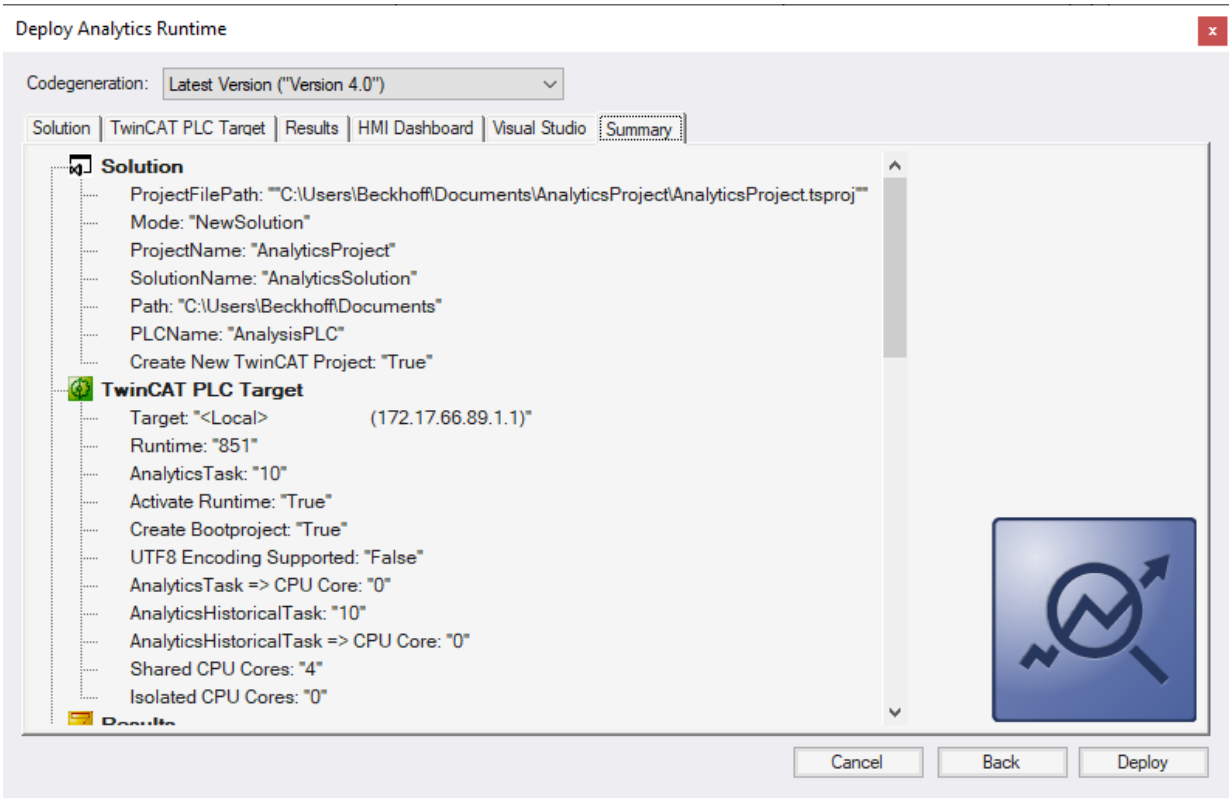
6. By clicking **Select Result Items** it is possible to select only the desired values.



7. If you have assigned HMI controls to one or more functions, the **HMI Dashboard** tab is enabled in the Deploy wizard. Here, various settings can be made to generate a customized Dashboard that fits your needs.

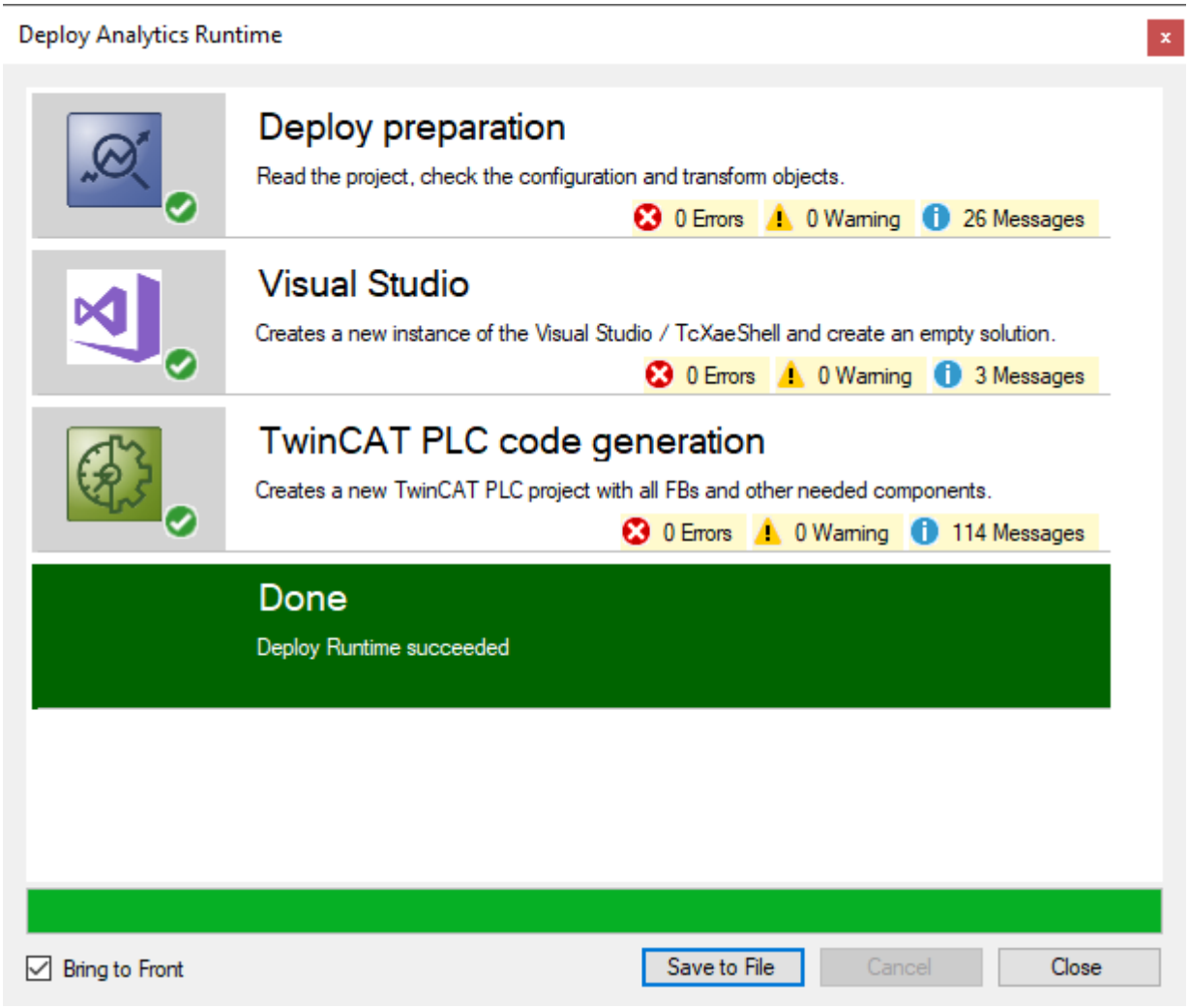


8. On the next tab, **Visual Studio**, you can select which Visual Studio version or TwinCAT XAE shell should be used for generation, if several are installed.



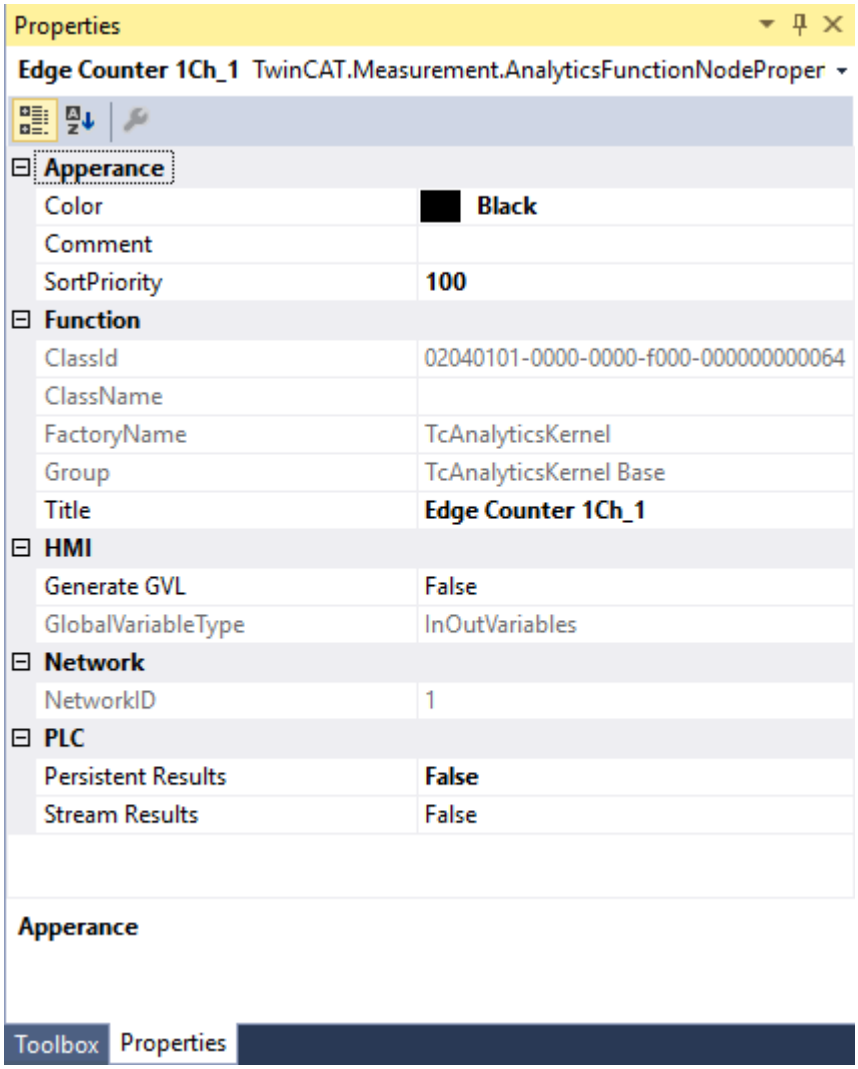
9. The last tab shows you all the settings you have made for the generation. Now you can start the generation process by clicking **Deploy**.

⇨ In the overview window, each step during the generation process is displayed, clearly arranged and divided into categories.



6.1.1 Algorithm properties

Each algorithm of the Analytics Configurator is providing some properties. The sections of HMI and PLC are necessary for the automatic code generation.



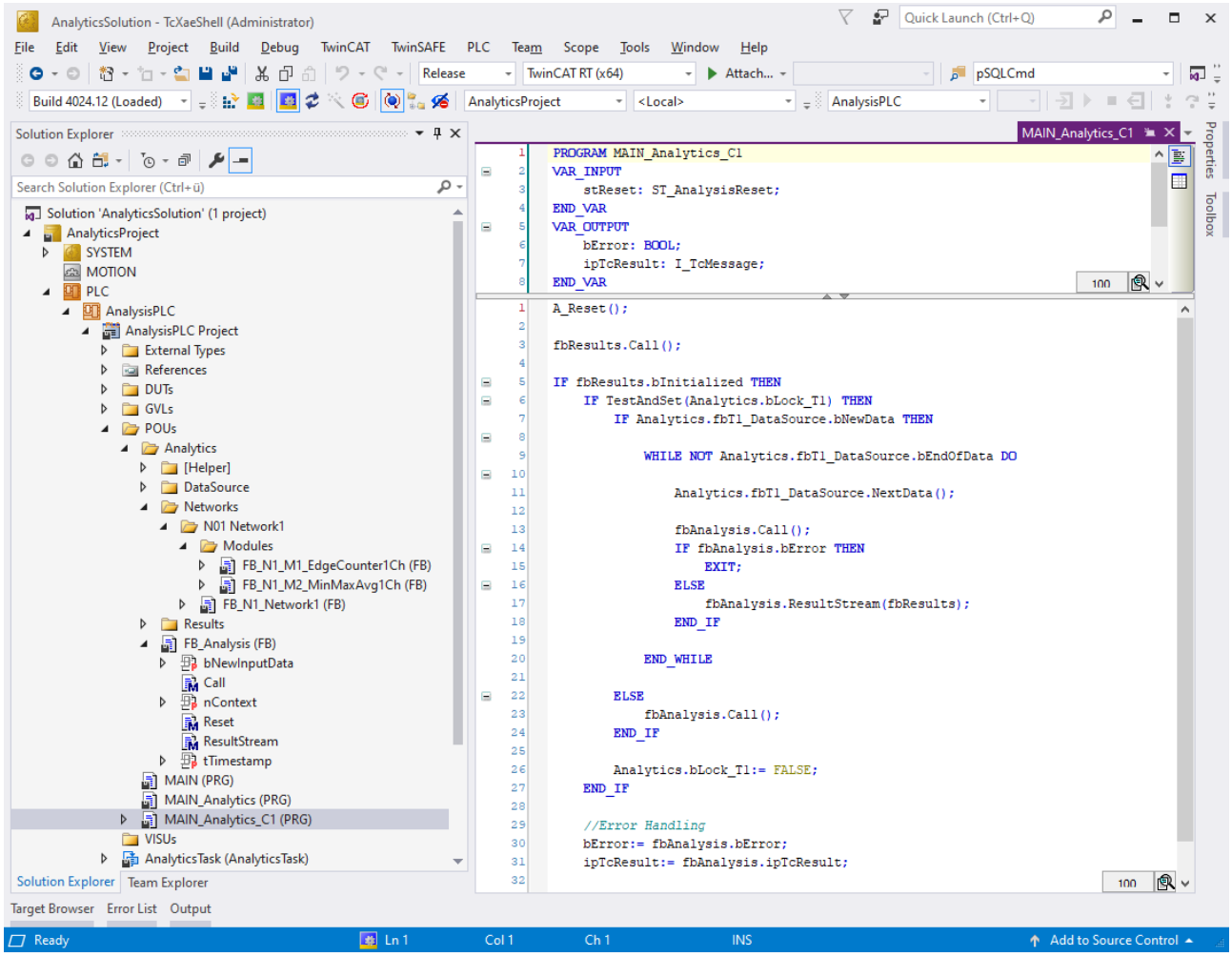
HMI

- **Generate GVL:** Enable the generation of an Global Variable List with a collection of variables and corresponding data type mapping for TwinCAT HMI
- **GlobalVariableType:** Choose the type with InOutVariables just for inputs and outputs of the algorithm or KeyValuePairs for general mapping to STRING for tables
-

PLC

- **Persistent Results:** Enable this flag to store results of algorithm persistent to target system of the Analytics Runtime
- **Stream Results:** Enable this flag to add the In- and Outputs of the algorithm to a result stream which will be generated by the code generation

6.1.2 PLC Code



NOTICE

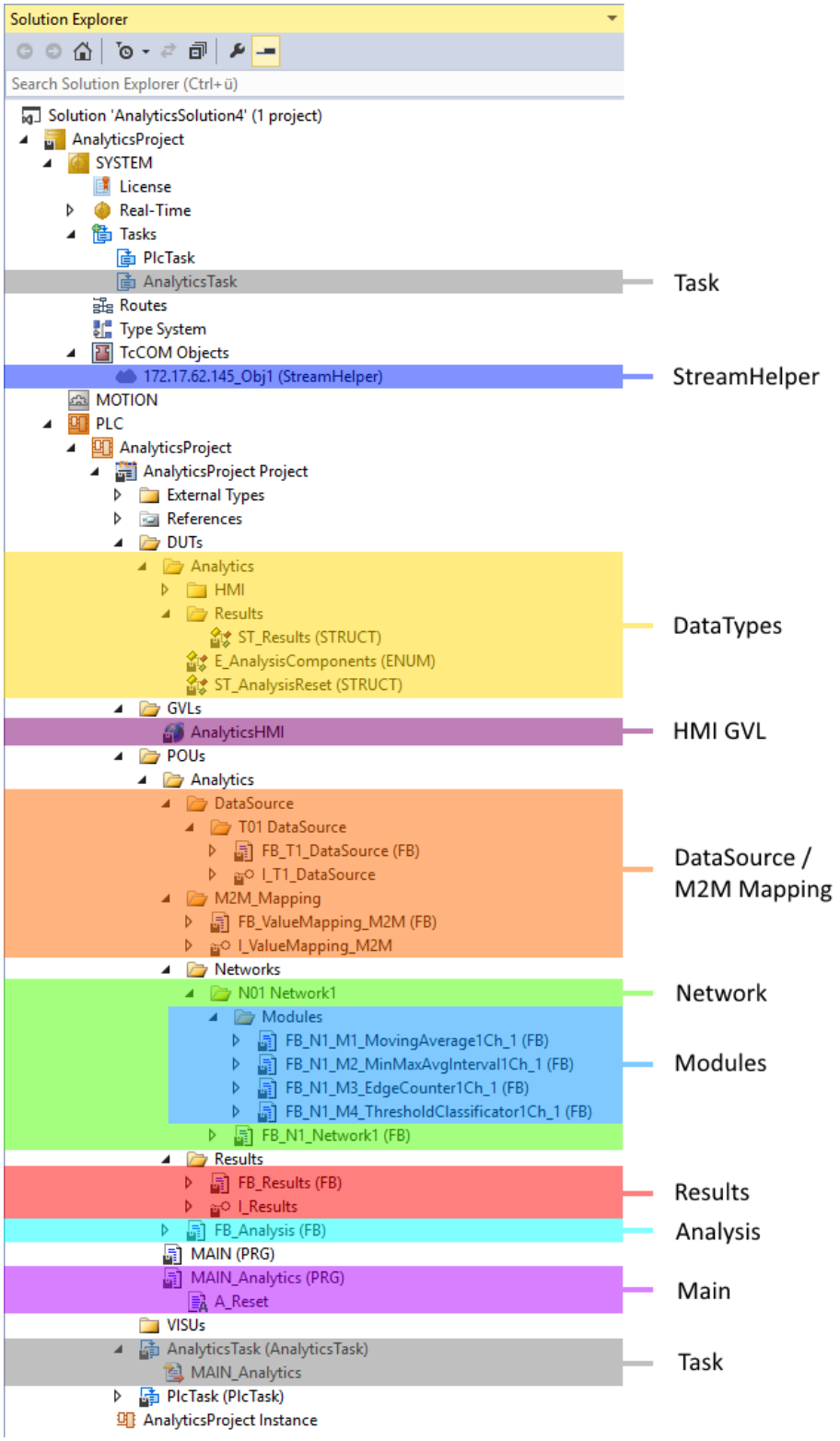
Compatibility of automatically generated PLC code

The automatically generated PLC code is based on the TwinCAT Analytics Library. Interfaces of the base models of the library are compatible with earlier versions after release of the library. The automatically generated code itself is only a sample code! This code generation may change from version to version.

Code generation version compatibility

	Version 2.1 (obsolete)	Version 3.0 (obsolete)	Version 4.0	Version 4.1	Version 5.0
Analytics algorithms	(X)	(X)	X	X	X
Filter algorithms	(X)	(X)	X	X	X
HMI support	(X)	(X)	X	X	X
HMI with advanced input stream handling	-	(X)	X	X	X
Support of array inputs	(X)	(X)	X	X	X
Support of oversampling inputs	-	-	X	X	X
Network templates	(X)	(X)	X	X	X
"Closed" network templates	-	-	X	X	X
Network with inputs, outputs and parameters	-	-	X	X	X
"Condition Monitoring" network templates	-	-	X	X	X
Lambda algorithms	-	-	X	X	X
Writing parameters via HMI	-	-	-	X	X
"Power Monitoring" algorithms	-	-	X	X	X
Scope Support	-	-	-	-	X

6.1.2.1 Code version 2.1



Task:

A separate task is created for the analytics analysis.

StreamHelper:

If one or more data sources are of type MQTT binary stream, the code generation creates an instance of a StreamHelper object to process the incoming binary stream patterns.

DataTypes:

The data types are created for the analysis. They contain STRUCTs for the reset function or result processing and ENUMs to select the various components.

HMI GVL:

To conveniently map module inputs and outputs with the HMI dashboard, selected variables are generated as global variables.

DataSource/M2M Mapping:

The FB DataSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs. The FB ValueMapping_M2M manages the value mapping between the modules (M2M - Module to Module) from the module INPUTs to the module OUTPUTs.

Network:

All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

Modules:

The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

Results:

If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker or saves the data to the Analytics binary file.

Analysis:

The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

MAIN:

The FB Analysis is called in the program MAIN_Analytics. The program is assigned to the separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.

All reset calls are defined in the action A_Reset.

Requirements

Development environment	Target platform	PIc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics Version >= 3.1.0.0

6.1.2.1.1 FB_DataSource

The DataSource FB manage the receiving of the input values of the different sources. In the OUTPUT declaration you can find all configured inputs.

Syntax

Definition:

```
FUNCTION_BLOCK FB_T[n]_DataSource IMPLEMENTS I_T[n]_DataSource
VAR
END_VAR
```

Methods

Name	Definition location	Description
Call [▶ 55]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
GetData [▶ 55]	Local	Method to get the data 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

6.1.2.1.1.1 GetData

Syntax

```
METHOD GetData : BOOL
VAR_INPUT
    nElement : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nElement	UDINT	Element ID to get the specific sample

Return value

Name	Type	Description
GetData	BOOL	Is TRUE if a new element is selected

6.1.2.1.1.2 Call

Syntax

```
METHOD Call : BOOL
```

Return value

Name	Type	Description
Call	BOOL	

6.1.2.1.2 FB_Network

All modules are sorted in a specific network to get a better overview and structure of the configured analysis.

Syntax

Definition:

```
FUNCTION_BLOCK FB_N[n]_[Network1]
VAR_INPUT
    [module FBs]
END_VAR
VAR_OUTPUT
    bError: BOOL;
```

```
    ipTcResult: I_TcMessage;
END_VAR
```

Inputs

Name	Type	Description
Module FBs		FBs of the configured modules

Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE as soon as an error situation occurs.
ipTcResult	I_TcMessage	Message interface from the TwinCAT 3 EventLogger, which provides details on the return value.

Methods

Name	Definition location	Description
Call [▶ 60]	Local	Method for background communication. The method must be called cyclically.
Reset [▶ 60]	Local	Reset the Network with all sub modules
ValueMapping [▶ 56]	Local	Map the input values to the different module inputs
SetHMIValues [▶ 57]	Local	Optional: Map in- outputs of the modules to the global HMI variable

Requirements

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

6.1.2.1.2.1 ValueMapping

Syntax

```
METHOD ValueMapping : BOOL
VAR_INPUT
    ipDataSource : I_T[n]_DataSource;
END_VAR
```

Inputs

Name	Type	Description
ipDataSource	I_T[n]_DataSource	Data for the analysis from the specific data source

Return value

Name	Type	Description
ValueMapping	BOOL	

6.1.2.1.2.2 SetHMIValues

Syntax

```
METHOD SetHMIValues : BOOL
VAR_INPUT
    pHMI_N[n]_[Network1] : POINTER TO ST_HMI_N[n]_[Network1];
END_VAR
```

Inputs

Name	Type	Description
pHMI_N[n]_[Network1]	POINTER TO ST_HMI_N[n]_[Network1]	Pointer to global HMI struct

Return value

Name	Type	Description
SetHMIValues	BOOL	Is TRUE if done

6.1.2.1.2.3 Reset

Syntax

```
METHOD Reset : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Is TRUE if done

6.1.2.1.2.4 Call

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    ipDataSource: I_T[n]_DataSource;
    [ipValueMapping_M2M: I_ValueMapping_M2M;]
END_VAR
```

Inputs

Name	Type	Description
ipDataSource	I_T[n]_DataSource	Data for the analysis.
ipValueMapping_M2M	I_ValueMapping_M2M	Optional: Needed for mapping values between modules

Return value

Name	Type	Description
Call	BOOL	

6.1.2.1.3 FB_Module

The module FBs contains all inputs and outputs of the configured modules from the Workbench Configurator. It is also possible to reconfigure the modules at runtime. You only have to change the parameter and then start the reconfigure process with a rising edge at the bReconfigure INPUT.

Syntax

Definition:

```
FUNCTION_BLOCK FB_N[n]_M[n]_[Module]
VAR_INPUT
    [module inputs]
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
    [module outputs]
END_VAR
```

Inputs

Name	Type	Description
Module inputs		Inputs of the selected module

Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE as soon as an error situation occurs.
ipTcResult	I_TcMessage	Message interface from the TwinCAT 3 EventLogger, which provides details on the return value.
Module outputs		Outputs of the selected module

Methods

Name	Definition location	Description
Call [▶ 58]	Local	Method for background communication. The method must be called cyclically.
Reset [▶ 59]	Local	Reset the module
SetHMI [▶ 59]	Local	Optional: Sets the in- outputs to the global HMI structs

Requirements

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

6.1.2.1.3.1 Call

Syntax

```
METHOD Call : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	

6.1.2.1.3.2 Reset

Syntax

```
METHOD Reset : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Is TRUE if done

6.1.2.1.3.3 SetHMI

Syntax

```
METHOD SetHMIValues : BOOL
VAR_INPUT
    pHMI_N[n]_[Network1] : POINTER TO ST_HMI_N[n]_[Network1];
END_VAR
```

Inputs

Name	Type	Description
pHMI_N[n]_[Network1]	POINTER TO ST_HMI_N[n]_[Network1]	Pointer to global HMI struct

Return value

Name	Type	Description
SetHMI	BOOL	Is TRUE if done

6.1.2.1.4 FB_Analysis

In the analysis FB the whole analytics routine is defined. All configured networks with their modules and error handling is created.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Analysis
VAR_INPUT
    [network FBs]
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
```

Inputs

Name	Type	Description
Network FBs		FBs of the configured networks

Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE as soon as an error situation occurs.
ipTcResult	I_TcMessage	Message interface from the TwinCAT 3 EventLogger, which provides details on the return value.

Methods

Name	Definition location	Description
Call [▶ 60]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
Reset [▶ 60]	Local	Reset the whole analysis
ResultStream [▶ 61]	Local	Optional: If a result stream has to be created

Requirements

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

6.1.2.1.4.1 Call

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    ipDataSource: I_T[n]_DataSource;
END_VAR
```

Inputs

Name	Type	Description
ipDataSource	I_T[n]_DataSource	Data for the analysis.

Return value

Name	Type	Description
Call	BOOL	

6.1.2.1.4.2 Reset

Syntax

```
METHOD Reset : BOOL
VAR_IN_OUT
    stReset: ST_AnalysisReset;
END_VAR
```

Inputs

Name	Type	Description
stReset	ST_AnalysisReset	Struct to define witch module or network should be reset.

Return value

Name	Type	Description
Reset	BOOL	Is TRUE if done

6.1.2.1.4.3 ResultStream

Syntax

```
METHOD ResultStream : BOOL
VAR_INPUT
    ipResults: I_Results;
END_VAR
```

Inputs

Name	Type	Description
ipResults	I_Results	Interface pointer to the Result FB

Return value

Name	Type	Description
Call	BOOL	

6.1.2.1.5 FB_Results

If results of the analysis has to be stored or streamed, the result FB managed this and streamed the selected variables to the message broker or store the data into the analytics binary file.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Results
VAR_OUTPUT
    stResults: ST_Results;
END_VAR
```

Outputs

Name	Type	Description
stResults	ST_Results	Result struct which contains all items of the result stream

Methods

Name	Definition location	Description
Call [▶ 62]	Local	Method for background communication. The method must be called cyclically.
AddResult [▶ 62]	Local	Add a sample to the result stream
SendResults [▶ 62]	Local	Sends all buffered samples of the result stream
Release [▶ 63]	Local	Close stream or file of the result stream

Requirements

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

6.1.2.1.5.1 Call

Syntax

```
METHOD Call : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	

6.1.2.1.5.2 AddResult

Syntax

```
METHOD AddResult : BOOL
VAR_INPUT
    tTimestamp: ULINT;
    stSample: ST_Results;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Timestamp of the sample
stSample	ST_Results	Sample struct

Return value

Name	Type	Description
AddResult	BOOL	

6.1.2.1.5.3 SendResults

Syntax

```
METHOD SendResults : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
SendResults	BOOL	

6.1.2.1.5.4 Release

Syntax

```
METHOD Release : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Release	BOOL	

6.1.2.1.6 MAIN_Analytics

In the MAIN_Analytics program the analysis FB is called. The program is assign to the separated Task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First you have to choose the component you would like to reset. Then a rising edge at the bReset INPUT starts the reset process.

Inside of the A_Reset Action all reset calls are defined.

Syntax

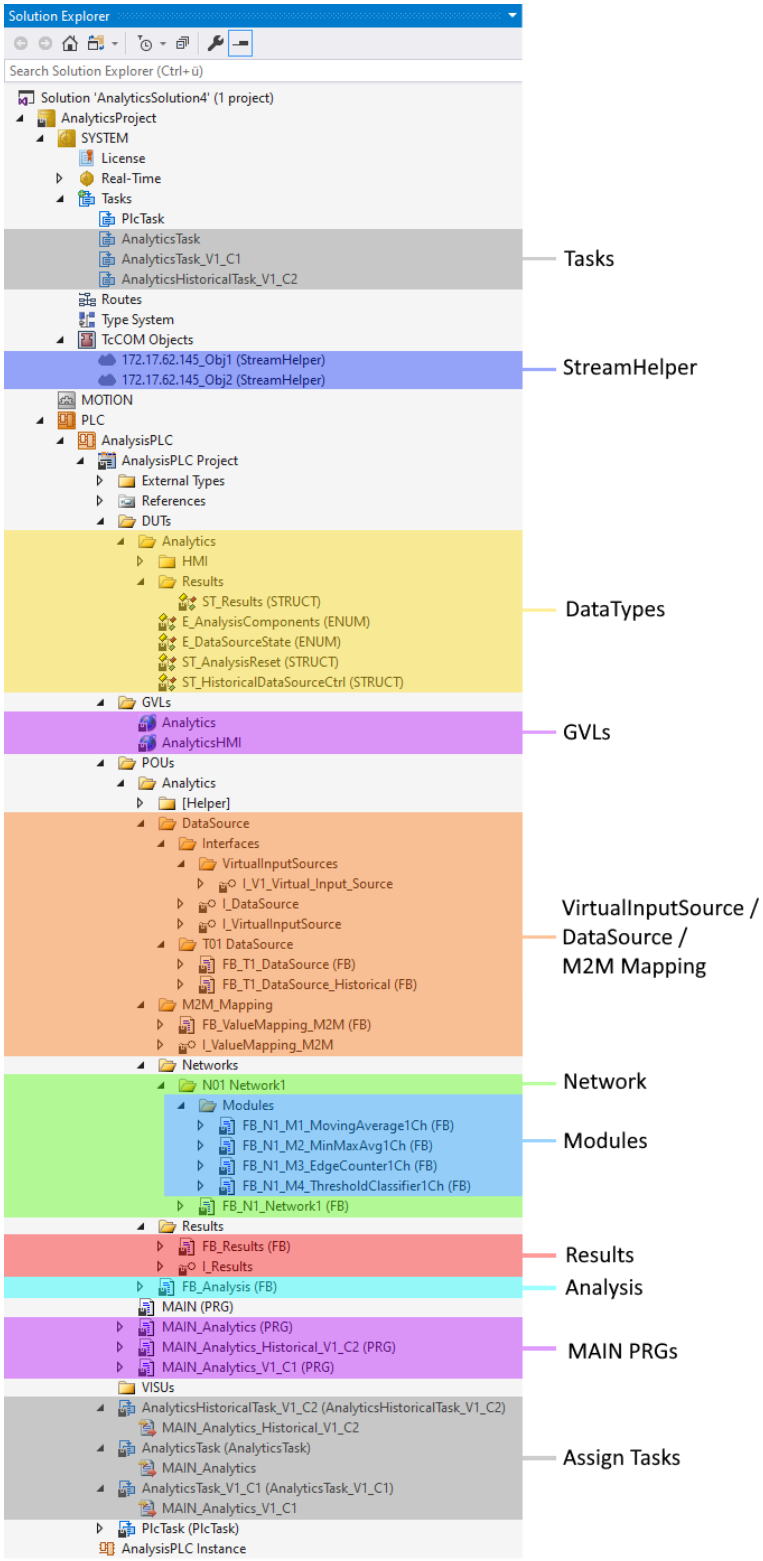
Definition:

```
PROGRAM MAIN_Analytics
VAR_INPUT
    stReset: ST_AnalysisReset;
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
```

Requirements

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

6.1.2.2 Code version 3.0



Tasks:

A separate task is created for the analytics analysis and for each configuration of a Virtual Input Source.

StreamHelper:

For each data source of type MQTT binary stream, the code generation creates an instance of a stream helper object to process the incoming binary stream patterns.

DataTypes:

The data types are created for the analysis. They contain STRUCTs for the reset function or result processing and ENUMs to select the various components.

GVLs:

To conveniently map module inputs and outputs with the HMI dashboard, selected variables are generated as global variables. In addition, the Data Source function block instances and various parameters are generated as global variables.

VirtualInputSource / DataSource / M2M Mapping:

The Virtual Input Source interfaces abstract the Data Source symbols from the analysis. The FB DataSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs. The FB ValueMapping_M2M manages the value mapping between the modules (M2M - Module to Module) from the module INPUTs to the module OUTPUTs.

Network:

All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

Modules:

The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

Results:

If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker in binary or Json format, or saves the data locally to an Analytics binary file.

Analysis:

The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

MAIN PRGs:

In the MAIN_Analytics program, the DataSource FBs are called, the reset function is managed and, if appropriate, the values are mapped with the HMI dashboard. The program is assigned to a separate task.

The FB Analysis is called in the programs MAIN_Analytics_Vx_Cx. The programs are each assigned to a separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.

All reset calls are defined in the action A_Reset.

Requirements

Development environment	Target platform	PIc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics Version >= 3.1.0.0

6.1.2.2.1 FB_DataSource

The FB DataSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs.

Syntax

Definition:

```
FUNCTION_BLOCK FB_T[n]_DataSource IMPLEMENTS I_DataSource, I_V[n]_Virtual_Input_Source
VAR_OUTPUT
eDataState: E_DataSourceState;
END_VAR
```

Methods

Name	Definition location	Description
Call [▶ 66]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
GetData [▶ 66]	Local	Method to retrieve the data of the specified element.
GetDataOversampling [▶ 67]	Local	Method to retrieve the oversampling data of the specified element.
NewDataAvailable [▶ 67]	Local	Method to check if new data is available.
HistoricalCtrl [▶ 67]	Local	Method for retrieving historical data.
UpdateRecordList [▶ 68]	Local	Method for updating the historical record list.

Requirements

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

6.1.2.2.1.1 GetData

Syntax

```
METHOD GetData : BOOL
VAR_INPUT
    nElement : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nElement	UDINT	Element ID to obtain the specific example.

Return value

Name	Type	Description
GetData	BOOL	Is TRUE if a new element is selected.

6.1.2.2.1.2 Call

Syntax

```
METHOD Call : BOOL
```

Return value

Name	Type	Description
Call	BOOL	

6.1.2.2.1.3 GetDataOversampling

Syntax

```
METHOD GetDataOversampling : BOOL
VAR_INPUT
nElement : UDINT;
nSample: : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nElement	UDINT	Element ID to obtain the specific sample element.
nSample	UDINT	Sample ID to obtain the specific sample.

Return value

Name	Type	Description
GetDataOversampling	BOOL	Is TRUE if a new element is selected.

6.1.2.2.1.4 NewDataAvailable

Syntax

```
METHOD NewDataAvailable : BOOL
VAR_INPUT
    nLastDataHandle : ULINT;
END_VAR
```

Inputs

Name	Type	Description
nLastDataHandle	ULINT	Handle from the last fetched data packet.

Return value

Name	Type	Description
NewDataAvailable	BOOL	Is TRUE if new data is available.

6.1.2.2.1.5 HistoricalCtrl

Syntax

```
METHOD HistoricalCtrl : BOOL
VAR_INPUT
    stCtrl : REFERENCE TO ST_HMI_DataSourceCtrl;
    stHistStreamInfo : REFERENCE TO ST_HMI_DataSourceHist;
    stRecordInfo : REFERENCE TO ST_HMI_DataSourceHistRecordInfo;
END_VAR
```

Inputs

Name	Type	Description
stCtrl	ST_HMI_DataSourceCtrl	
stHistStreamInfo	ST_HMI_DataSourceHist	
stRecordInfo	ST_HMI_DataSourceHistRecordInfo	

Return value

Name	Type	Description
HistoricalCtrl	BOOL	

6.1.2.2.1.6 UpdateRecordList

Syntax

```
METHOD UpdateRecordList : BOOL
VAR_INPUT
    stCtrl : REFERENCE TO ST_HMI_DataSourceCtrl;
    stHistStreamInfo : REFERENCE TO ST_HMI_DataSourceHist;
    sStreamSystemID : GUID;
END_VAR
```

Inputs

Name	Type	Description
stCtrl	ST_HMI_DataSourceCtrl	
stHistStreamInfo	ST_HMI_DataSourceHist	
sStreamSystemID	GUID	

Return value

Name	Type	Description
UpdateRecordList	BOOL	

6.1.2.2.2 FB_Network

All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

Syntax

Definition:

```
FUNCTION_BLOCK FB_N[n]_[Network1]
VAR_INPUT
    [module FBs]
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
```

Inputs

Name	Type	Description
Module FBs		Function blocks of the configured modules.

Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.

Methods

Name	Definition location	Description
Call [▶ 73]	Local	Method for background communication. The method must be called cyclically.
Reset [▶ 73]	Local	Resetting the network with all submodules.
ValueMapping [▶ 69]	Local	Assignment of the input values to the various module inputs.
SetHMIValues [▶ 69]	Local	Optional: Mapping of the inputs/outputs of the modules to the global HMI variable.

Requirements

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

6.1.2.2.2.1 ValueMapping

Syntax

```
METHOD ValueMapping : BOOL
VAR_INPUT
    ipVirtual_Input_Source : I_V[n]_Virtual_Input_Source;
END_VAR
```

Inputs

Name	Type	Description
ipVirtual_Input_Source	I_V[n]_Virtual_Input_Source	Data for analysis from the specific data source.

Return value

Name	Type	Description
ValueMapping	BOOL	

6.1.2.2.2.2 SetHMIValues

Syntax

```
METHOD SetHMIValues : BOOL
VAR_INPUT
    pHMI_N[n]_[Network1] : POINTER TO ST_HMI_N[n]_[Network1];
END_VAR
```

Inputs

Name	Type	Description
pHMI_N[n]_[Network1]	POINTER TO ST_HMI_N[n]_[Network1]	Pointer to the global HMI structure.

Return value

Name	Type	Description
SetHMIValues	BOOL	Is TRUE when completed.

6.1.2.2.2.3 Reset

Syntax

```
METHOD Reset : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

6.1.2.2.2.4 Call

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    ipVirtual_Input_Source: I_V[n]_Virtual_Input_Source;
    [ipValueMapping_M2M: I_ValueMapping_M2M;]
END_VAR
```

Inputs

Name	Type	Description
ipVirtual_Input_Source	I_T[n]_Virtual_Input_Source	Data for the analysis.
ipValueMapping_M2M	I_ValueMapping_M2M	Optional: Necessary for mapping values between modules.

Return value

Name	Type	Description
Call	BOOL	

6.1.2.2.3 FB_Module

The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

Syntax

Definition:

```
FUNCTION_BLOCK FB_N[n]_M[n]_[Module]
VAR_INPUT
    [module inputs]
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
    [module outputs]
END_VAR
```

Inputs

Name	Type	Description
Module inputs		Inputs of the selected module.

Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.
Module outputs		Outputs of the selected module.

Methods

Name	Definition location	Description
Call [▶ 71]	Local	Method for background communication. The method must be called cyclically.
Reset [▶ 71]	Local	Resetting the module.
SetHMI [▶ 72]	Local	Optional: Sets the inputs/outputs to the global HMI structures.

Requirements

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

6.1.2.2.3.1 Call

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    ipVirtual_Input_Source : I_V[n]_Virtual_Input_Source;
    [ipValueMapping_M2M : I_ValueMapping_M2M;]
END_VAR
```

Inputs

Name	Type	Description
ipVirtual_Input_Source	I_V[n]_Virtual_Input_Source	Data for the analysis.
ipValueMapping_M2M	I_ValueMapping_M2M	Optional: Necessary for mapping values between modules.

Return value

Name	Type	Description
Call	BOOL	

6.1.2.2.3.2 Reset

Syntax

```
METHOD Reset : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

6.1.2.2.3.3 SetHMI

Syntax

```
METHOD SetHMIValues : BOOL
VAR_INPUT
    pHMI_N[n]_[Network1] : POINTER TO ST_HMI_N[n]_[Network1];
END_VAR
```

Inputs

Name	Type	Description
pHMI_N[n]_[Network1]	POINTER TO ST_HMI_N[n]_[Network1]	Pointer to the global HMI structure.

Return value

Name	Type	Description
SetHMI	BOOL	Is TRUE when completed.

6.1.2.2.4 FB_Analysis

The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Analysis
VAR_INPUT
    [network FBs]
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
```

Inputs

Name	Type	Description
Network FBs		Function blocks of the configured networks.

Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.

Methods

Name	Definition location	Description
Call [▶ 73]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
Reset [▶ 73]	Local	Resets the entire analysis.
ResultStream [▶ 74]	Local	Optional: If a result stream needs to be created.

Requirements

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

6.1.2.2.4.1 Call

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    ipVirtual_Input_Source: I_V[n]_Virtual_Input_Source;
END_VAR
```

Inputs

Name	Type	Description
ipVirtual_Input_Source	I_V[n]_Virtual_Input_Source	Data for the analysis.

Return value

Name	Type	Description
Call	BOOL	

6.1.2.2.4.2 Reset

Syntax

```
METHOD Reset : BOOL
VAR_IN_OUT
    stReset: ST_AnalysisReset;
END_VAR
```

Inputs

Name	Type	Description
stReset	ST_AnalysisReset	Structure to define which module or network is to be reset.

Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

6.1.2.2.4.3 ResultStream

Syntax

```
METHOD ResultStream : BOOL
VAR_INPUT
    ipResults: I_Results;
END_VAR
```

Inputs

Name	Type	Description
ipResults	I_Results	Interface pointer to the FB Results.

Return value

Name	Type	Description
Call	BOOL	

6.1.2.2.5 FB_Results

If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker or saves the data to the Analytics binary file.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Results
VAR_OUTPUT
    stResults: ST_Results;
END_VAR
```

Outputs

Name	Type	Description
stResults	ST_Results	Result structure that contains all elements of the result stream.

Methods

Name	Definition location	Description
Call [▶ 75]	Local	Method for background communication. The method must be called cyclically.
AddResult [▶ 75]	Local	Add a sample to the result stream
SendResults [▶ 75]	Local	Sends all buffered samples of the result stream
Release [▶ 75]	Local	Close stream or file of the result stream

Requirements

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

6.1.2.2.5.1 Call

Syntax

```
METHOD Call : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	

6.1.2.2.5.2 AddResult

Syntax

```
METHOD AddResult : BOOL
VAR_INPUT
    tTimestamp: ULINT;
    stSample: ST_Results;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Timestamp of the sample
stSample	ST_Results	Sample structure

Return value

Name	Type	Description
AddResult	BOOL	

6.1.2.2.5.3 SendResults

Syntax

```
METHOD SendResults : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
SendResults	BOOL	

6.1.2.2.5.4 Release

Syntax

```
METHOD Release : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Release	BOOL	

6.1.2.2.6 MAIN_Analytics

In the MAIN_Analytics program, the DataSource FBs are called, the reset function is managed and, if appropriate, the values are mapped with the HMI Dashboard. The program is assigned to a separate task.

Syntax

Definition:

```
PROGRAM MAIN_Analytics
VAR_INPUT
    sCurrentStreamSystemID: GUID;
    stHistStreamInfo: REFERENCE TO ST_HMI_DataSourceHist;
    stHistRecordInfo: REFERENCE TO ST_HMI_DataSourceHistRecordInfo;
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
```

Requirements

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

6.1.2.2.7 MAIN_Analytics_V[n]_C[n]

The FB Analysis is called in the program MAIN_Analytics_V[n]_C[n]. The program is assigned to the separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.

All reset calls are defined in the action A_Reset.

Mapping of the HMI values takes place in the action A_MapToHMI.

Syntax

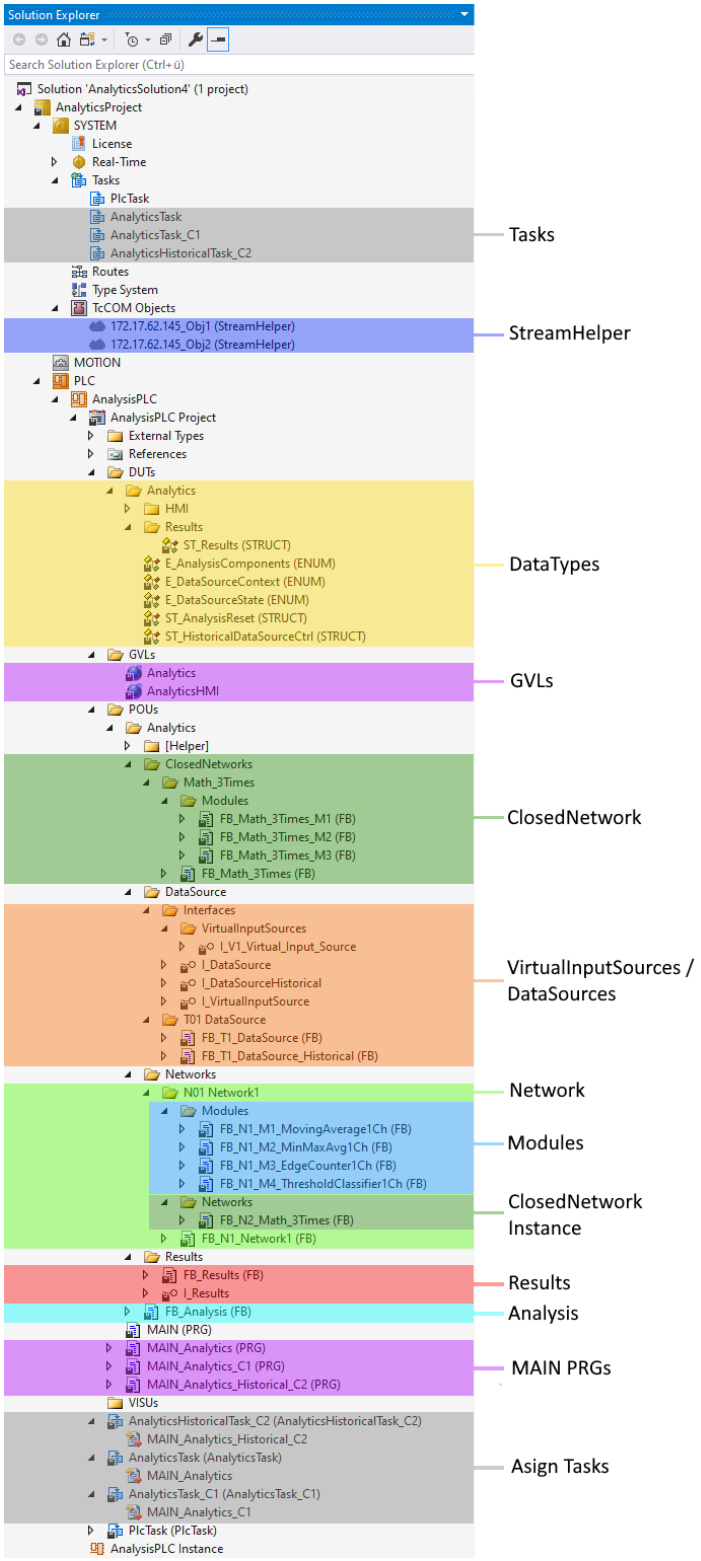
Definition:

```
PROGRAM MAIN_Analytics_V[n]_C[n]
VAR_INPUT
    stReset: ST_AnalysisReset;
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
    nLastDataHandle: ULINT;
END_VAR
```

Requirements

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

6.1.2.3 Code version 4.x



Tasks:

A separate task is created for the analytics analysis and for each configuration of a Virtual Input Source.

StreamHelper:

For each data source of type MQTT binary stream, the code generation creates an instance of a stream helper object to process the incoming binary stream patterns.

DataTypes:

The data types are created for the analysis. They contain STRUCTs for the reset function or result processing and ENUMs to select the various components.

GVLs:

To conveniently map module inputs and outputs with the HMI dashboard, selected variables are generated as global variables. In addition, the Data Source function block instances and various parameters are generated as global variables.

ClosedNetwork:

The ClosedNetwork FBs are generated once with all subnetworks and modules. They can be instantiated multiple times in the analysis. In this way, the generated code can be reduced and simplified.

VirtualInputSource / DataSource:

The VirtualInputSource interfaces abstract the DataSource symbols from the analysis. The FB DataSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs.

Network:

All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

Modules:

The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

ClosedNetwork Instance:

In this FB the corresponding ClosedNetwork is instantiated for the analysis. Internally used modules are no longer generated as module FBs in this case.

Results:

If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker in binary or Json format, or saves the data locally to an Analytics binary file.

Analysis:

The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

MAIN PRGs:

In the MAIN_Analytics program, the DataSource FBs are called, the reset function is managed and, if appropriate, the values are mapped with the HMI dashboard. The program is assigned to a separate task.

The FB Analysis is called in the programs MAIN_Analytics_Vx_Cx. The programs are each assigned to a separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.

All reset calls are defined in the action A_Reset.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics Version >= 3.1.0.0

6.1.2.3.1 FB_DataSource

The FB DataSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs.

Syntax

Definition:

```
FUNCTION_BLOCK FB_T[n]_DataSource IMPLEMENTS I_DataSource, I_V[n]_Virtual_Input_Source
VAR_OUTPUT
eDataState: E_DataSourceState;
END_VAR
```

Methods

Name	Definition location	Description
Call [▶ 80]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
GetData [▶ 79]	Local	Method to retrieve the data of the specified element.
GetDataOversampling [▶ 80]	Local	Method to retrieve the oversampling data of the specified element
NewDataAvailable [▶ 80]	Local	Method to check if new data is available.
HistoricalCtrl [▶ 80]	Local	Method for retrieving historical data
UpdateRecordList [▶ 81]	Local	Method for updating the historical record list.

Requirements

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

6.1.2.3.1.1 GetData

Syntax

```
METHOD GetData : BOOL
VAR_INPUT
    nElement : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nElement	UDINT	Element ID to obtain the specific sample

Return value

Name	Type	Description
GetData	BOOL	Is TRUE if a new element is selected

6.1.2.3.1.2 Call

Syntax

```
METHOD Call : BOOL
```

Return value

Name	Type	Description
Call	BOOL	

6.1.2.3.1.3 GetDataOversampling

Syntax

```
METHOD GetDataOversampling : BOOL
VAR_INPUT
nElement : UDINT;
nSample: : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nElement	UDINT	Element ID to obtain the specific sample element
nSample	UDINT	Sample ID to obtain the specific sample

Return value

Name	Type	Description
GetDataOversampling	BOOL	Is TRUE if a new element is selected

6.1.2.3.1.4 NewDataAvailable

Syntax

```
METHOD NewDataAvailable : BOOL
VAR_INPUT
    nLastDataHandle : ULINT;
END_VAR
```

Inputs

Name	Type	Description
nLastDataHandle	ULINT	Handle of the last fetched data packet

Return value

Name	Type	Description
NewDataAvailable	BOOL	Is TRUE if new data is available

6.1.2.3.1.5 HistoricalCtrl

Syntax

```
METHOD HistoricalCtrl : BOOL
VAR_INPUT
    stCtrl : REFERENCE TO ST_HMI_DataSourceCtrl;
```

```
    stHistStreamInfo : REFERENCE TO ST_HMI_DataSourceHist;
    stRecordInfo : REFERENCE TO ST_HMI_DataSourceHistRecordInfo;
END_VAR
```

Inputs

Name	Type	Description
stCtrl	ST_HMI_DataSourceCtrl	
stHistStreamInfo	ST_HMI_DataSourceHist	
stRecordInfo	ST_HMI_DataSourceHistRecordInfo	

Return value

Name	Type	Description
HistoricalCtrl	BOOL	

6.1.2.3.1.6 UpdateRecordList

Syntax

```
METHOD UpdateRecordList : BOOL
VAR_INPUT
    stCtrl : REFERENCE TO ST_HMI_DataSourceCtrl;
    stHistStreamInfo : REFERENCE TO ST_HMI_DataSourceHist;
    sStreamSystemID : GUID;
END_VAR
```

Inputs

Name	Type	Description
stCtrl	ST_HMI_DataSourceCtrl	
stHistStreamInfo	ST_HMI_DataSourceHist	
sStreamSystemID	GUID	

Return value

Name	Type	Description
UpdateRecordList	BOOL	

6.1.2.3.1.7 NextData

Syntax

```
METHOD NextData : BOOL
VAR_INPUT
END_VAR
```

6.1.2.3.1.8 NextDataOversample

Syntax

```
METHOD GetDataOversampling : BOOL
VAR_INPUT
nMaxOversampling : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nMaxOversampling	UDINT	Specifies the maximum oversampling factor.

Return value

Name	Type	Description
NextDataOversample	BOOL	Is TRUE if a new element is selected

6.1.2.3.2 FB_Network

All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

Syntax

Definition:

```
FUNCTION_BLOCK FB_N[n]_[Network1]
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
VAR
    [module FBs]
END_VAR
```

Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.

Methods

Name	Definition location	Description
Call [▶ 73]	Local	Method for background communication. The method must be called cyclically.
Reset [▶ 73]	Local	Resetting the network with all submodules.
ValueMapping [▶ 83]	Local	Assignment of the input values to the various module inputs.
SetHMIValues [▶ 83]	Local	Optional: Mapping of the inputs/outputs of the modules to the global HMI variable.

Requirements

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

6.1.2.3.2.1 ValueMapping

Syntax

```
METHOD ValueMapping : BOOL
VAR_INPUT
    pAnalysis : POINTER TO FB_Analysis;
END_VAR
```

Inputs

Name	Type	Description
pAnalysis	FB_Analysis	Instance of the analysis FB

Return value

Name	Type	Description
ValueMapping	BOOL	

6.1.2.3.2.2 SetHMIValues

Syntax

```
METHOD SetHMIValues : BOOL
VAR_INPUT
    pHMI_N[n]_[Network1] : POINTER TO ST_HMI_N[n]_[Network1];
END_VAR
```

Inputs

Name	Type	Description
pHMI_N[n]_[Network1]	POINTER TO ST_HMI_N[n]_[Network1]	Pointer to the global HMI structure

Return value

Name	Type	Description
SetHMIValues	BOOL	Is TRUE when completed

6.1.2.3.2.3 Call

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    pAnalysis: POINTER TO FB_Analysis;
END_VAR
```

Inputs

Name	Type	Description
pAnalysis	FB_Analysis	Instance of the analysis FB.

Return value

Name	Type	Description
Call	BOOL	

6.1.2.3.2.4 Reset

Syntax

```
METHOD Reset : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

6.1.2.3.3 FB_Module

The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

Syntax

Definition:

```
FUNCTION_BLOCK FB_N[n]_M[n]_[Module]
VAR_INPUT
    [module inputs]
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
    [module outputs]
END_VAR
```

Inputs

Name	Type	Description
Module inputs		Inputs of the selected module.

Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.
Module outputs		Outputs of the selected module.

Methods

Name	Definition location	Description
Call [▶ 85]	Local	Method for background communication. The method must be called cyclically.
Reset [▶ 85]	Local	Resetting the module.
SetHMI [▶ 85]	Local	Sets the inputs/outputs to the global HMI structures.
GetHMI [▶ 86]	Local	Optional: Sets the inputs of the global HMI structures to the inputs of the module

Requirements

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

6.1.2.3.3.1 Call

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    ipVirtual_Input_Source : I_V[n]_Virtual_Input_Source;
    [ipValueMapping_M2M : I_ValueMapping_M2M;]
END_VAR
```

Inputs

Name	Type	Description
ipVirtual_Input_Source	I_V[n]_Virtual_Input_Source	Data for the analysis
ipValueMapping_M2M	I_ValueMapping_M2M	Optional: Necessary for mapping values between modules

Return value

Name	Type	Description
Call	BOOL	

6.1.2.3.3.2 Reset

Syntax

```
METHOD Reset : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

6.1.2.3.3.3 SetHMI

Syntax

```
METHOD GetHMI : BOOL
VAR_INPUT
    nContent : DINT
    pContent : PVOID
    bHMIReinit : BOOL
END_VAR
```

```
VAR
    pHMI_C[n]_[Content] : POINTER TO ST_HMI_C[n]_[Content];
END_VAR
```

Inputs

Name	Type	Description
nContent	DINT	HMI Content Index
pContent	PVOID	Pointer to the HMI content structure
bHMIReinit	BOOL	Initialize the HMI content structure

Return value

Name	Type	Description
SetHMI	BOOL	Is TRUE when completed

6.1.2.3.3.4 GetHMI (4.1)

Syntax

```
METHOD GetHMI : BOOL
VAR_INPUT
    nContent :  DINT
    pContent :  PVOID
END_VAR
```

```
VAR
    pHMI_C[n]_[Content] : POINTER TO ST_HMI_C[n]_[Content];
END_VAR
```

Inputs

Name	Type	Description
nContent	DINT	HMI Content Index
pContent	PVOID	Pointer to the HMI content structure

Return value

Name	Type	Description
GetHMI	BOOL	Is TRUE when completed

6.1.2.3.4 FB_Analysis

The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Analysis
VAR_INPUT
    ipV[n]_VirtualInputs: I_V[n]_Virtual_Input_Source;
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
VAR
    [network FBs]
END_VAR
```

Inputs

Name	Type	Description
ipV[n]_VirtualInputs	I_V[n]_Virtual_Input_Source	Data for analysis from the specific data source

Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.

Methods

Name	Definition location	Description
Call [▶ 87]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
Reset [▶ 87]	Local	Reset the whole analysis
ResultStream [▶ 88]	Local	Optional: If a result stream has to be created

Requirements

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

6.1.2.3.4.1 Call

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    ipVirtual_Input_Source: I_V[n]_Virtual_Input_Source;
END_VAR
```

Inputs

Name	Type	Description
ipVirtual_Input_Source	I_V[n]_Virtual_Input_Source	Data for the analysis.

Return value

Name	Type	Description
Call	BOOL	

6.1.2.3.4.2 Reset

Syntax

```
METHOD Reset : BOOL
VAR_IN_OUT
    stReset: ST_AnalysisReset;
END_VAR
```

Inputs

Name	Type	Description
stReset	ST_AnalysisReset	Structure to define which module or network is to be reset.

Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

6.1.2.3.4.3 ResultStream

Syntax

```
METHOD ResultStream : BOOL
VAR_INPUT
    ipResults: I_Results;
END_VAR
```

Inputs

Name	Type	Description
ipResults	I_Results	Interface pointer to the FB Results

Return value

Name	Type	Description
Call	BOOL	

6.1.2.3.5 FB_Results

If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker or saves the data to the Analytics binary file.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Results
VAR_OUTPUT
    stResults: ST_Results;
END_VAR
```

Outputs

Name	Type	Description
stResults	ST_Results	Result structure that contains all elements of the result stream.

Methods

Name	Definition location	Description
Call [▶ 89]	Local	Method for background communication. The method must be called cyclically.
AddResult [▶ 89]	Local	Add a sample to the result stream
SendResults [▶ 89]	Local	Sends all buffered samples of the result stream
Release [▶ 90]	Local	Close stream or file of the result stream

Requirements

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

6.1.2.3.5.1 Call

Syntax

```
METHOD Call : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	

6.1.2.3.5.2 AddResult

Syntax

```
METHOD AddResult : BOOL
VAR_INPUT
    tTimestamp: ULINT;
    stSample: ST_Results;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Timestamp of the sample
stSample	ST_Results	Sample structure

Return value

Name	Type	Description
AddResult	BOOL	

6.1.2.3.5.3 SendResults

Syntax

```
METHOD SendResults : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
SendResults	BOOL	

6.1.2.3.5.4 Release

Syntax

```
METHOD Release : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Release	BOOL	

6.1.2.3.6 MAIN_Analytics

In the MAIN_Analytics program, the DataSource FBs are called, the reset function is managed and, if appropriate, the values are mapped with the HMI Dashboard. The program is assigned to a separate task.

Syntax

Definition:

```
PROGRAM MAIN_Analytics
VAR_INPUT
    sCurrentStreamSystemID: GUID;
    stHistStreamInfo: REFERENCE TO ST_HMI_DataSourceHist;
    stHistRecordInfo: REFERENCE TO ST_HMI_DataSourceHistRecordInfo;
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
```

Requirements

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

6.1.2.3.7 MAIN_Analytics_C[n]

The FB Analysis is called in the program MAIN_Analytics_C[n]. The program is assigned to the separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.

All reset calls are defined in the action A_Reset.

Mapping of the HMI values takes place in the action A_MapToHMI.

Syntax

Definition:

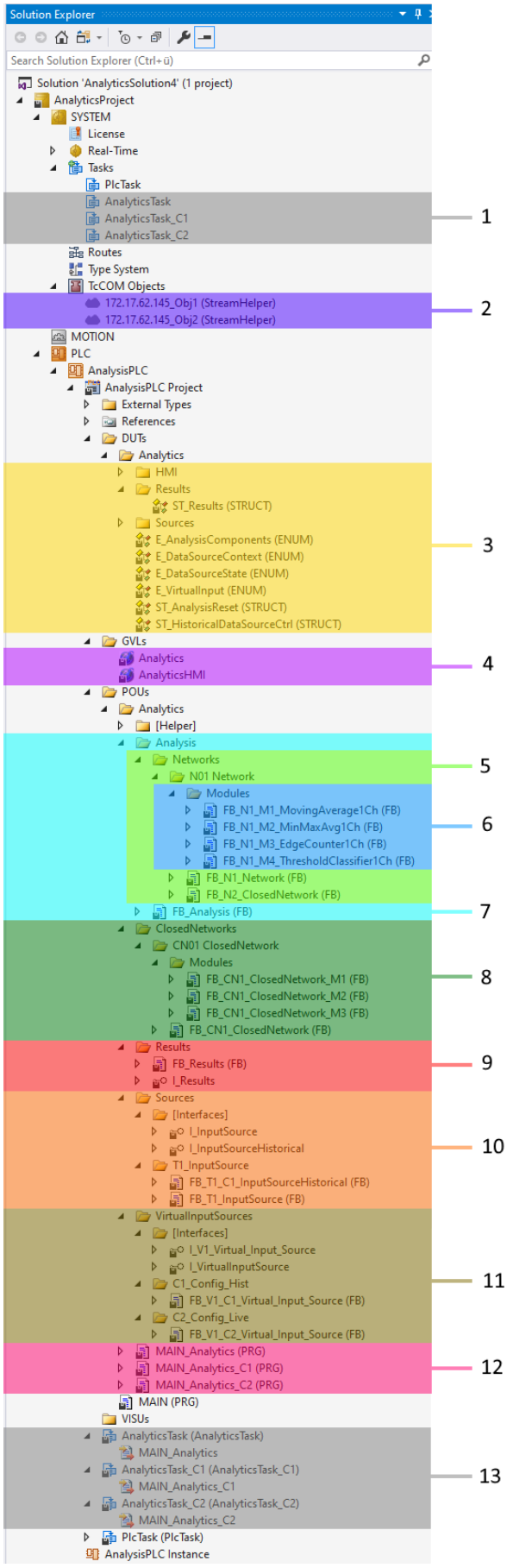
```
PROGRAM MAIN_Analytics_C[n]
VAR_INPUT
    stReset: ST_AnalysisReset;
END_VAR
VAR_OUTPUT
```

```
    bError: BOOL;
    ipTcResult: I_TcMessage;
    nLastDataHandle: ULINT;
END_VAR
```

Requirements

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

6.1.2.4 Code version 5.0



1: Tasks

A separate task is created for the analytics analysis and for each configuration of a Virtual Input Source.

2: StreamHelper

For each data source of type MQTT binary stream, the code generation creates an instance of a stream helper object to process the incoming binary stream patterns.

3: DataTypes

The data types are created for the analysis. They contain STRUCTs for the reset function or result processing and ENUMs to select the various components.

4: GVLs

To conveniently map module inputs and outputs with the HMI dashboard, selected variables are generated as global variables. In addition, the Data Source function block instances and various parameters are generated as global variables.

5: Network

All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

6: Modules

The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

7: Analysis

The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

8: ClosedNetwork

The ClosedNetwork FBs are generated once with all subnetworks and modules. They can be instantiated multiple times in the analysis. In this way, the generated code can be reduced and simplified.

9: Results

If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker in binary or Json format, or saves the data locally to an Analytics binary file.

10: InputSource

The FB InputSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs.

11: VirtualInputSource

The VirtualInputSource interfaces abstract the InputSource symbols from the analysis.

12: MAIN PRGs

In the MAIN_Analytics program, the DataSource FBs are called, the reset function is managed and, if appropriate, the values are mapped with the HMI dashboard. The program is assigned to a separate task.

In the programs MAIN_Analytics_Vx_Cx the FB-Analysis is called. The programs are each assigned to a separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.

All reset calls are defined in the action A_Reset.

Requirements

Development environment	Target platform	Plc libraries to include
TwinCAT v3.1.4024.0	PC or CX (x64, x86)	Tc3_Analytics Version >= 3.1.0.0

6.1.2.4.1 FB_Analysis

The entire analysis routine is defined in the FB Analysis. All configured networks with their modules and error handling are created.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Analysis
VAR_INPUT
    nConfigurationID: INT;
    ipV[n]_VirtualInputs: I_V[n]_Virtual_Input_Source;
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
VAR
    [network FBs]
END_VAR
```

Inputs

Name	Type	Description
nConfigurationID	INT	Configuration index
ipV[n]_VirtualInputs	I_V[n]_Virtual_Input_Source	Data for analysis from the specific data source

Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.

Properties

Name	Type	Access	Description
nContext	DWORD	Get	Context index

Methods

Name	Definition location	Description
Call [▶ 95]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
Reset [▶ 95]	Local	Resets the entire analysis.
ResultStream [▶ 96]	Local	Optional: If a result stream needs to be created.
SetHMIValues [▶ 96]	Local	Method for filling the HMI structures
GetHMIValues [▶ 96]	Local	Method for setting the parameters from the HMI into the analysis

Requirements

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

6.1.2.4.1.1 Call

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    ipVirtual_Input_Source: I_V[n]_Virtual_Input_Source;
END_VAR
```

Inputs

Name	Type	Description
ipVirtual_Input_Source	I_V[n]_Virtual_Input_Source	Data for the analysis.

Return value

Name	Type	Description
Call	BOOL	

6.1.2.4.1.2 Reset

Syntax

```
METHOD Reset : BOOL
VAR_IN_OUT
    stReset: ST_AnalysisReset;
END_VAR
```

Inputs

Name	Type	Description
stReset	ST_AnalysisReset	Structure to define which module or network is to be reset.

Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

6.1.2.4.1.3 ResultStream

Syntax

```
METHOD ResultStream : BOOL
VAR_INPUT
    ipResults: I_Results;
END_VAR
```

Inputs

Name	Type	Description
ipResults	I_Results	Interface pointer to the FB results

Return value

Name	Type	Description
Call	BOOL	

6.1.2.4.1.4 SetHMIValues

Syntax

```
METHOD SetHMIValues : BOOL
VAR_INPUT
    pHMI_N[n]_[Network1] : POINTER TO ST_HMI_N[n]_[Network1];
END_VAR
```

Inputs

Name	Type	Description
pHMI_N[n]_[Network1]	POINTER TO ST_HMI_N[n]_[Network1]	Pointer to the global HMI structure

Return value

Name	Type	Description
SetHMIValues	BOOL	Is TRUE when completed

6.1.2.4.1.5 GetHMIValues

Syntax

```
METHOD GetHMIValues : BOOL
VAR_INPUT
    pHMI_N[n]_[Network1] : POINTER TO ST_HMI_N[n]_[Network1];
END_VAR
```

Inputs

Name	Type	Description
pHMI_N[n]_[Network1]	POINTER TO ST_HMI_N[n]_[Network1]	Pointer to the global HMI structure

Return value

Name	Type	Description
SetHMIValues	BOOL	Is TRUE when completed

6.1.2.4.2 FB_Network

All modules are sorted in a specific network to achieve a better overview and structure of the configured analysis.

Syntax

Definition:

```
FUNCTION_BLOCK FB_N[n]_[Network1]
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
END_VAR
VAR
    [module FBs]
END_VAR
```

Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.

Methods

Name	Definition location	Description
Call [▶ 73]	Local	Method for background communication. The method must be called cyclically.
Reset [▶ 73]	Local	Resetting the network with all submodules.
ValueMapping [▶ 98]	Local	Assignment of the input values to the various module inputs.

Requirements

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

6.1.2.4.2.1 Call

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    pAnalysis: POINTER TO FB_Analysis;
END_VAR
```

Inputs

Name	Type	Description
pAnalysis	FB_Analysis	Instance of the analysis FB.

Return value

Name	Type	Description
Call	BOOL	

6.1.2.4.2.2 Reset

Syntax

```
METHOD Reset : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

6.1.2.4.2.3 ValueMapping

Syntax

```
METHOD ValueMapping : BOOL
VAR_INPUT
    pAnalysis : POINTER TO FB_Analysis;
END_VAR
```

Inputs

Name	Type	Description
pAnalysis	FB_Analysis	Instance of the analysis FB

Return value

Name	Type	Description
ValueMapping	BOOL	

6.1.2.4.3 FB_Module

The module FBs contain all inputs and outputs of the configured modules from the workbench configurator. It is also possible to reconfigure the modules during runtime. To do this, simply change the parameter and then start the reconfiguration process with a rising edge at INPUT bReconfigure.

Syntax

Definition:

```
FUNCTION_BLOCK FB_N[n]_M[n]_[Module]
VAR_INPUT
    [module inputs]
END_VAR
VAR_INPUT PERSISTENT
    [module persistent parameter inputs]
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
    [module outputs]
END_VAR
```

Inputs

Name	Type	Description
Module inputs		Inputs of the selected module.

Outputs

Name	Type	Description
bError	BOOL	Becomes TRUE when an error situation occurs.
ipTcResult	I_TcMessage	Message interface of the TwinCAT 3 EventLogger, which provides further information about the return value.
Module outputs		Outputs of the selected module.

Properties

Name	Type	Access	Description
nContext	DWORD	Get	Context index

Methods

Name	Definition location	Description
Call [▶ 99]	Local	Method for background communication. The method must be called cyclically.
Reset [▶ 100]	Local	Resetting the module.
SetHMI [▶ 100]	Local	Sets the inputs/outputs to the global HMI structures.
GetHMI [▶ 100]	Local	Optional: Sets the inputs of the global HMI structures to the inputs of the module

Requirements

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

6.1.2.4.3.1 Call

Syntax

```
METHOD Call : BOOL
VAR_INPUT
    ipVirtual_Input_Source : I_V[n]_Virtual_Input_Source;
    [ipValueMapping_M2M : I_ValueMapping_M2M;]
END_VAR
```

Inputs

Name	Type	Description
ipVirtual_Input_Source	I_V[n]_Virtual_Input_Source	Data for the analysis
ipValueMapping_M2M	I_ValueMapping_M2M	Optional: Necessary for mapping values between modules

Return value

Name	Type	Description
Call	BOOL	

6.1.2.4.3.2 Reset

Syntax

```
METHOD Reset : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Reset	BOOL	Is TRUE when completed.

6.1.2.4.3.3 SetHMI

Syntax

```
METHOD SetHMI : BOOL
VAR_INPUT
    nContent : DINT
    pContent : PVOID
    bHMIReinit : BOOL
END_VAR
```

```
VAR
    pHMI_C[n]_[Content] : POINTER TO ST_HMI_C[n]_[Content];
END_VAR
```

Inputs

Name	Type	Description
nContent	DINT	HMI Content Index
pContent	PVOID	Pointer to the HMI content structure
bHMIReinit	BOOL	Initialize the HMI content structure

Return value

Name	Type	Description
SetHMI	BOOL	Is TRUE when completed

6.1.2.4.3.4 GetHMI

Syntax

```
METHOD GetHMI : BOOL
VAR_INPUT
    nContent :  DINT
    pContent :  PVOID
END_VAR
```

```
VAR
    pHMI_C[n]_[Content] : POINTER TO ST_HMI_C[n]_[Content];
END_VAR
```

Inputs

Name	Type	Description
nContent	DINT	HMI Content Index
pContent	PVOID	Pointer to the HMI content structure

Return value

Name	Type	Description
GetHMI	BOOL	Is TRUE when completed

6.1.2.4.4 FB_InputSource

The FB DataSource manages the receipt of input values from the various sources. In the OUTPUT declaration you will find all configured inputs.

Syntax

Definition:

```
FUNCTION_BLOCK FB_T[n]_InputSource IMPLEMENTS I_InputSource
VAR
END_VAR
```

Interfaces

Type	Description
I_InputSource	Interface for communication with a data source

Methods

Name	Definition location	Description
Call [▶ 101]	Local	Method for background communication with the TwinCAT driver. The method must be called cyclically.
GetData [▶ 102]	Local	Method to retrieve the data of the specified element.
NewDataAvailable [▶ 102]	Local	Method to check if new data is available.
AddClient [▶ 103]	Local	Method for adding data Clients
ClientDone [▶ 103]	Local	Method to signal that the client has received all data.

Properties

Name	Type	Access	Description
bReadNewData	BOOL	Get	.
eDataState	E_DataSourceState	Get	
nDataHandle	ULINT	Get	
nElements	UDINT	Get	
nMaxOversamplingFactor	UDINT	Get	

Requirements

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

6.1.2.4.4.1 Call

Syntax

```
METHOD Call : BOOL
```

Return value

Name	Type	Description
Call	BOOL	

6.1.2.4.4.2 GetData

Syntax

```
METHOD GetData : BOOL
VAR_INPUT
    nElement    : UDINT;
    pInputs     : PVOID;
    nInputsSize : UDINT
END_VAR
VAR_OUTPUT
    nTimestamp  : ULINT;
    nContext    : DWORD;
END_VAR
```

Inputs

Name	Type	Description
nElement	UDINT	Element ID to obtain the specific sample
pInputs	PVOID	Pointer to the data structure
nInputsSize	UDINT	Size of the data structure

Outputs

Name	Type	Description
nTimestamp	ULINT	Timestamp data
nContext	DWORD	Data context

Return value

Name	Type	Description
GetData	BOOL	Is TRUE if a new element is selected

6.1.2.4.4.3 NewDataAvailable

Syntax

```
METHOD NewDataAvailable : BOOL
VAR_INPUT
    nLastDataHandle : ULINT;
END_VAR
```

Inputs

Name	Type	Description
nLastDataHandle	ULINT	Handle of the last fetched data packet

Return value

Name	Type	Description
NewDataAvailable	BOOL	Is TRUE if new data is available

6.1.2.4.4.4 AddClient

Syntax

```
METHOD AddClient : BOOL
VAR_OUTPUT
    nClientID    : DWORD;
END_VAR
```

Outputs

Name	Type	Description
nClientID	LWORD	Client-ID

6.1.2.4.4.5 ClientDone

Syntax

```
METHOD ClientDone : BOOL
VAR_OUTPUT
    nClientID    : DWORD;
END_VAR
```

Outputs

Name	Type	Description
nClientID	LWORD	Client-ID

6.1.2.4.5 FB_VirtualInputSource

The FB VirtualInputSource abstracts the InputSources for the different analysis configurations. The virtual inputs configured in the Analytics Workbench are set here.

Syntax

Definition:

```
FUNCTION_BLOCK FB_V[n]_C[n]_VirtualInputSource IMPLEMENTS I_VirtualInputSource,
I_V[n]_VirtualInputSource
VAR
END_VAR
```

Interfaces

Type	Description
I_VirtualInputSource	Interface for communication with a data source
I_V[n]_VirtualInputSource	Interface that provides all defined virtual inputs

Methods

Name	Definition location	Description
SourceSync [▶ 104]	Local	Method for synchronizing multiple FB instances
NextData [▶ 104]	Local	Method to accept the next data set
Done [▶ 104]	Local	Method of signaling that the entire data packet has been processed.

Properties

Name	Type	Access	Description
bEndOfData	BOOL	Get	Signals the end of the data packet
dtTimestamp	DCTIMESTRUCT	Get	Timestamp of the currently accepted data set
nDataHandle	ULINT	Get	Data handle
nContext	DWORD	Get	Data context
tTimestamp	ULINT	Get	Timestamp of the currently received data set

Requirements

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

6.1.2.4.5.1 SourceSync

Syntax

```
METHOD SourceSync : BOOL
```

6.1.2.4.5.2 NextData

Syntax

```
METHOD NextData : BOOL
VAR_INPUT
    nMaxOversampling    : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nMaxOversampling	UDINT	Specifies the maximum oversampling value

Return value

Name	Type	Description
NextData	BOOL	Is TRUE if a new element is selected

6.1.2.4.5.3 Done

Syntax

```
METHOD Done : BOOL
VAR_INPUT
END_VAR
```

6.1.2.4.6 FB_Results

If analysis results need to be saved or streamed, the FB Results manages this and streams the selected variables to the message broker or saves the data to the Analytics binary file.

Syntax

Definition:

```
FUNCTION_BLOCK FB_Results
VAR_OUTPUT
    nTimestamp: ULINT;
    stResults: ST_Results;
END_VAR
```

Outputs

Name	Type	Description
nTimestamp	ULINT	Associated timestamp of the result structure data
stResults	ST_Results	Result structure that contains all elements of the result stream.

Properties

Name	Type	Access	Description
bInitialized	BOOL	Get	Indicates whether the function block has been properly initialized
nMaxSamples	INT	Get	Maximum number of buffered results
nResultCount	INT	Get	Current number of buffered results

Methods

Name	Definition location	Description
Call [▶ 105]	Local	Method for background communication. The method must be called cyclically.
AddResult [▶ 105]	Local	Add a sample to the result stream
SendResults [▶ 106]	Local	Sends all buffered samples of the result stream
Release [▶ 106]	Local	Close stream or file of the result stream

Requirements

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

6.1.2.4.6.1 Call

Syntax

```
METHOD Call : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Call	BOOL	

6.1.2.4.6.2 AddResult

Syntax

```
METHOD AddResult : BOOL
VAR_INPUT
    tTimestamp: ULINT;
    stSample: ST_Results;
END_VAR
```

Inputs

Name	Type	Description
tTimestamp	ULINT	Timestamp of the sample
stSample	ST_Results	Sample structure

Return value

Name	Type	Description
AddResult	BOOL	

6.1.2.4.6.3 SendResults

Syntax

```
METHOD SendResults : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
SendResults	BOOL	

6.1.2.4.6.4 Release

Syntax

```
METHOD Release : BOOL
VAR
END_VAR
```

Return value

Name	Type	Description
Release	BOOL	

6.1.2.4.7 MAIN_Analytics

In the MAIN_Analytics program, the InputSource FBs are called, the reset function is managed and, if necessary, the values are mapped with the HMI dashboard. The program is assigned to a separate task.

Syntax

Definition:

```
PROGRAM MAIN_Analytics
VAR
END_VAR
```

Requirements

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

6.1.2.4.8 MAIN_Analytics_C[n]

The FB Analysis is called in the program MAIN_Analytics_C[n]. The program is assigned to the separate task.

It is also possible to reset single modules, whole networks or all defined networks with only one rising flag. First, select the component to be reset. Then a rising edge at INPUT bReset starts the reset process.

All reset calls are defined in the action A_Reset.

The handling of the "InputSources" is done in the action A_InputSources.

The mapping of the HMI values is done in the action A_MapToHMI.

Syntax

Definition:

```
PROGRAM MAIN_Analytics_C[n]
VAR_INPUT
    stReset: ST_AnalysisReset;
END_VAR
VAR_OUTPUT
    bError: BOOL;
    ipTcResult: I_TcMessage;
    nAnalysisResultsTimestamp: ULINT;
    stAnalysisResults: ST_Results;
END_VAR
```

Requirements

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

6.2 HMI One-Click Dashboard

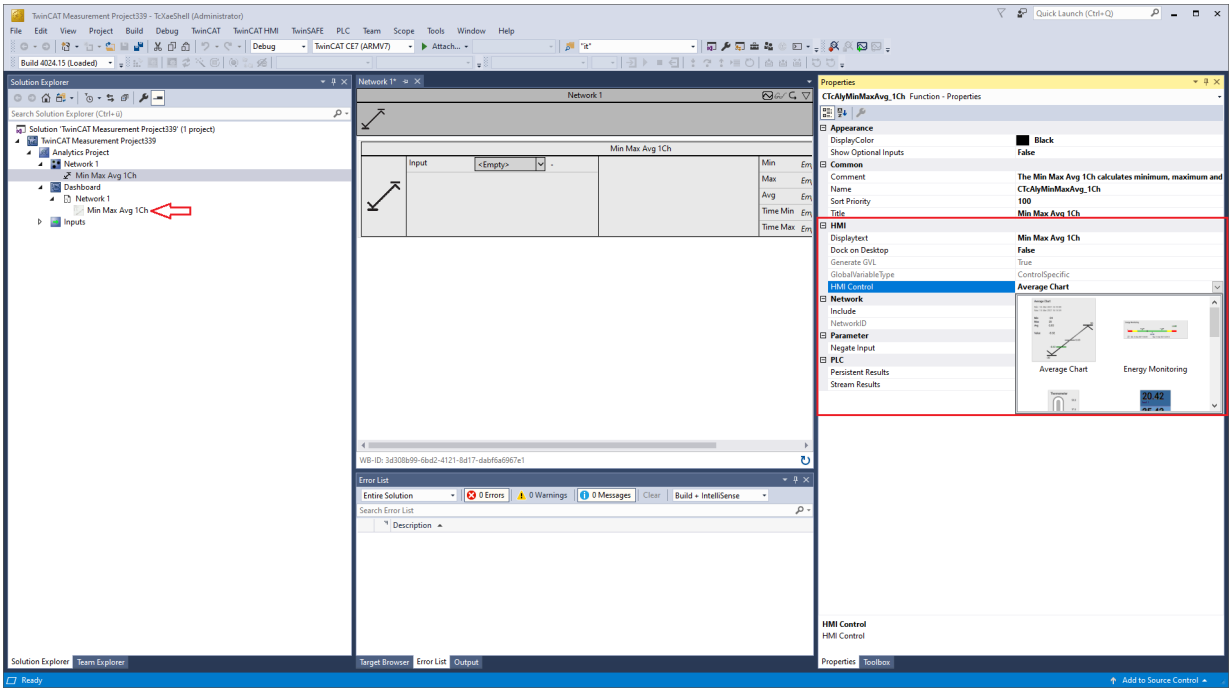
It is possible to automatically generate an HMI dashboard with HMI Controls for all modules and parameters configured in the TwinCAT Analytics Workbench Configurator. The HMI Dashboard is based on the TwinCAT HMI and visualizes the PLC data from the runtime deployment [▶ 43].

The automatically generated One-Click Dashboard is only available with the new HMI version 1.12. An Analytics Runtime license is required in order to use the Analytics HMI Controls.

✓ After configuring your Analytics Workbench project, an HMI Control can be selected for each algorithm.

1. Open the **Properties** window of the module and select an **HMI Control** from the dropdown list. You can change the display text for the title in the HMI dashboard (display text). You can also choose whether the control should be docked to the start page (Dock on Desktop). In the **Solution Explorer** all the controls

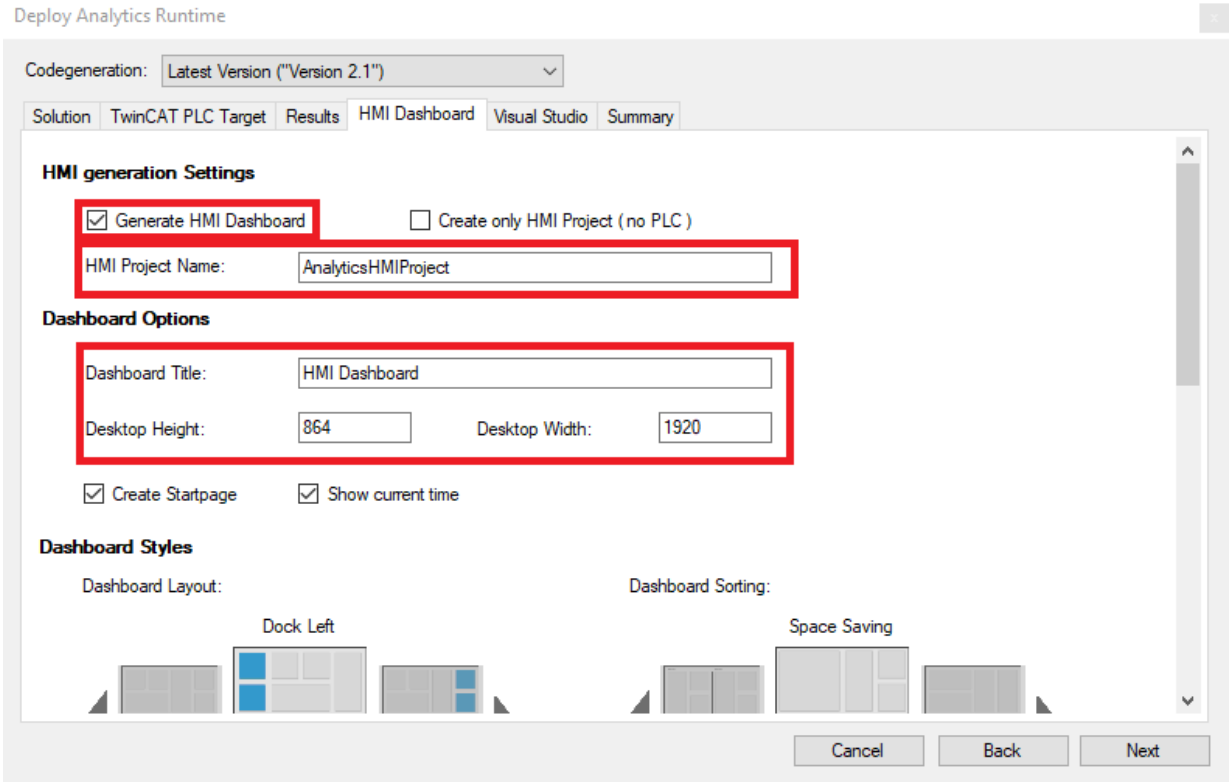
to be generated are stored under the Manage dashboard structure and content in Analytics project [▶ 110]



2. After completion of the configuration, click the **Deploy Analytics Runtime** command in the context menu. The Deploy Wizard starts and it is possible to set up the entire required configuration step by step for use.

   ⇨ You can configure your HMI Dashboard on the **HMI Dashboard** tab.

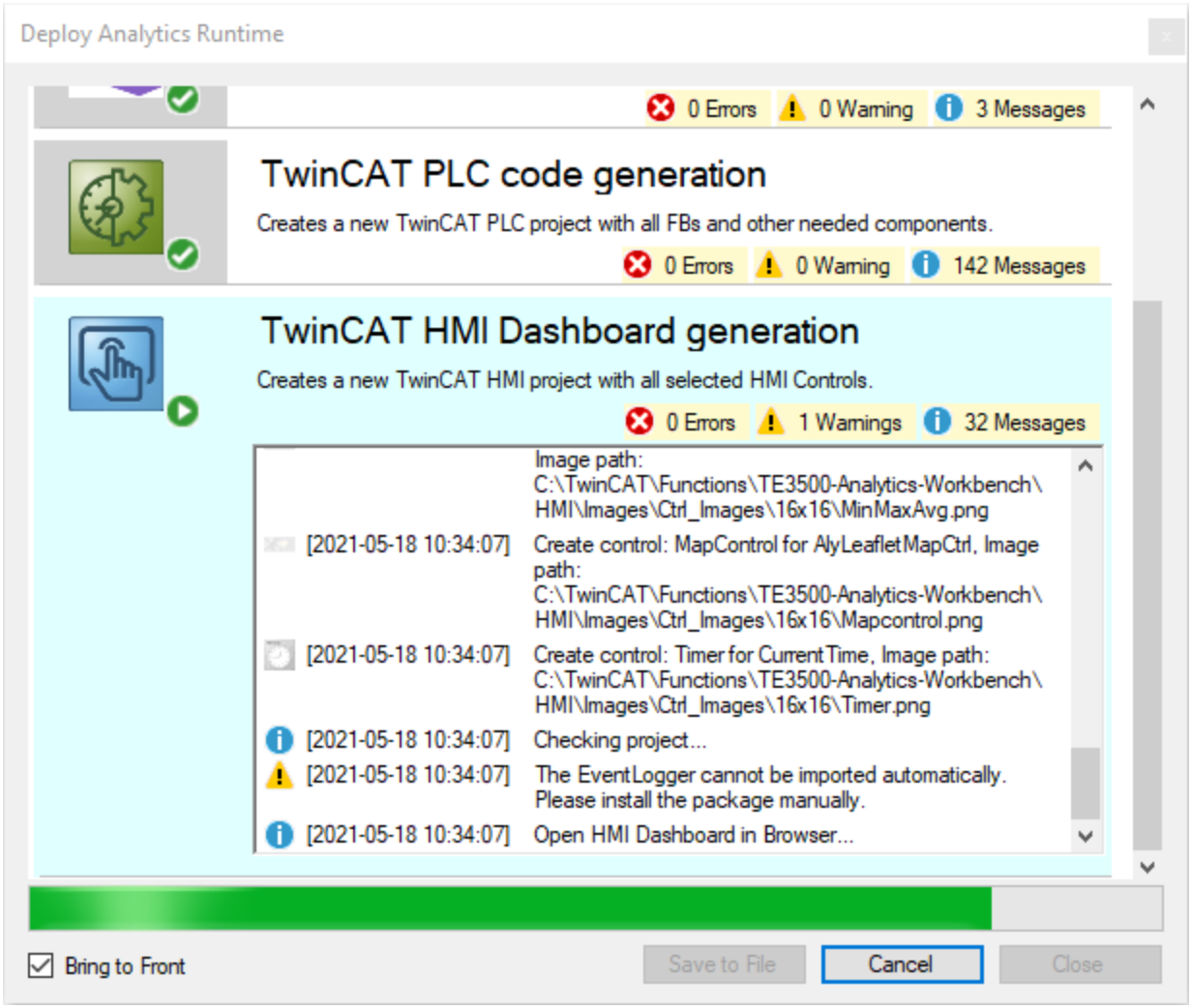
3. Activate the **Generate HMI Dashboard** checkbox. It is also possible to create only one HMI project without a PLC. Furthermore, you can also assign an **HMI Project Name** to the dashboard and set a Dashboard Title as well as the Desktop Height and Desktop Width in order to generate a tailor-made dashboard that suits your needs. The remaining configurations are explained in Dashboard Configuration [▶ 142].



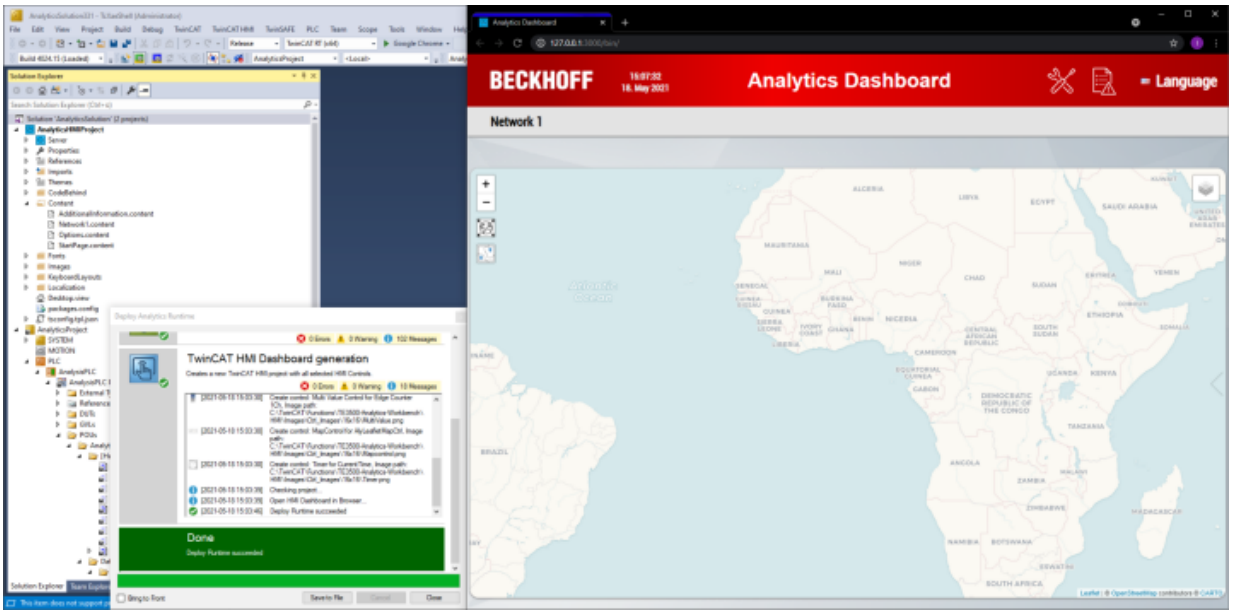
⇨ As usual, the last tab shows you all the settings you have made for the generation.

4. Now you can start the generation process by clicking **Deploy**.

⇨ The HMI generation begins immediately after runtime deployment (if selected). Each step for generating the HMI dashboard is also displayed in the overview window during the generation process.

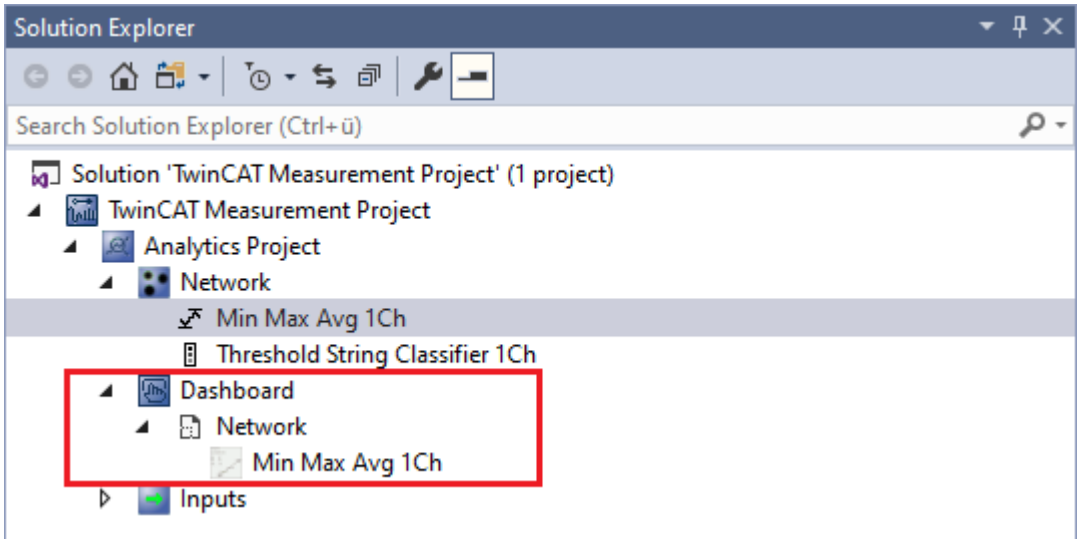


⇨ The dashboard opens automatically in your default browser.

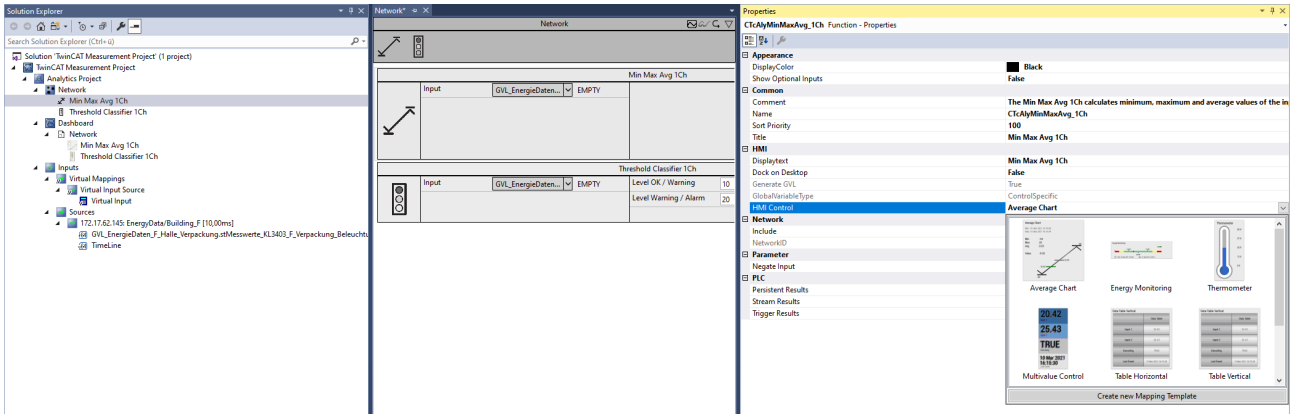


6.2.1 Manage dashboard structure and content in Analytics project

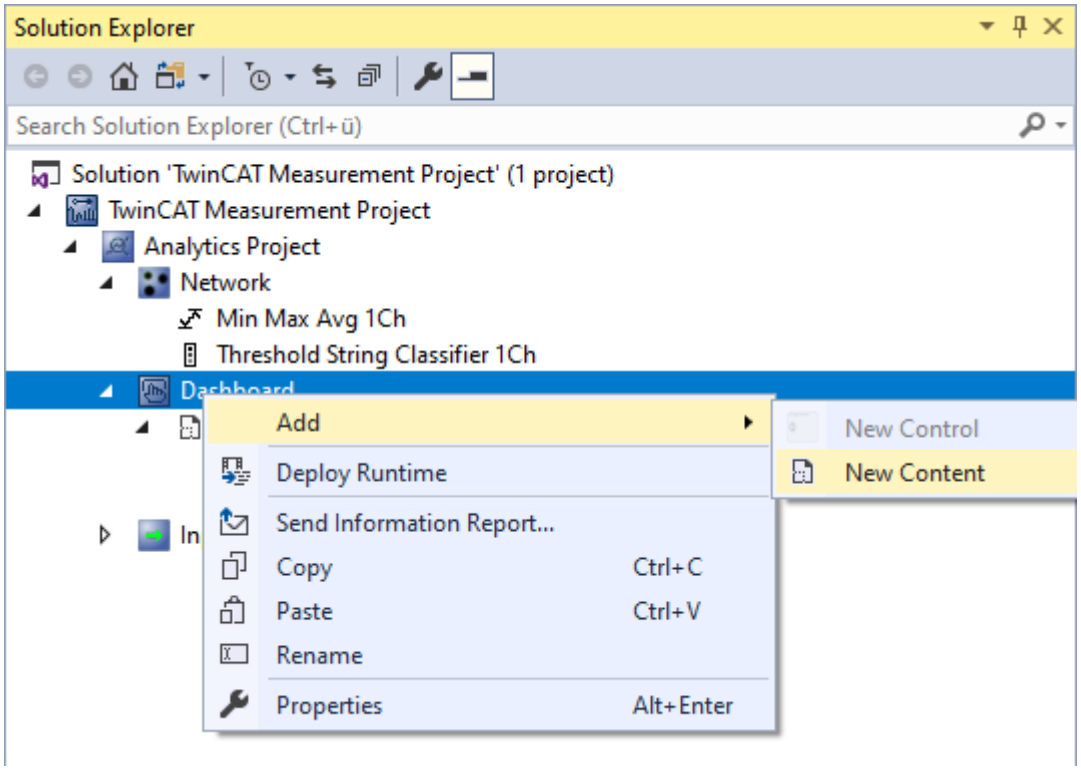
An Analytics project has a configuration of HMI contents (pages) and HMI controls (display elements) that are created during dashboard generation. This configuration can be viewed and changed via the **Dashboard node** in the Solution Explorer. You can rename the contents and controls at any time, move them to other contents via drag & drop, copy them (**Ctrl-C, Ctrl-V**) or delete them. A control can also be edited by double-clicking on it.



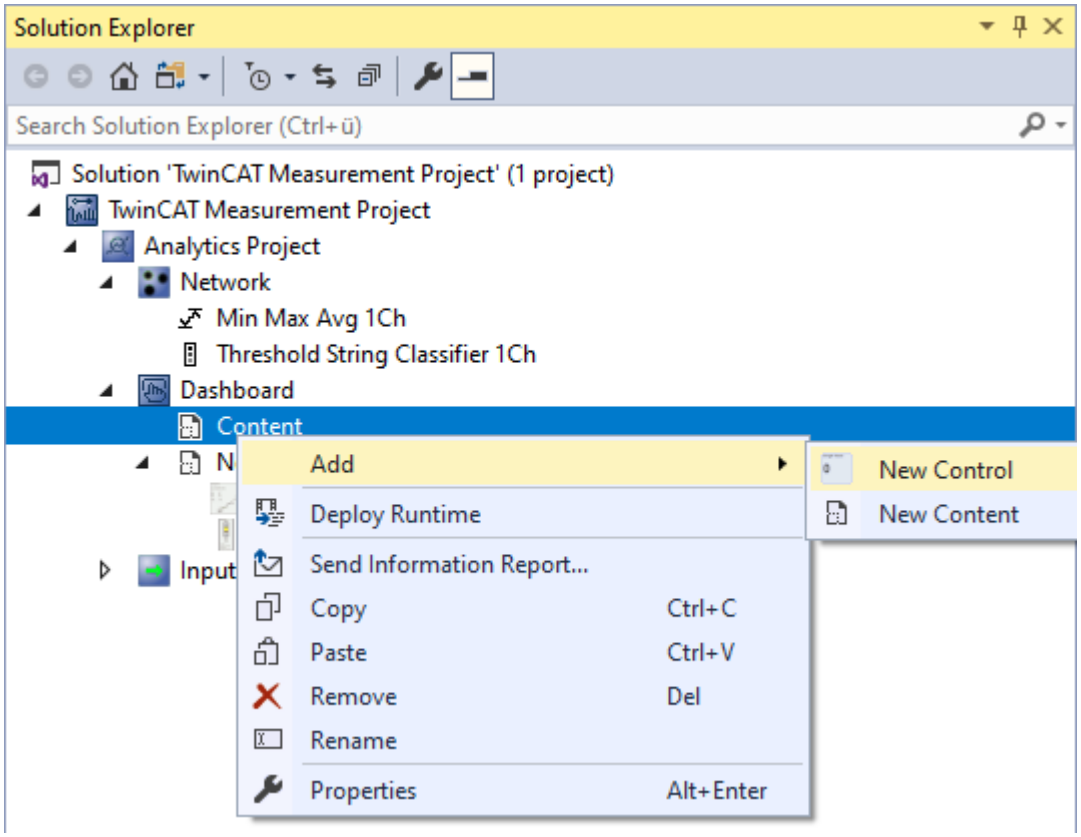
Each Analytics module has existing control mappings, which can be selected via the window **Properties** (you can also create/edit these yourself via "Create new Mapping Template" or via the Use customized and own controls [▶ 120] (point 2)). Once a control is selected for a module, it is listed under the **Dashboard** node.



You can create new content for the **Dashboard** and **Content** nodes by right-clicking.



Likewise, you can add **new controls** by right-clicking on a **Content** node.



This opens the **Analytics Dashboard Wizard**. This wizard guides you step-by-step through the configuration for adding a control.

Analytics Dashboard Wizard

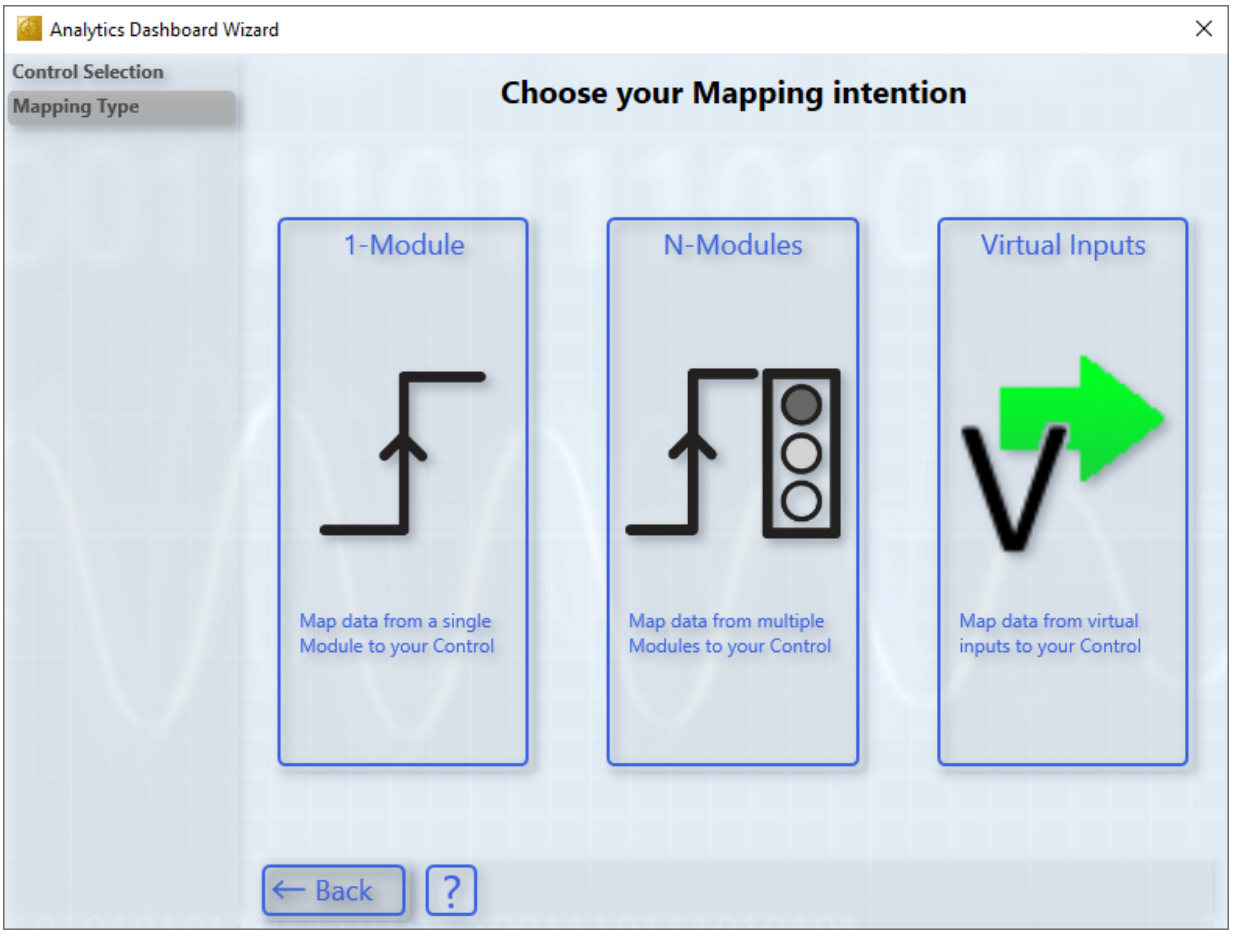
Familiarize yourself with the wizard by adding a control (select a control from the Properties window of the module). You can then look up the entire configuration in the wizard by double-clicking on the control. There you can follow the individual configuration steps.

Analytics Dashboard Wizard

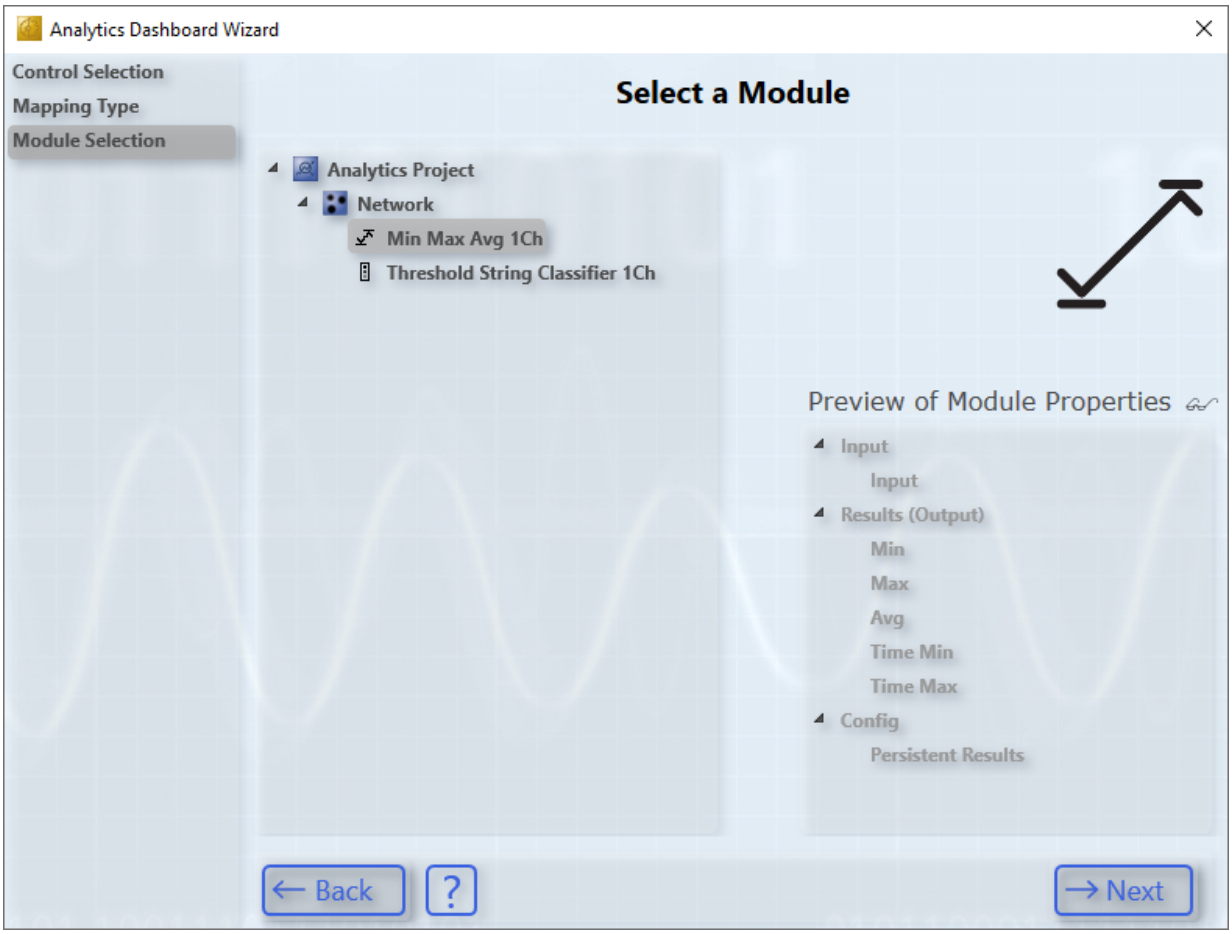
Select a control. All available controls are listed on the left. By default, only controls that are not algorithm-specific are listed. All controls can be made available by unchecking the **Show only default Controls** checkbox.

1. Select **Single Value** to display a single value in the dashboard. Click the **Next** button to continue.

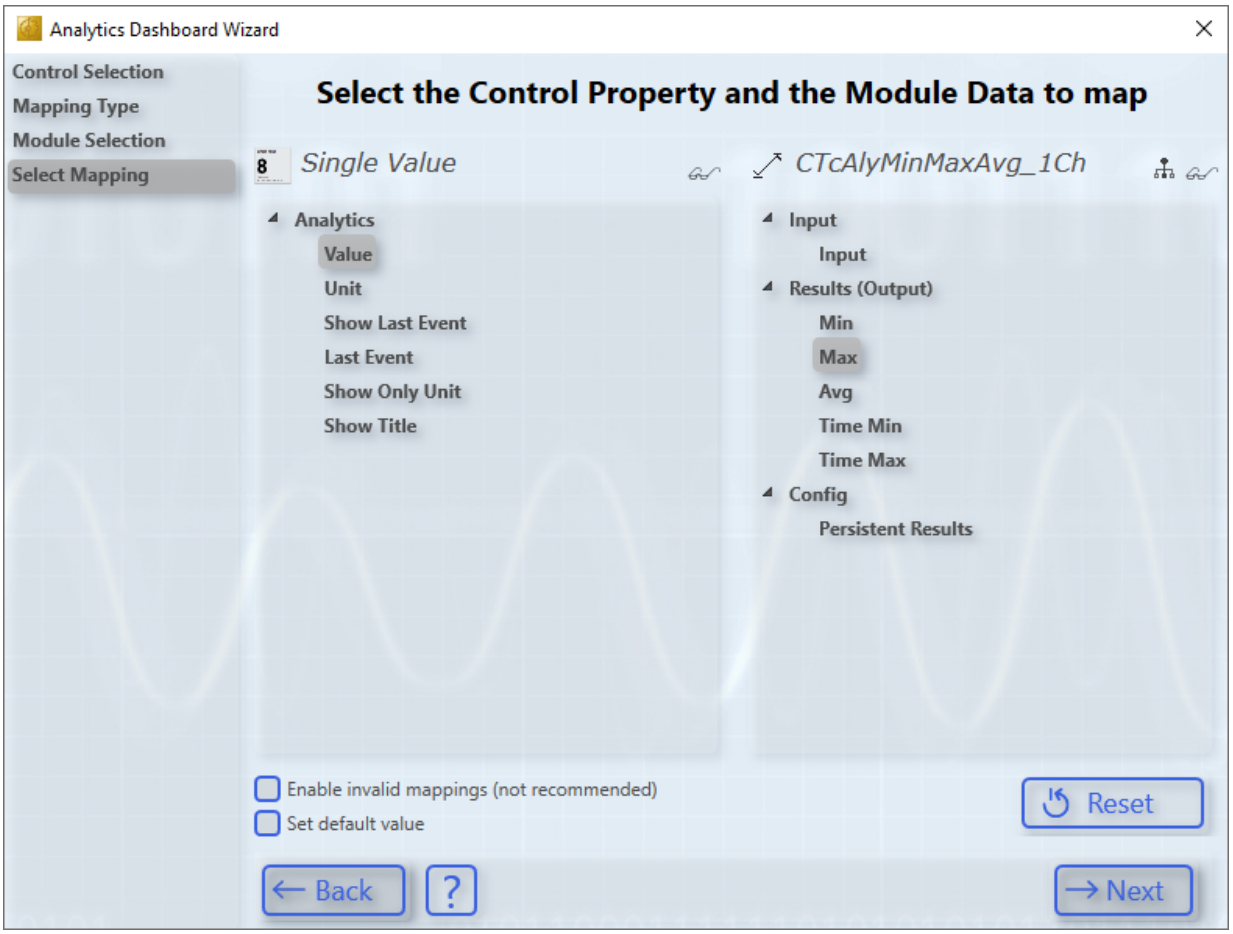
2. You can now link data from one module (**1-Module**), several modules (**N-Modules**) or from virtual inputs (**Virtual Inputs**). For this example, select **1-Module**.



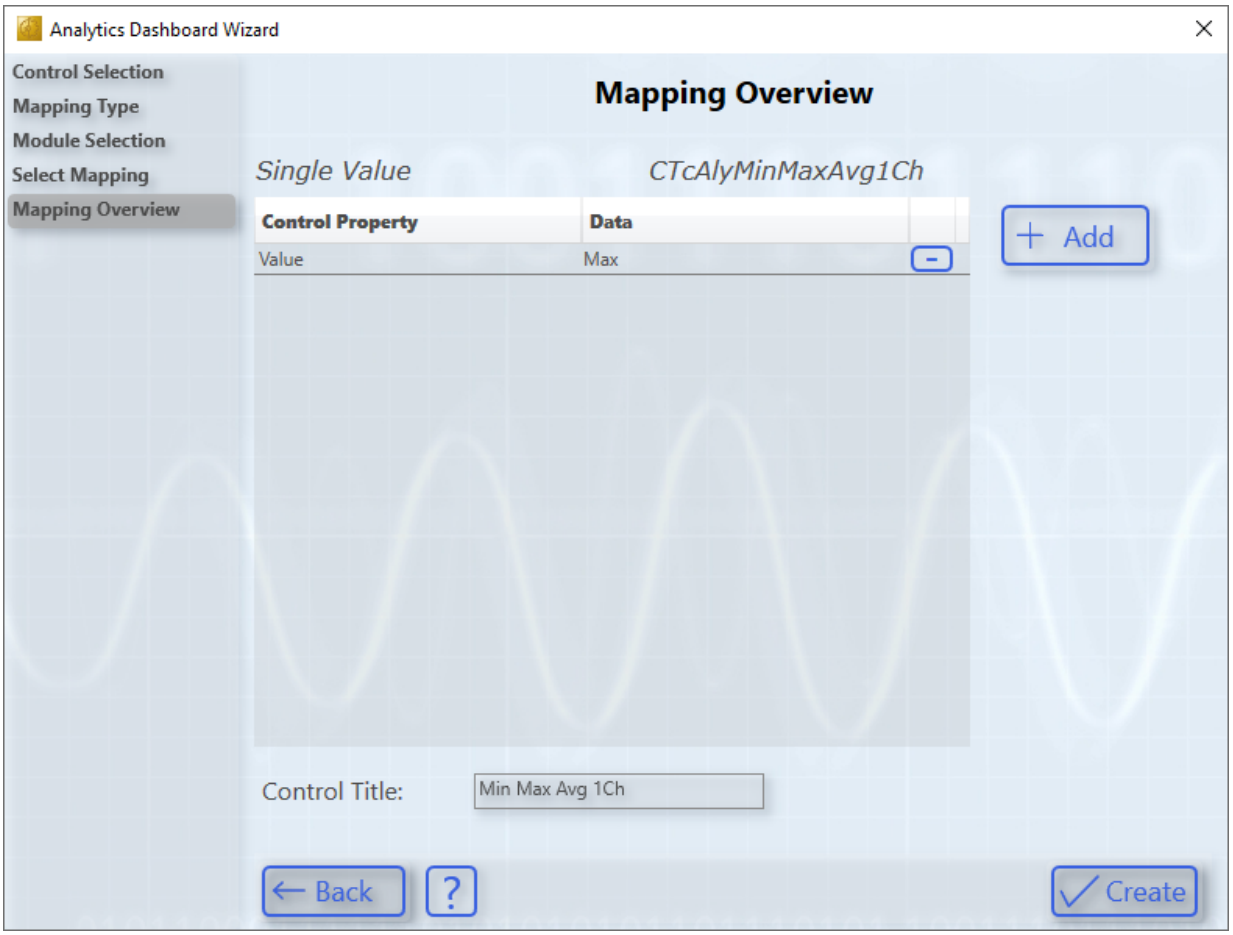
3. All modules from the Analytics project are listed. Select the module from which you want to display the data. A preview of the selected module is available on the right.



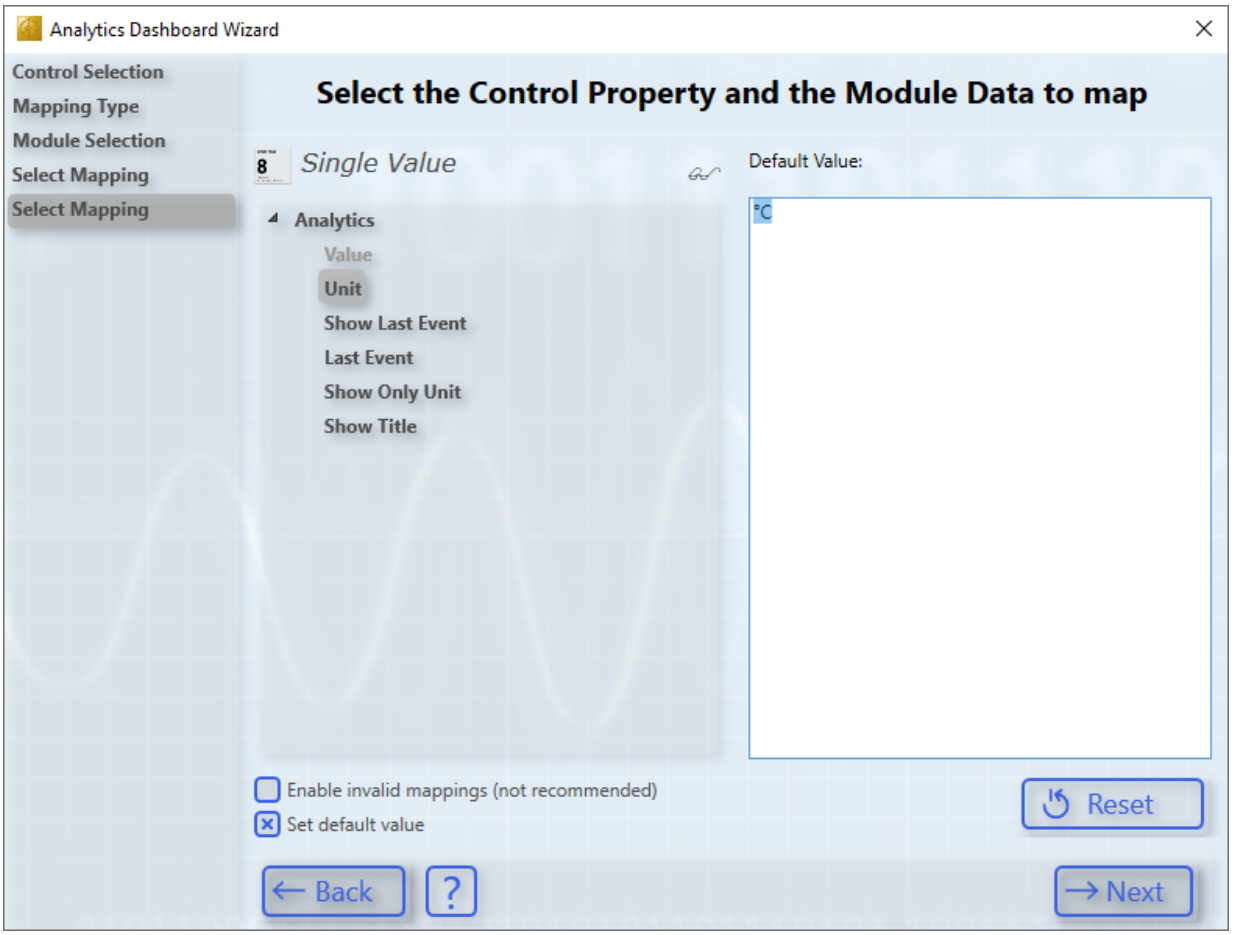
4. Here the first control property is linked to a module variable. To do this, select the **Value** to be displayed with the variable **Max**.



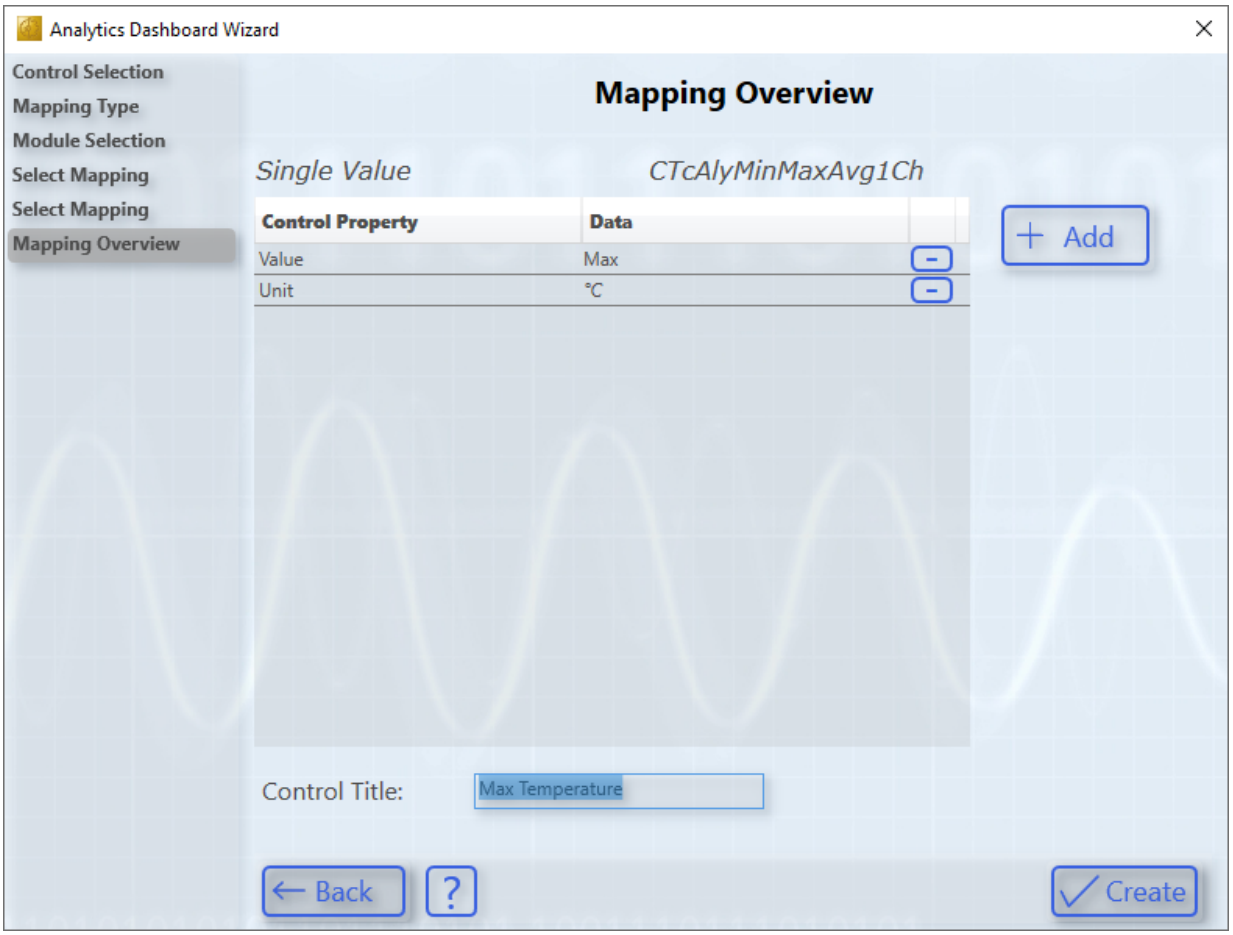
5. This overview page lists all existing links. Add another link with **Add**.



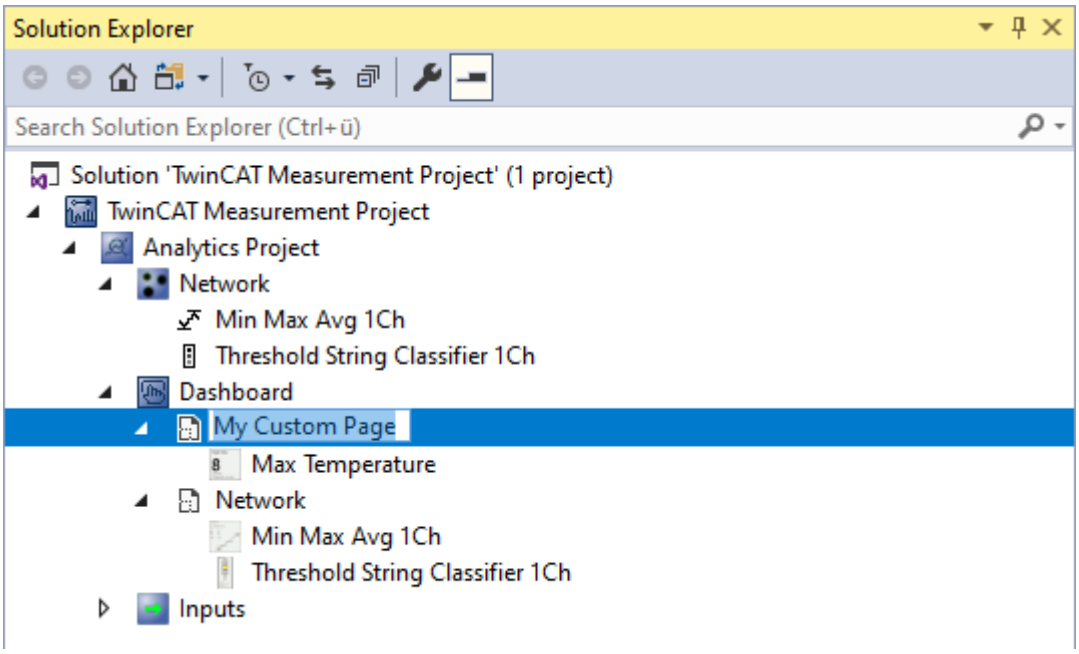
6. Select **Unit** and check **Set default value** to assign a static value. A text field opens on the right; enter **°C**.



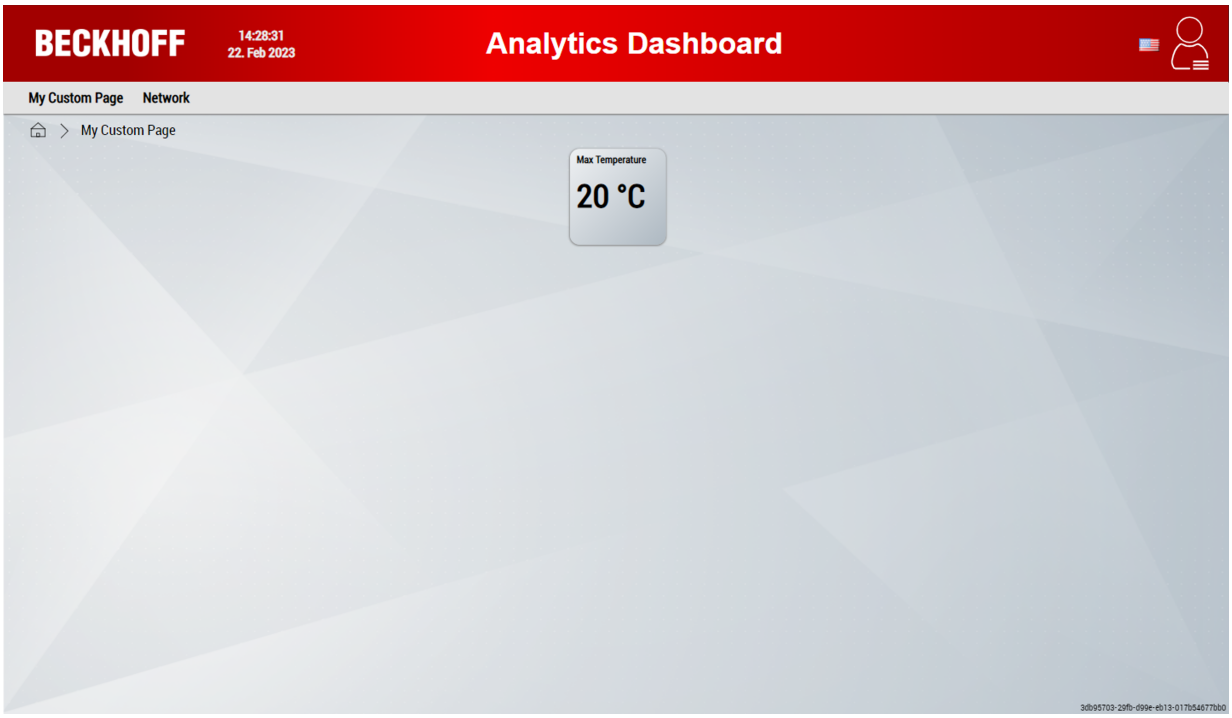
7. Change the title of the control to **Max Temperature** and add the control mapping via **Create**.



8. The **Max Temperature** control that was created appears in the **Dashboard** node. **Right-click** > **Rename** to rename the content.



⇨ After a successful HMI and PLC generation, you can open the previously created **My Custom Page** via the navigation. There you can see the manually created control.



6.2.2 Use customized and own controls

HMI Control Mapping Wizard

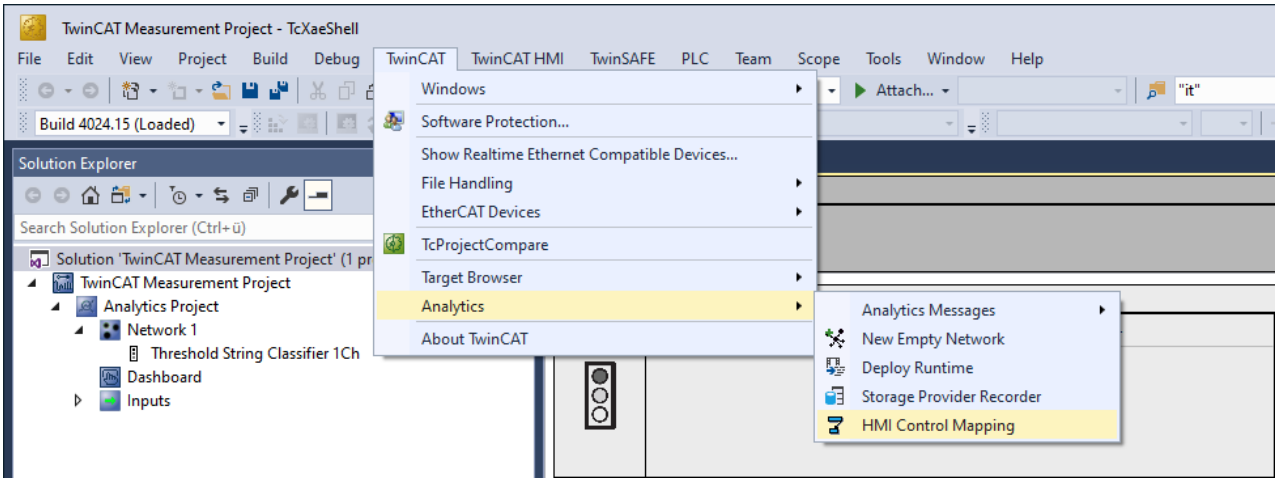
The HMI Control Mapping wizard enables the following:

1. Adding your own controls.

2. Mapping controls to module classes or changing existing mappings.

3. Mapping controls to module instances or virtual inputs (Analytics project must be open)

Open the wizard via the tab **TwinCAT > Analytics > HMI Control Mapping**.

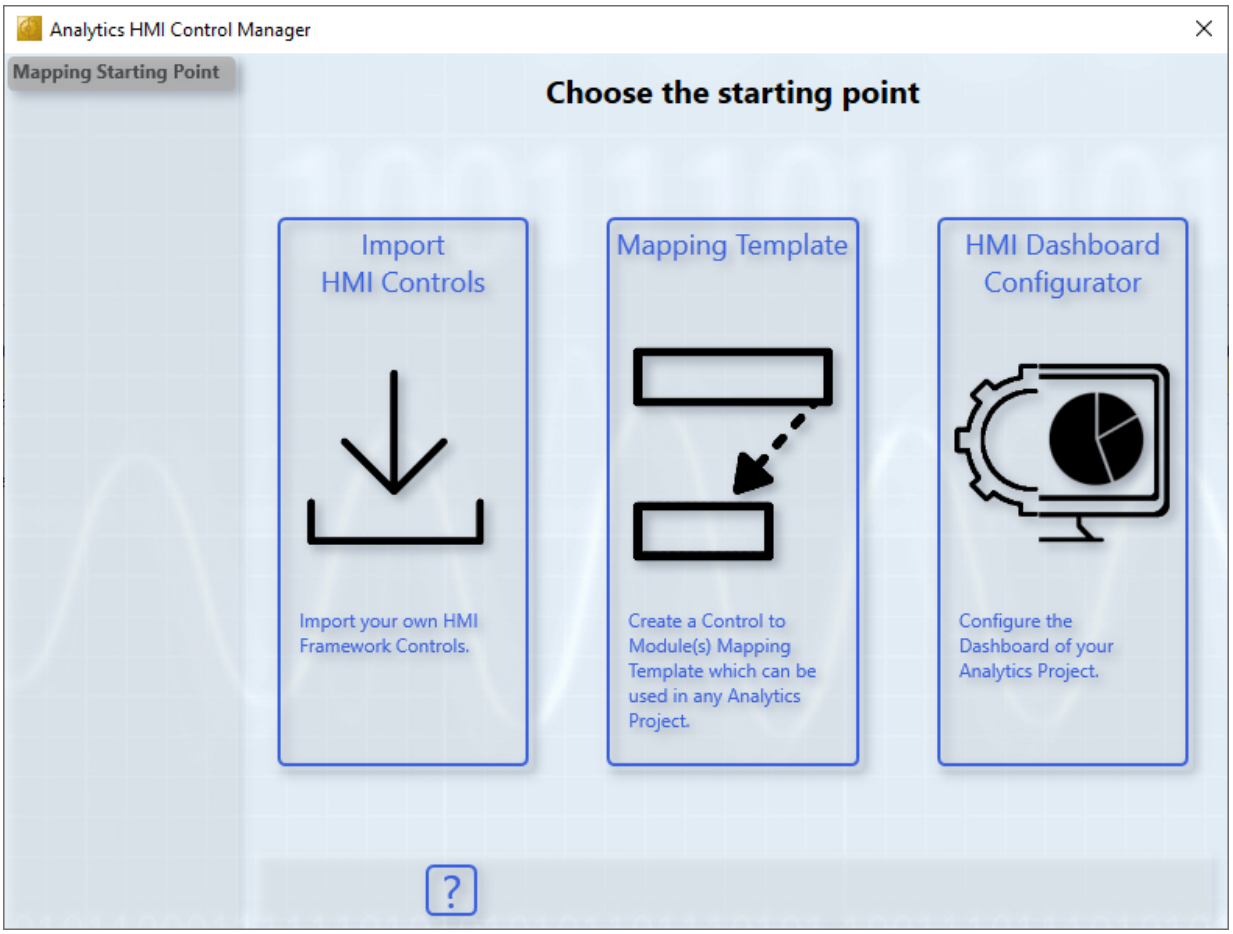


Additional help is offered via the question mark **?** in the wizard.

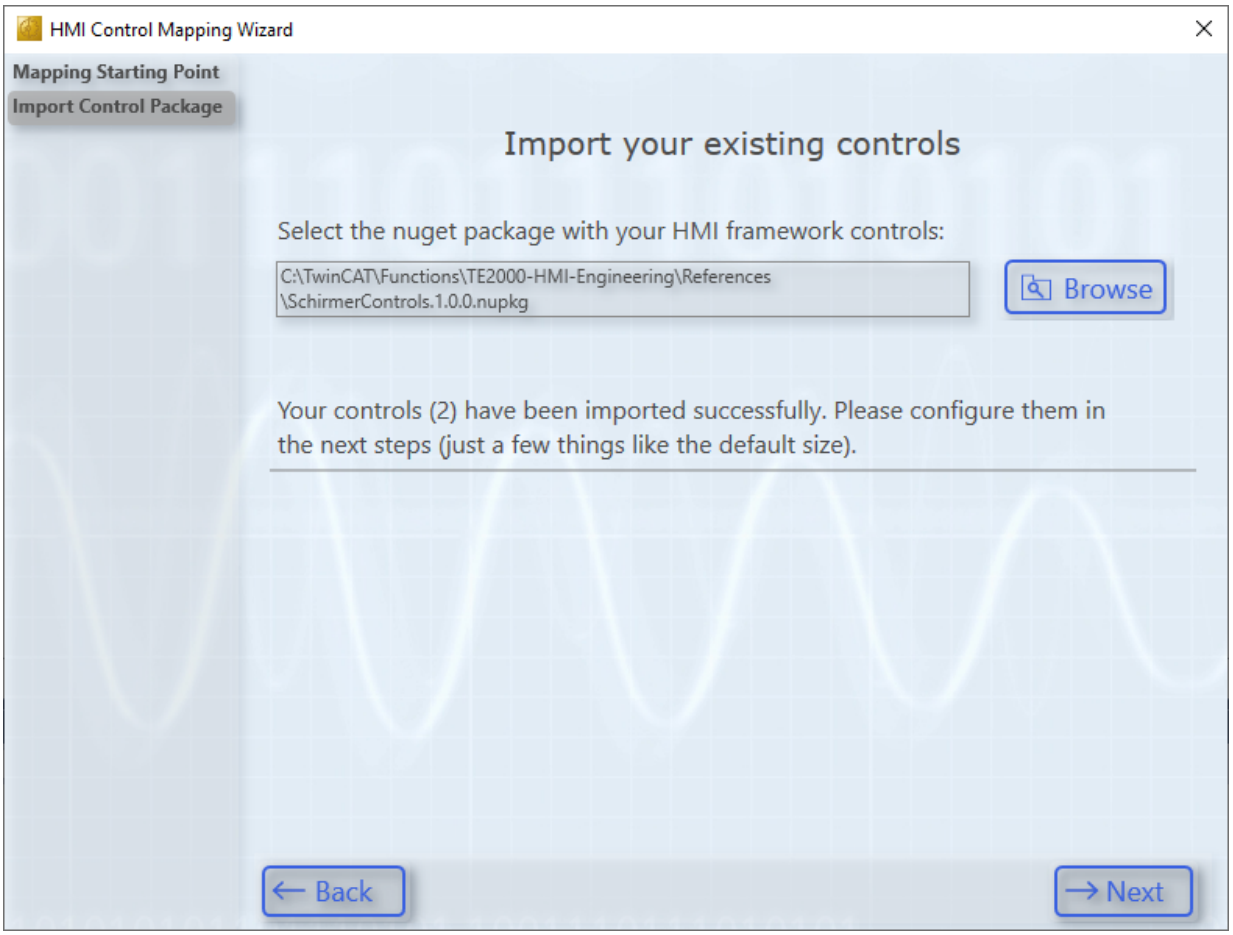
1. Adding your own Controls

The TwinCAT 3 HMI allows you to create your own HMI Framework controls and export them as a NuGet package.

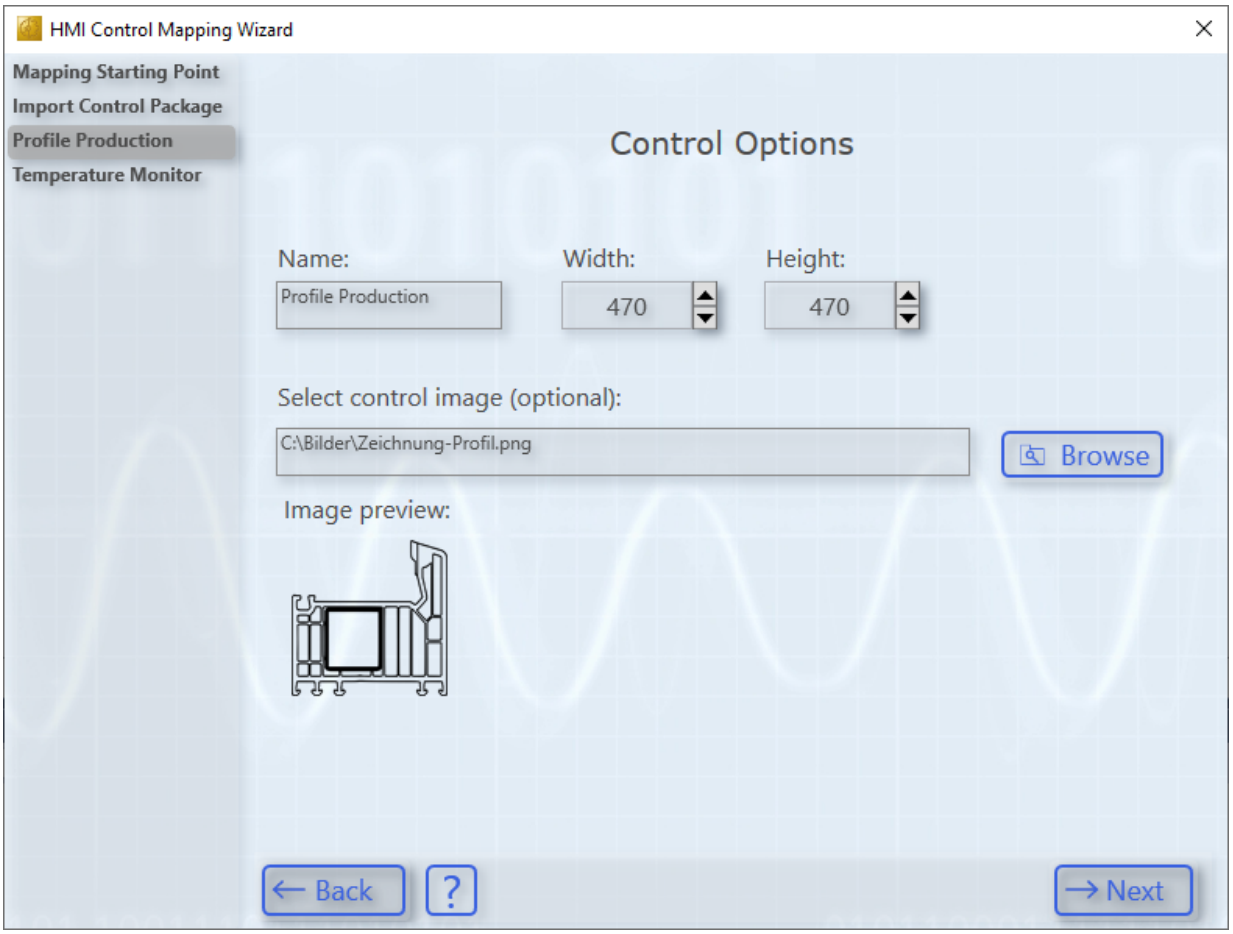
1. To assign your own framework controls to the Analytics modules, click **Import HMI Controls**



2. Select the NuGet package via **Browse**.



3. Next, choose a name, size and image for your control. Then click **Next** and for the last control click **Create**.

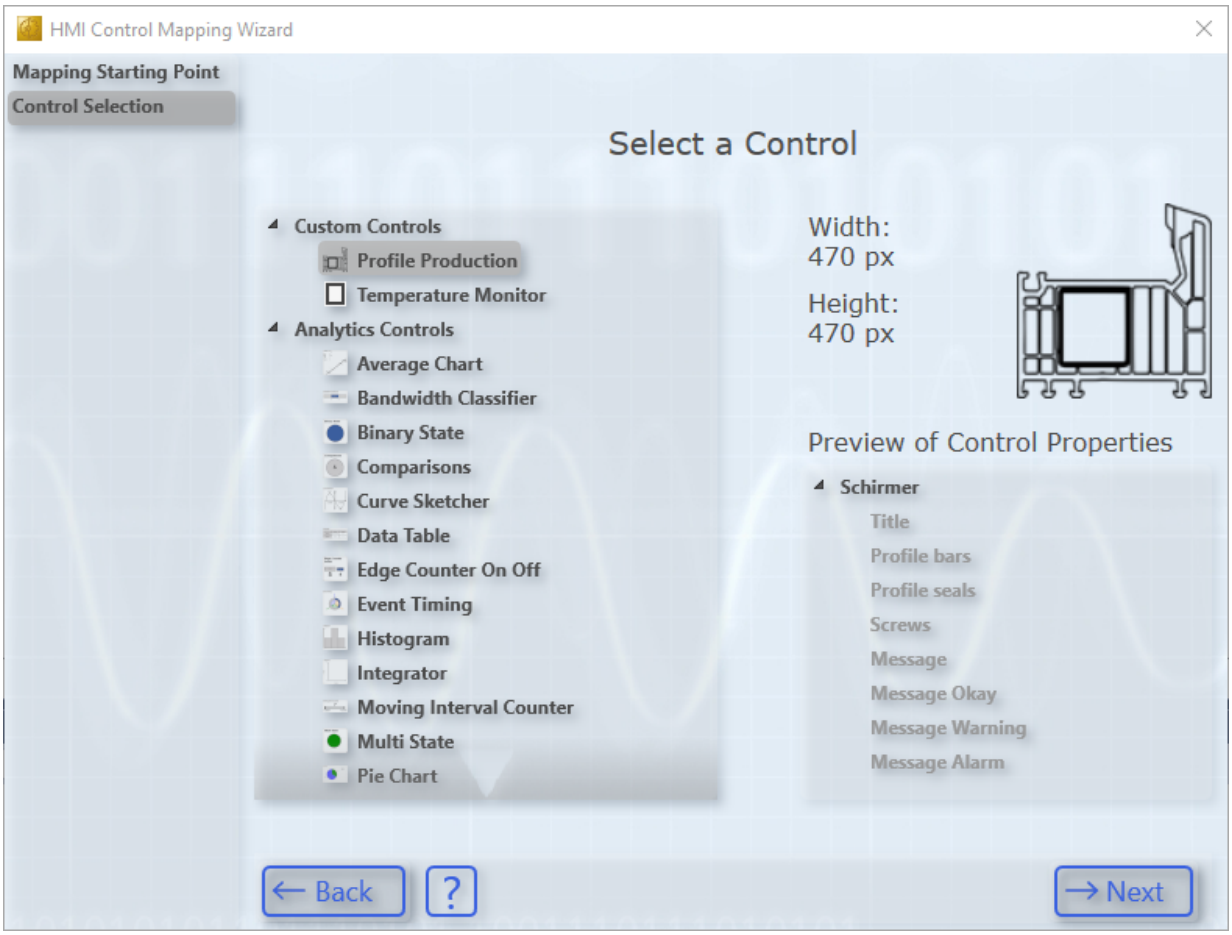


⇨ You will automatically be redirected so that you can create a mapping between your controls and Analytics modules.

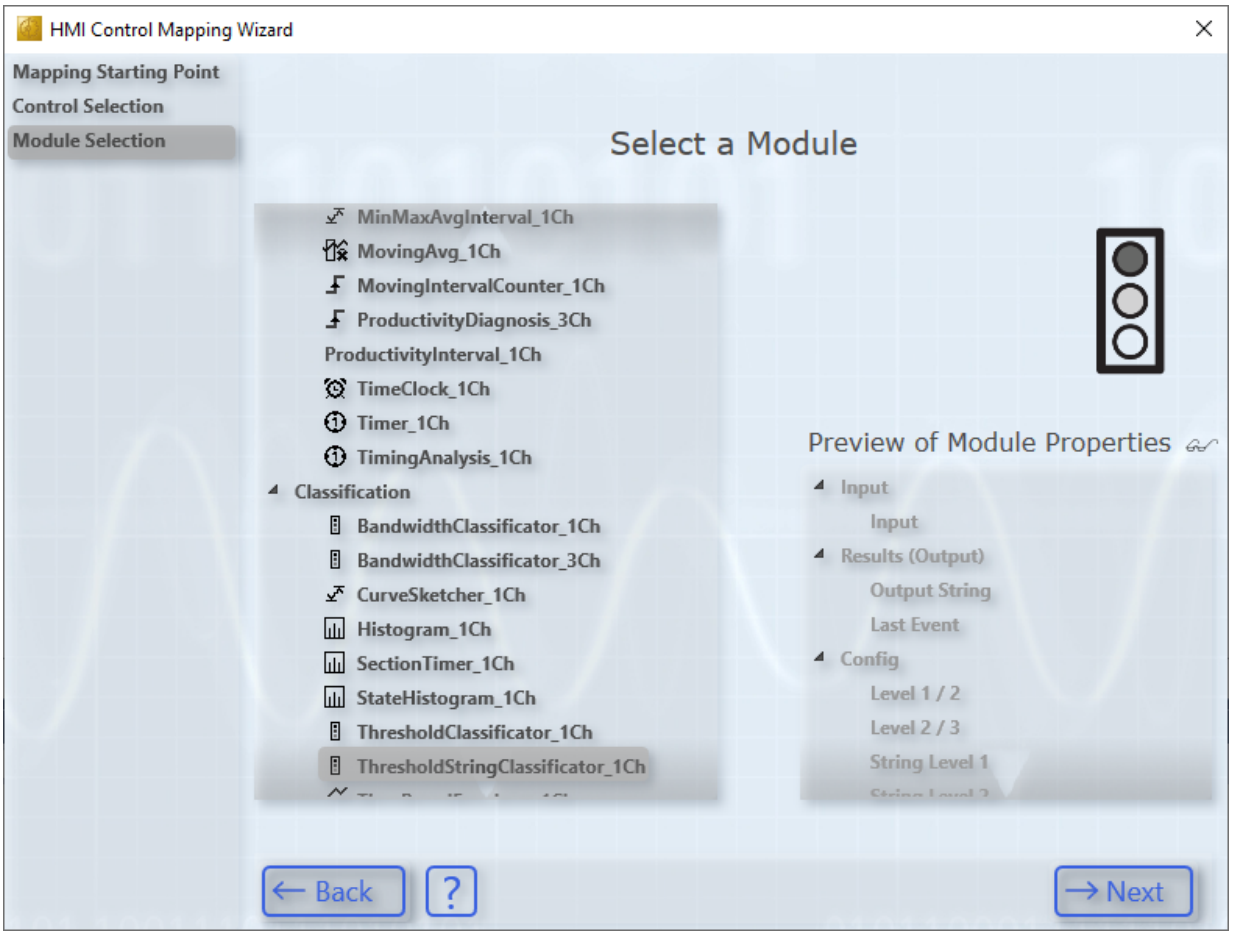
2. a) Mapping of controls to modules (further to 1.)

Under 2. b) this step is explained in more detail using "Binary State" control.

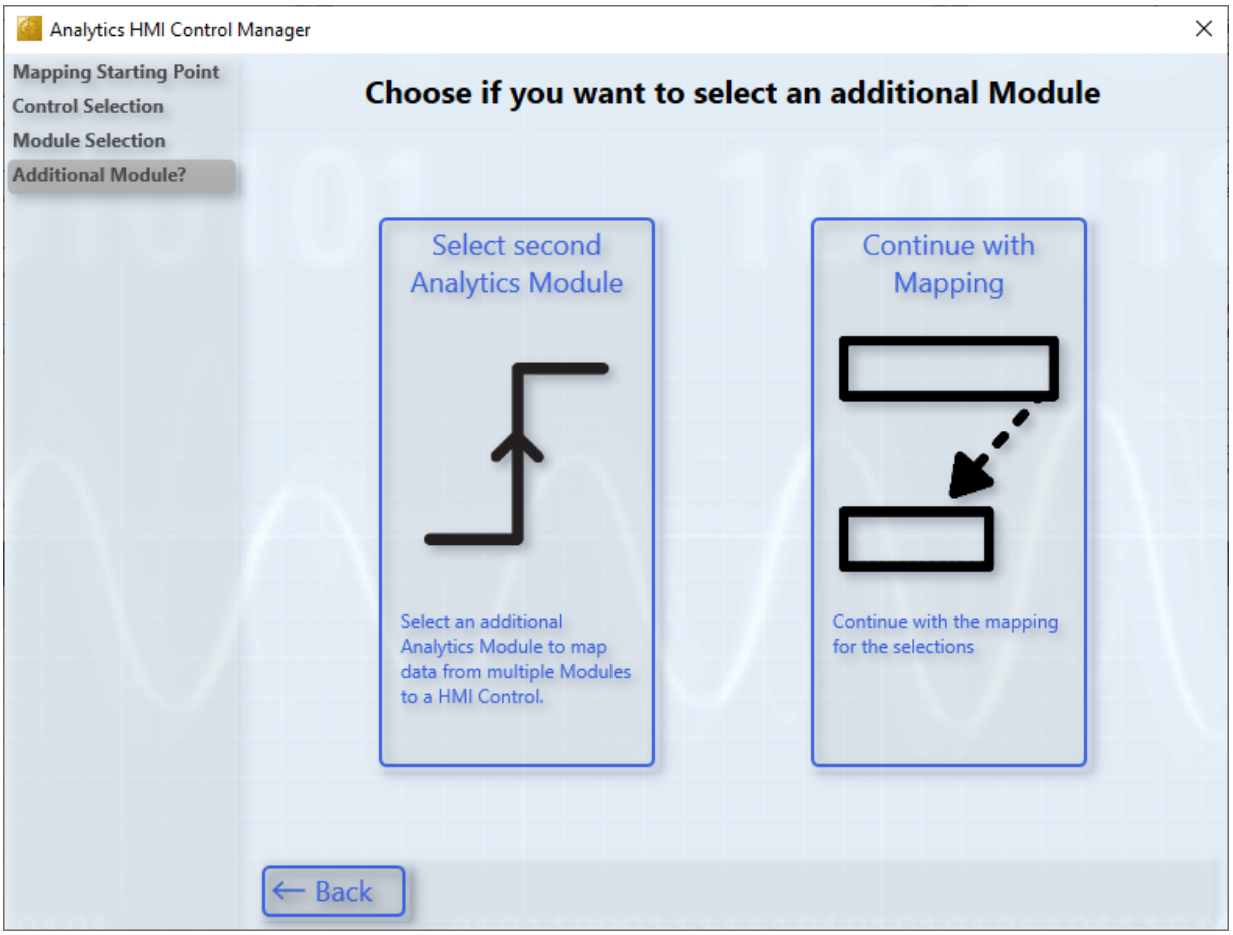
1. Now select your control.



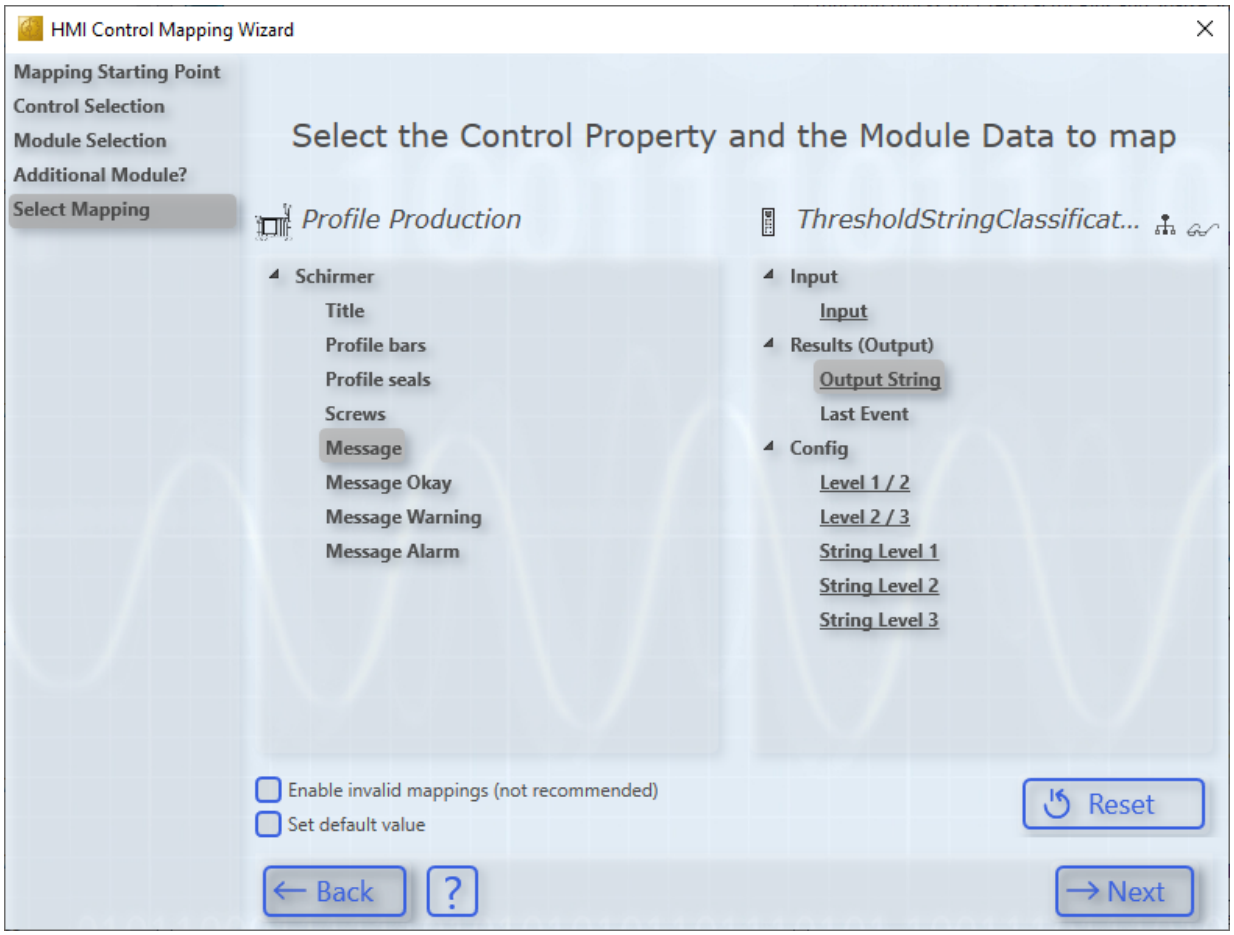
2. Now select the module to which you want to assign the control.



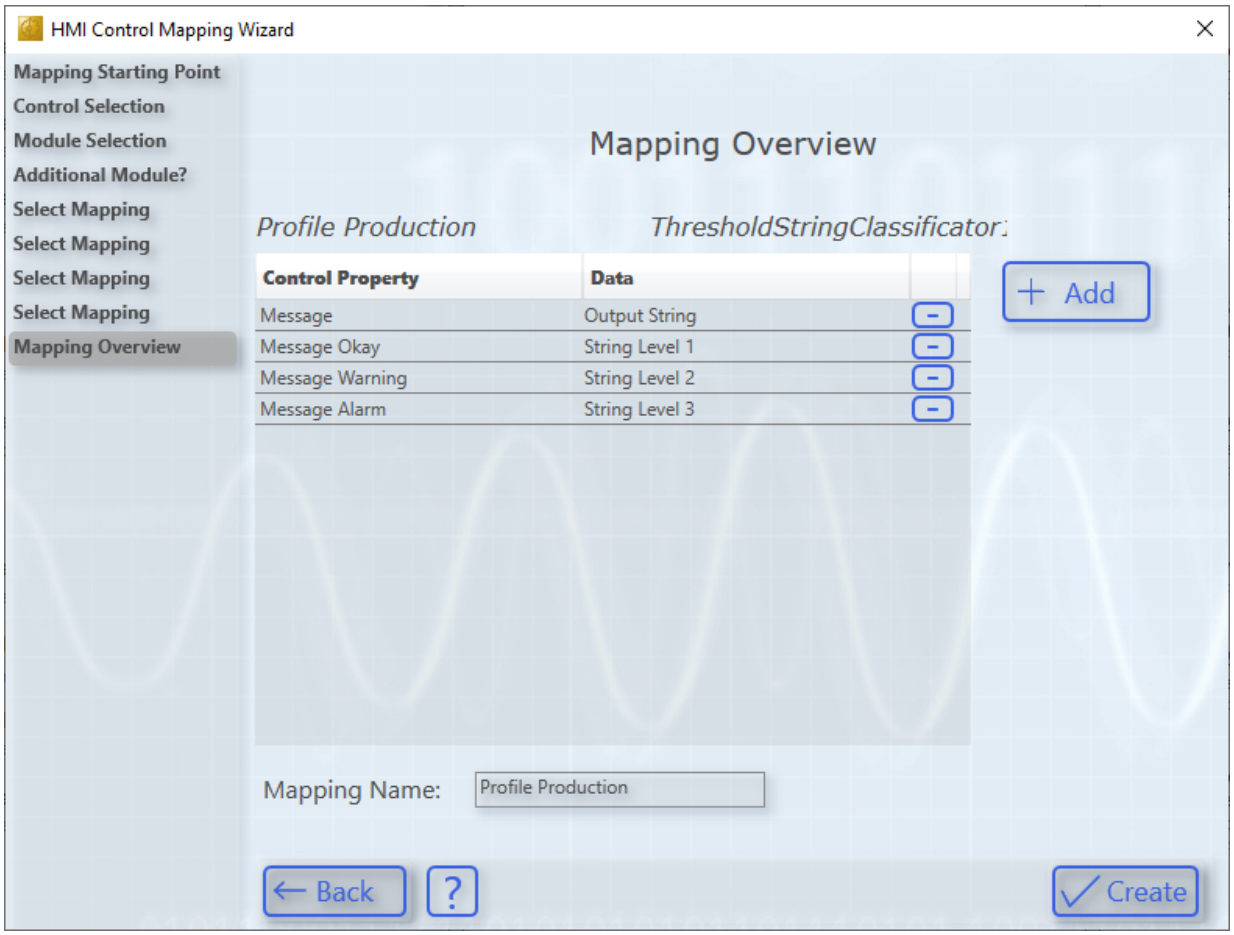
3. Select **Continue with Mapping**.



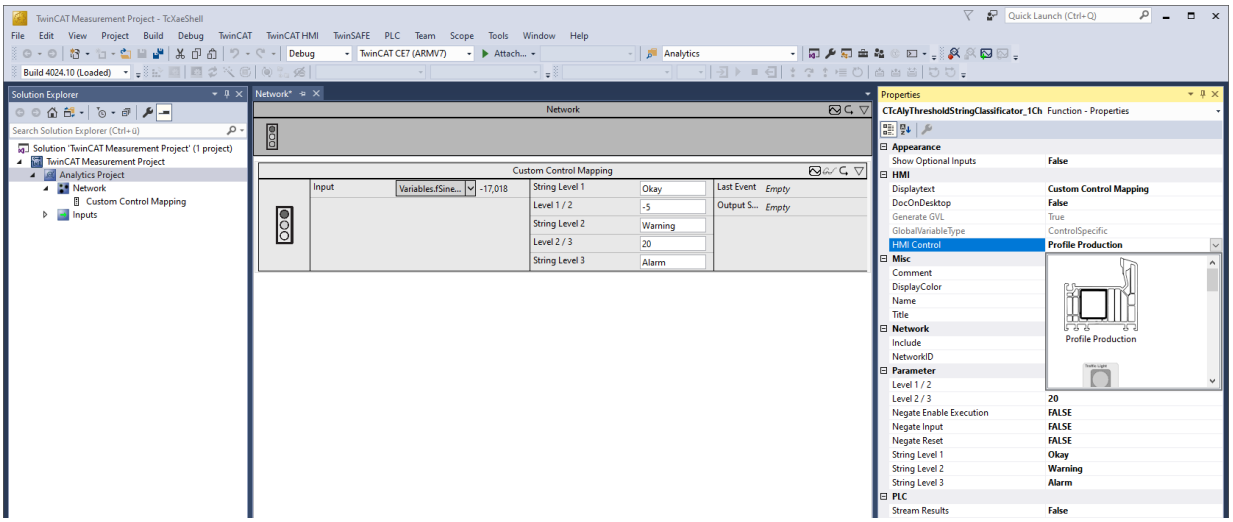
4. In the next steps, connect the control inputs to the module data.

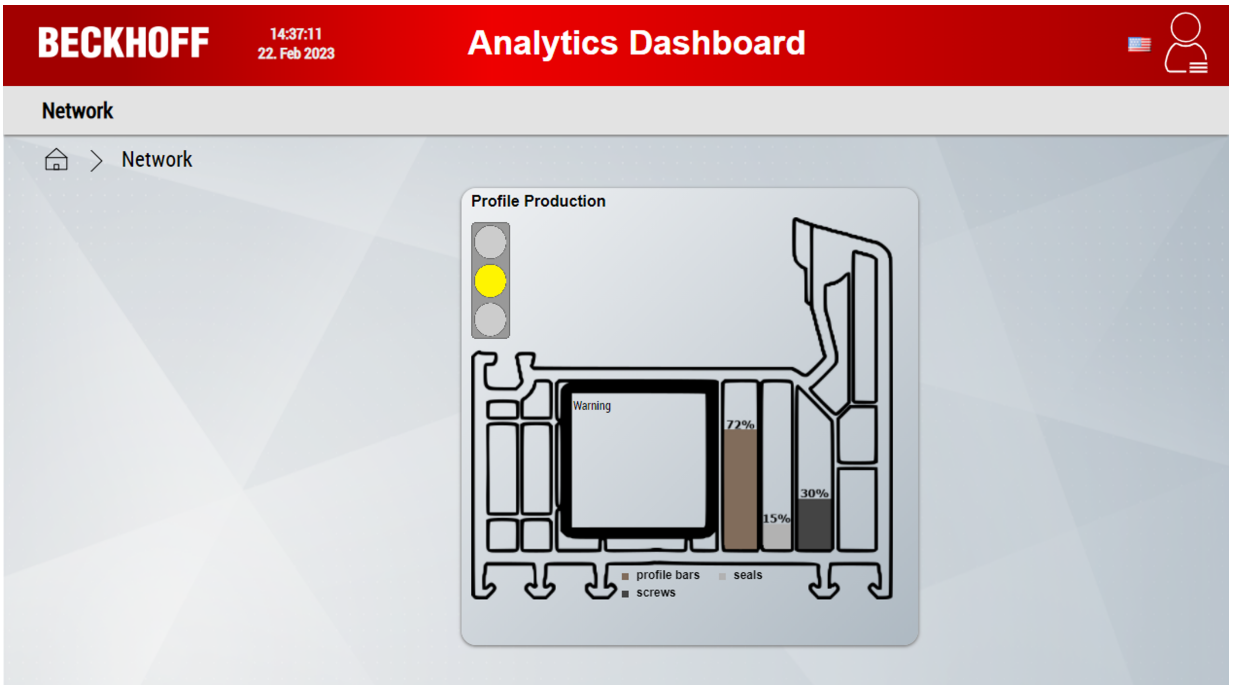


5. Last, click **Create** to add the mapping to the Analytics workbench.



6. You can now close the wizard and the mapping will automatically be available for the module. Select it and generate your dashboard.



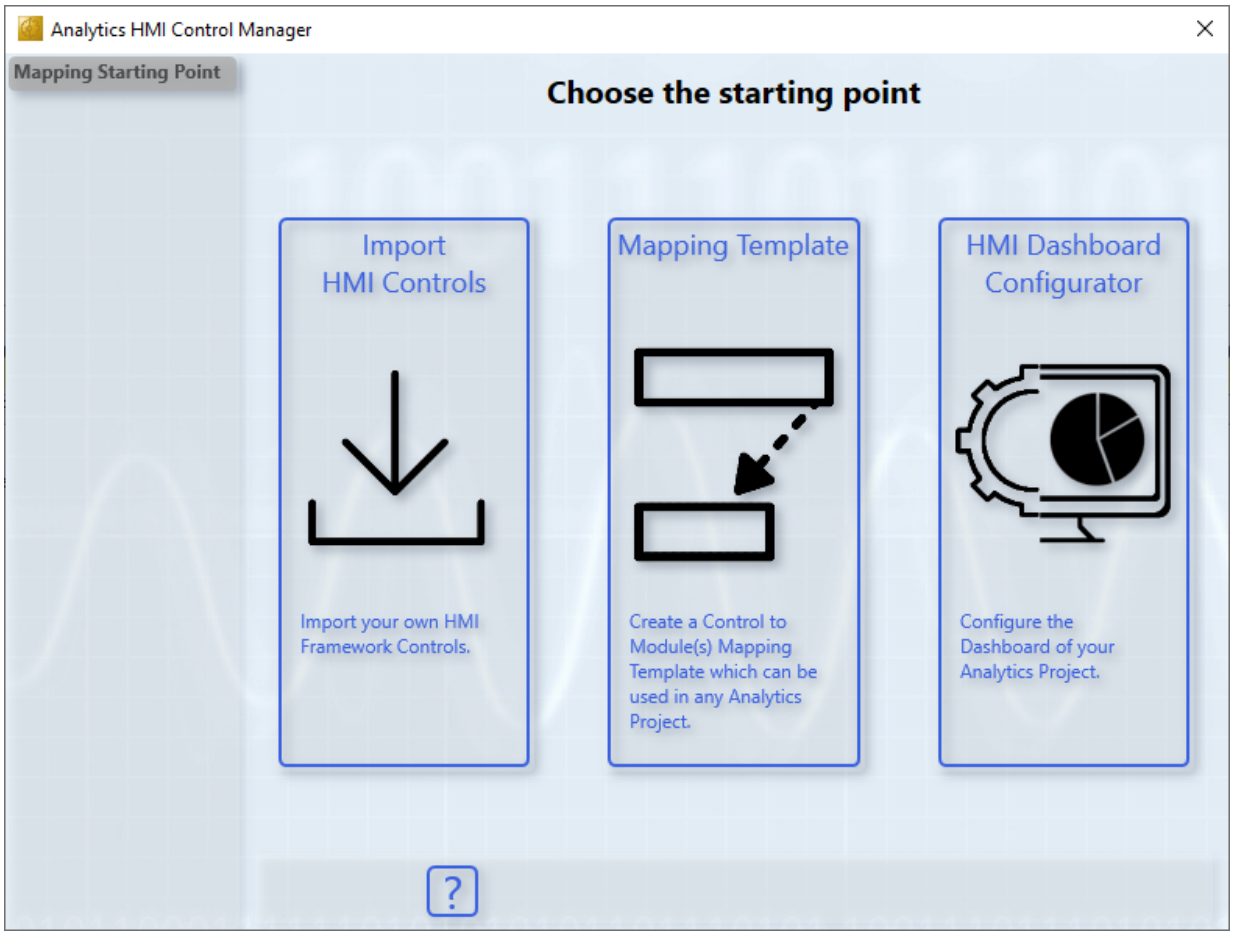


⇨ The dashboard is ready.

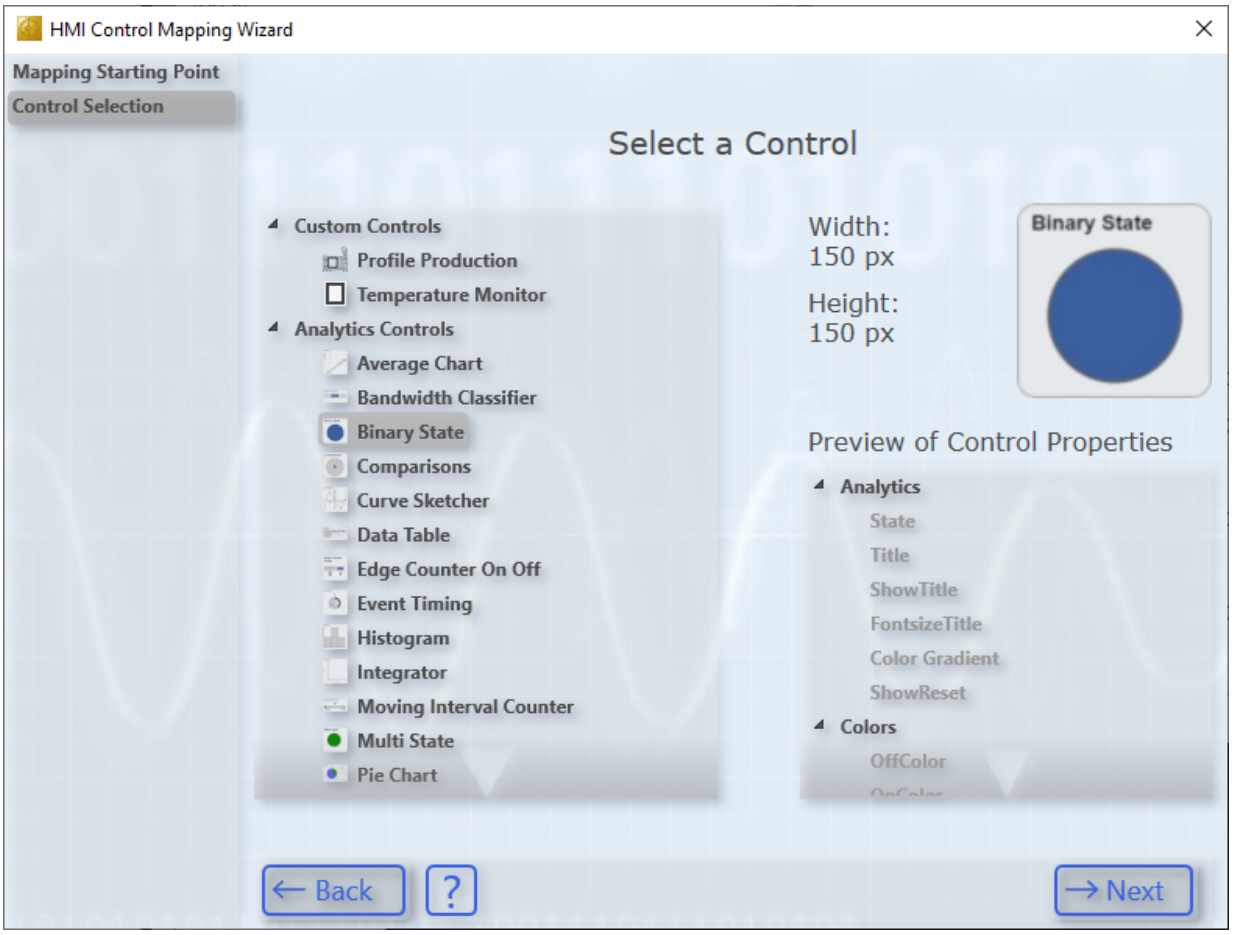
2.b) Mapping controls to modules (without own controls)

Analogous to 2.a), this section describes how to perform a mapping between a control and a module. An existing Analytics Control is used as an example. You can try this example directly.

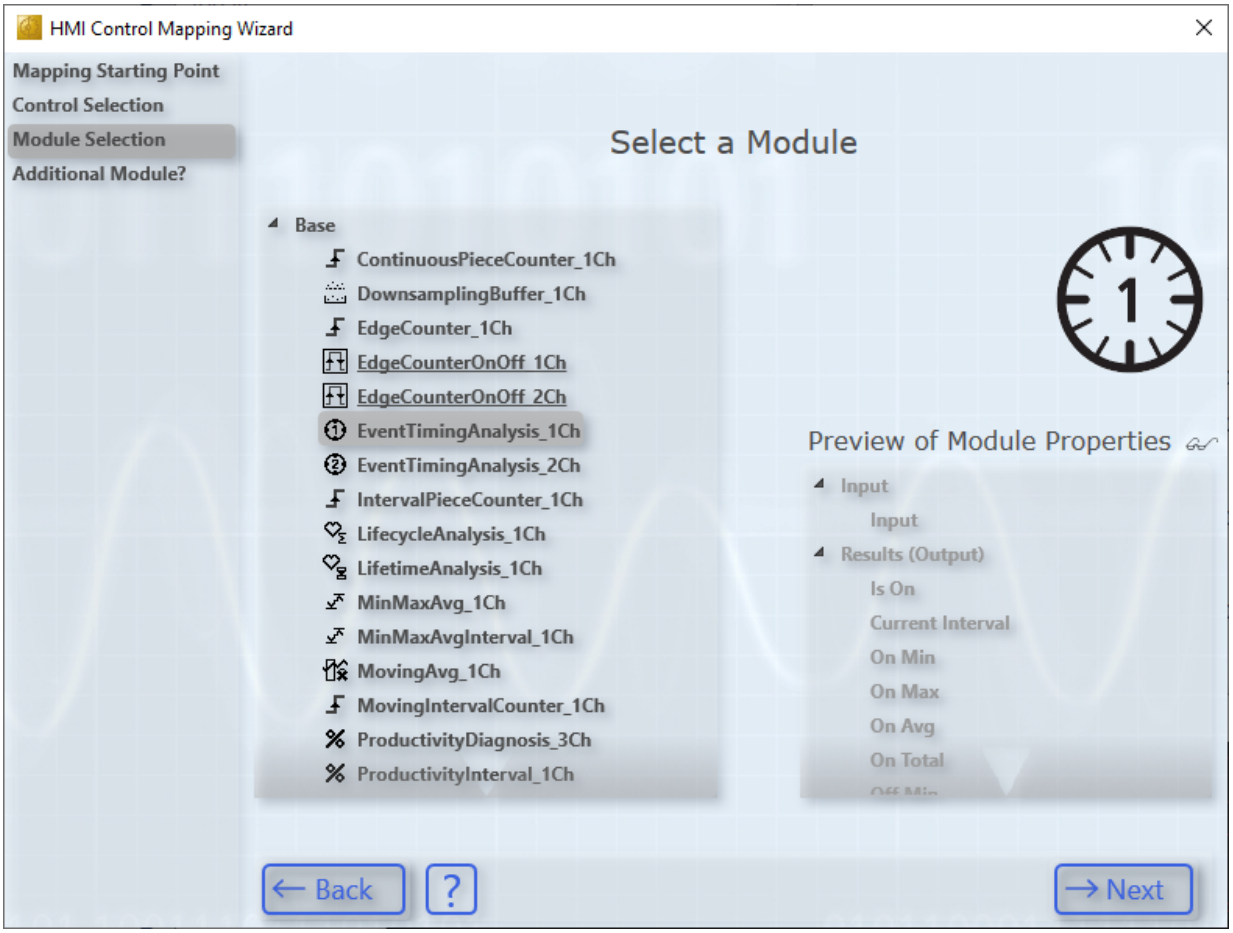
1. To do this, select the item **Mapping Template**.



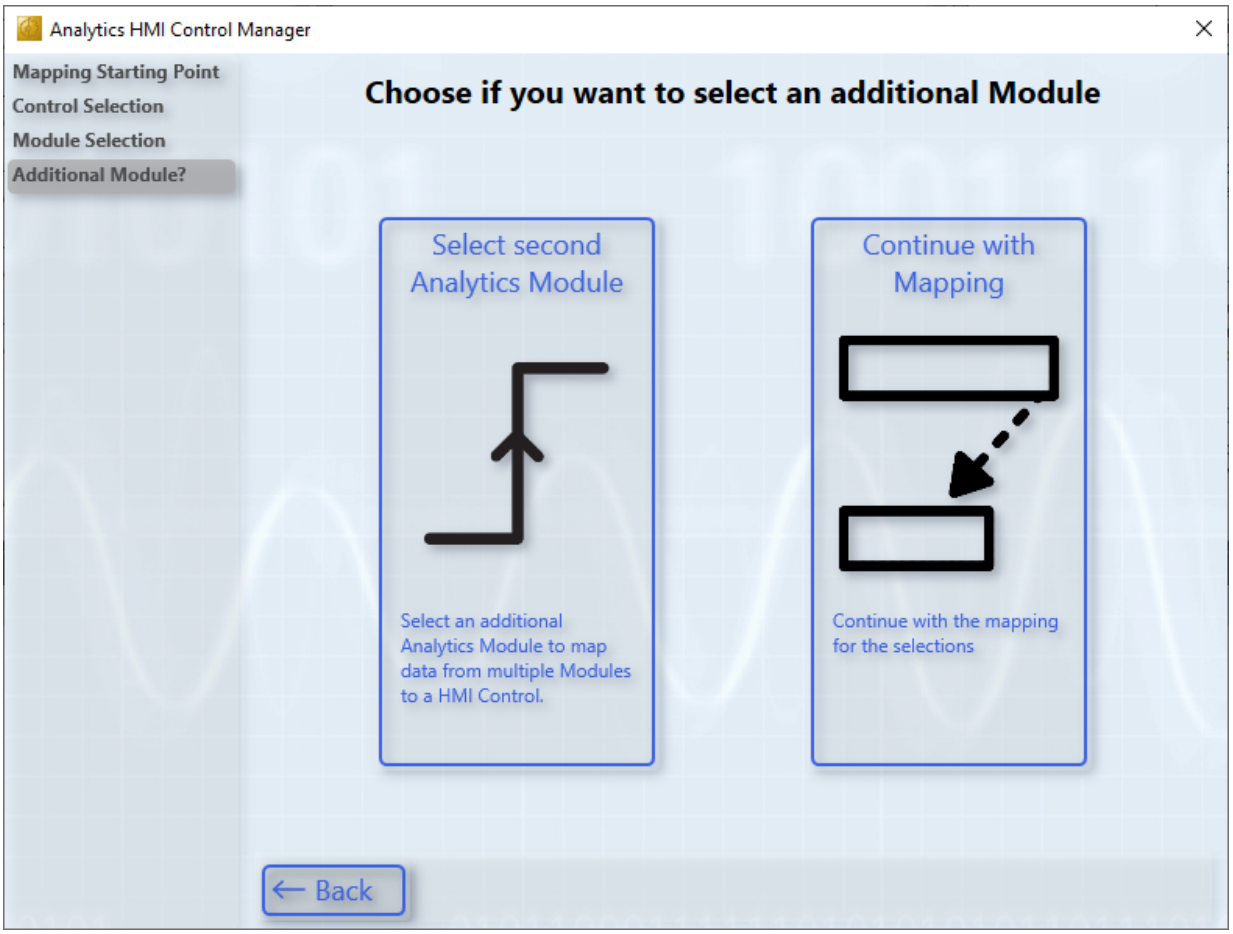
2. Select a control. The properties of the control are displayed on the right. It also shows how big the control will be on the dashboard.



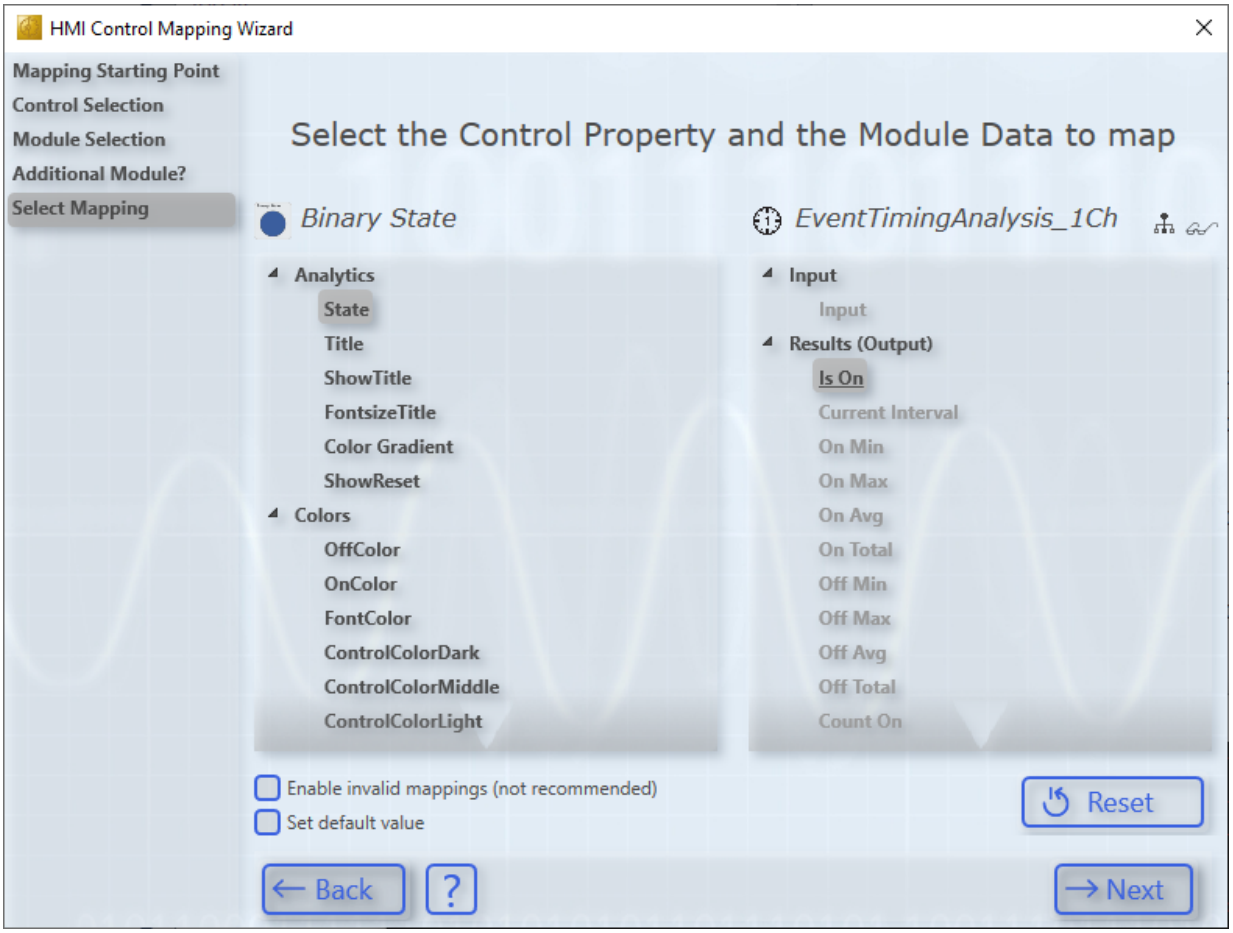
3. Select the module from which you want to display the data. The inputs, outputs and parameters are displayed as a preview, which can then be linked to the properties of the selected control.
   Modules that have already been mapped are underlined. These mappings can also be edited.



4. In the future, a control will be able to display data from different modules. Since this is not currently possible, select **Continue with Mapping**.

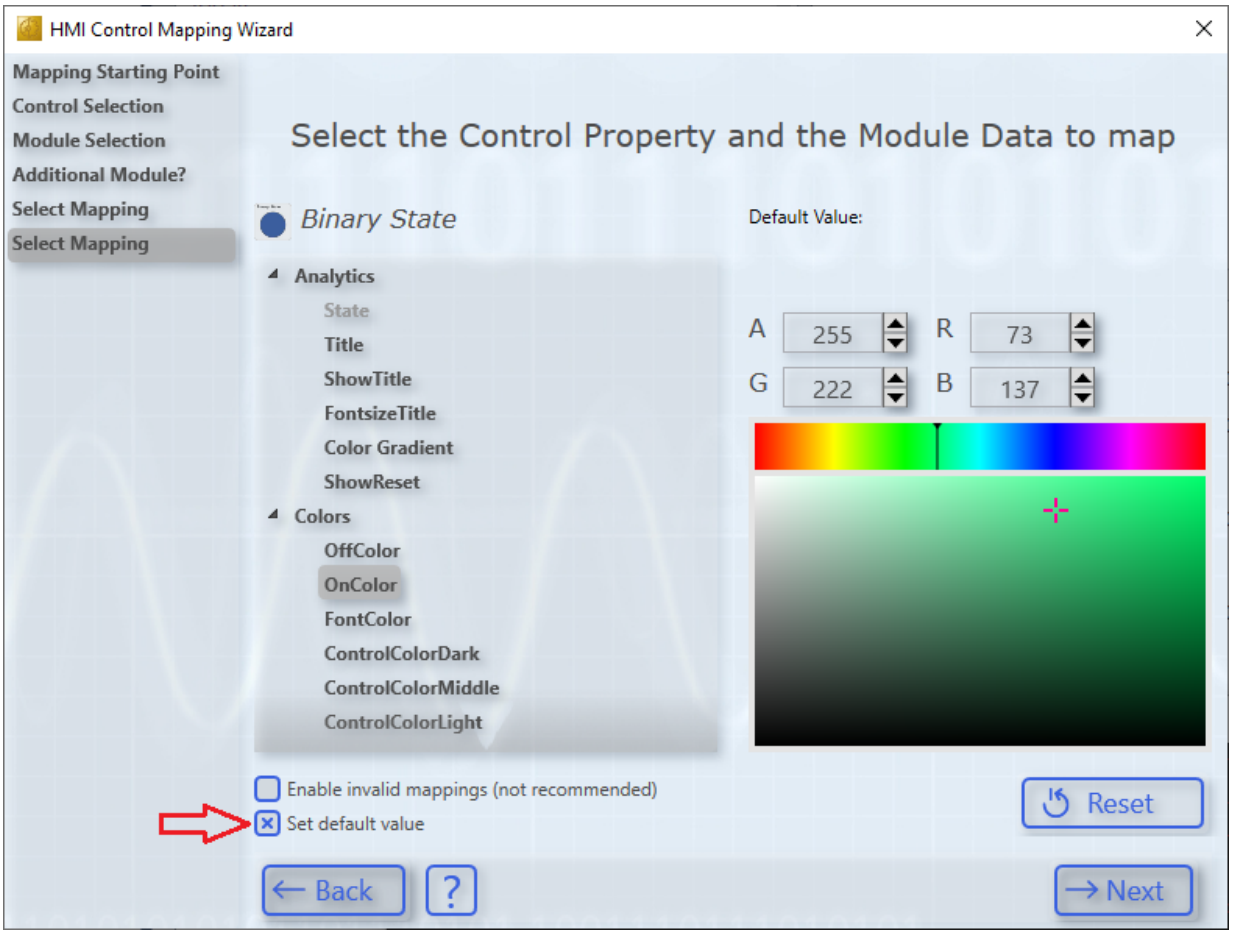


5. Next, select a control property to be mapped. All impossible variables that cannot be assigned due to their data type are grayed out. All others can be mapped.
   Now select the module variable you want to map with the control property.

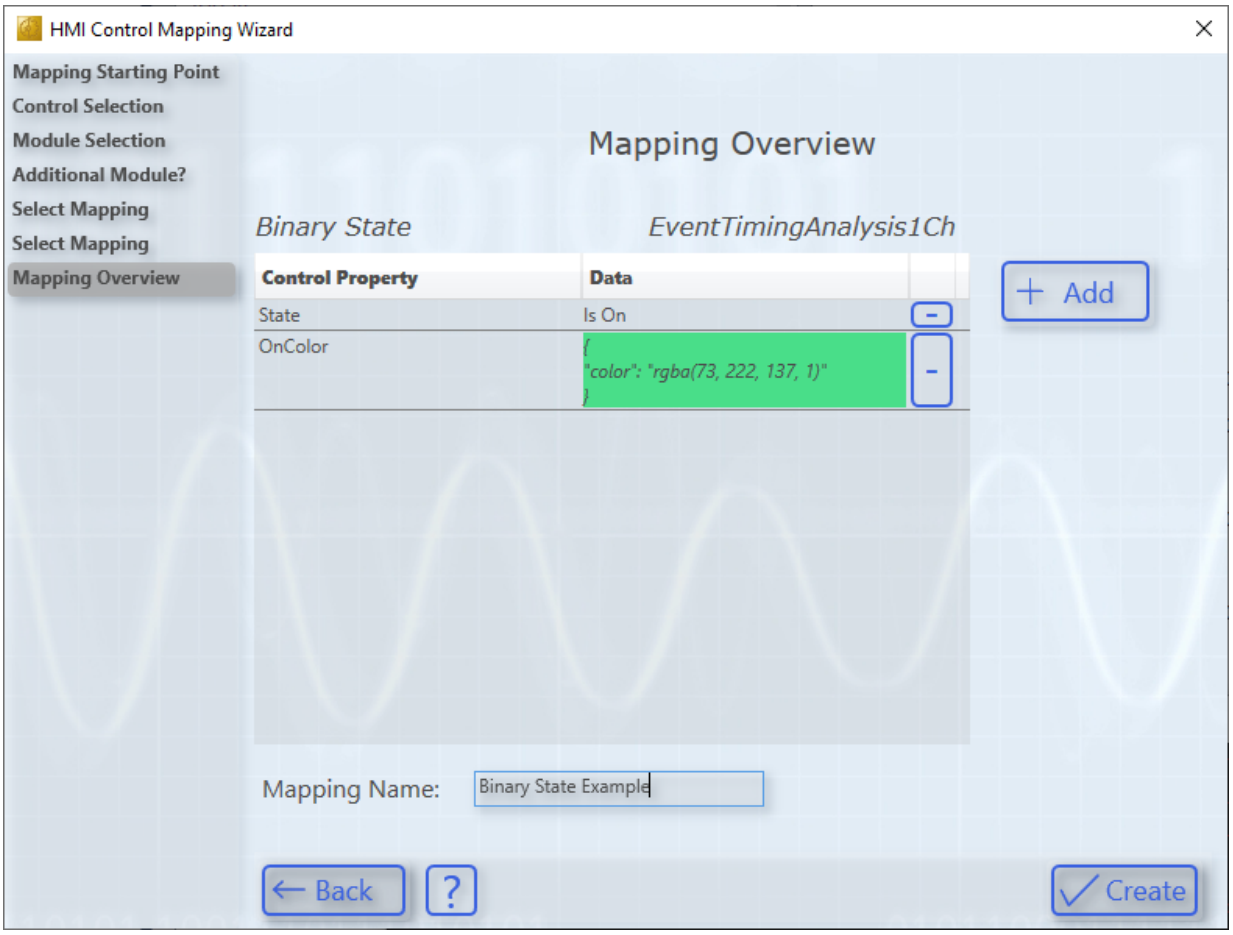


6. Likewise, you can check the **Set default value** checkbox to assign a default value to the control. This can be used, for example, to change default colors or to set Boolean values such as "ShowTitle" to "False" if no title is to be displayed in the Analytics control.

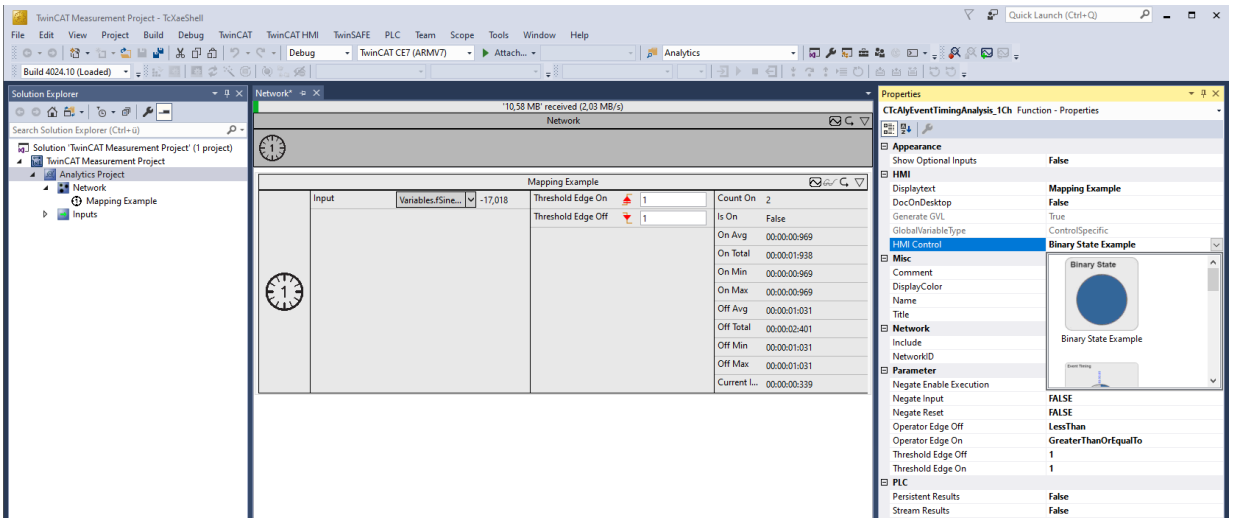
In this example, the "OnColor" is changed. Based on the data type, a specific selection option is provided.

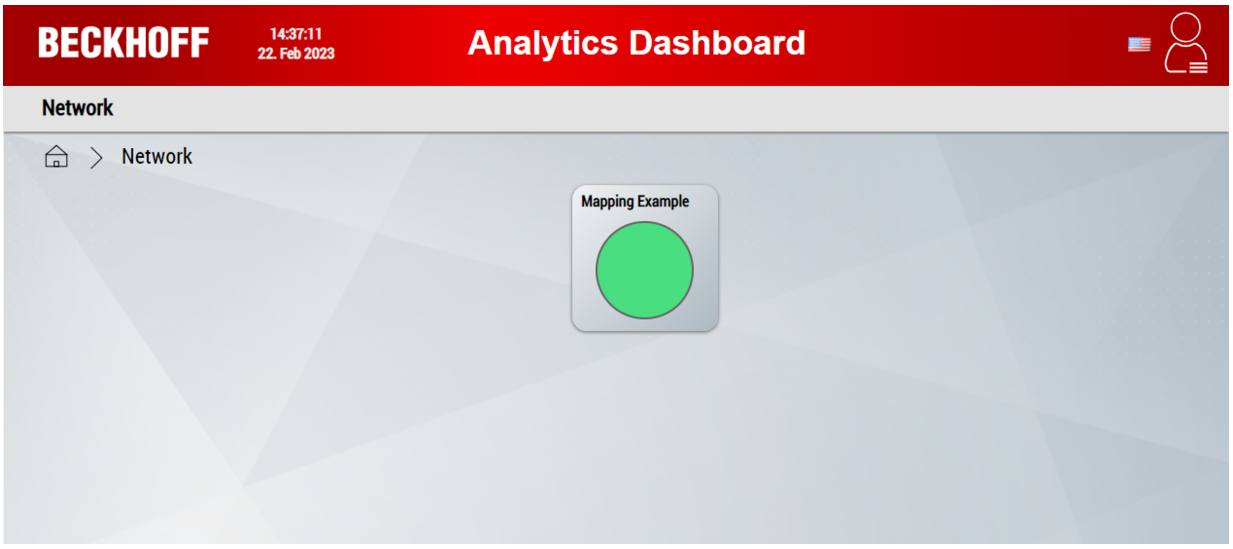


7. As far as you do not want to add any more entries via **Add**, you can complete the mapping via **Create**. It is recommended to assign a meaningful mapping name.



8. You can close the wizard and the mapping will automatically be available for the module. Select it and generate your dashboard.



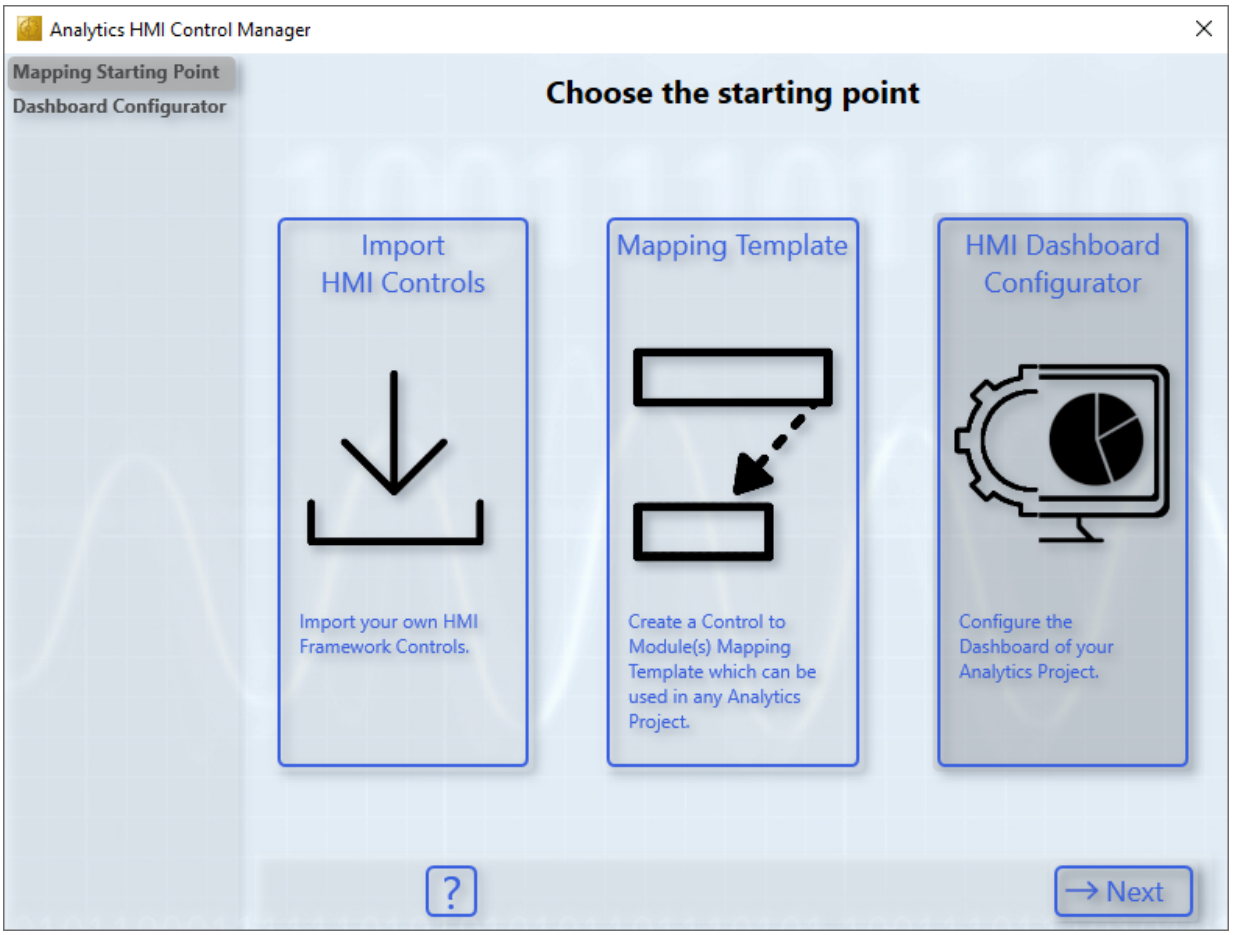


⇨ The dashboard is ready.

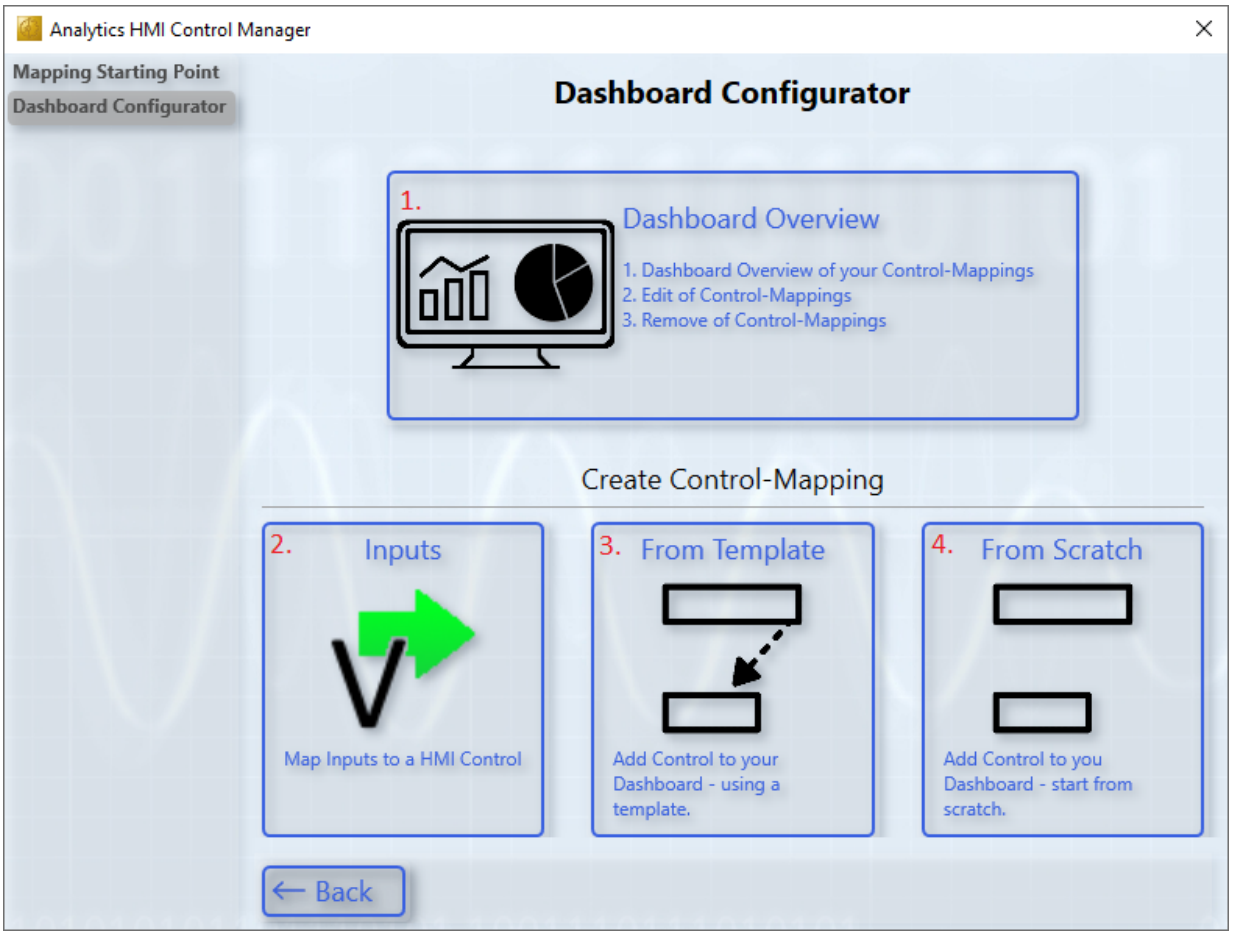
3. Mapping of controls to module instances

In addition to mapping controls to module classes, the module instances of a project can also be directly linked to controls. This is possible via the Manage dashboard structure and content in Analytics project [▶ 110] as well as via the wizard.

1. At the beginning of the wizard, click **HMI Dashboard Configurator**.



2. There you have several choices.



1. Overview and editing options for all controls
   - This function is also available by double-clicking a control in the Solution Explorer.
2. Adding a control that displays your input data
   - This function is also possible via the Manage dashboard structure and content in Analytics project [▶ 110].
3. Adding a control that displays module data (selection via a template)
   - This function can also be set for an individual module via the *Properties* window. This is the only way to display data from several modules.
   - You can see directly which data from the template are linked and can adjust them directly.
4. Adding a control that displays data from modules (fully manual)
   - This function is also possible via the Manage dashboard structure and content in Analytics project [▶ 110].

For the individual points you have to follow the steps of the wizard. The steps are the same or very similar to section "2. Mapping of controls to modules".

6.2.3 Configure user management and access rights

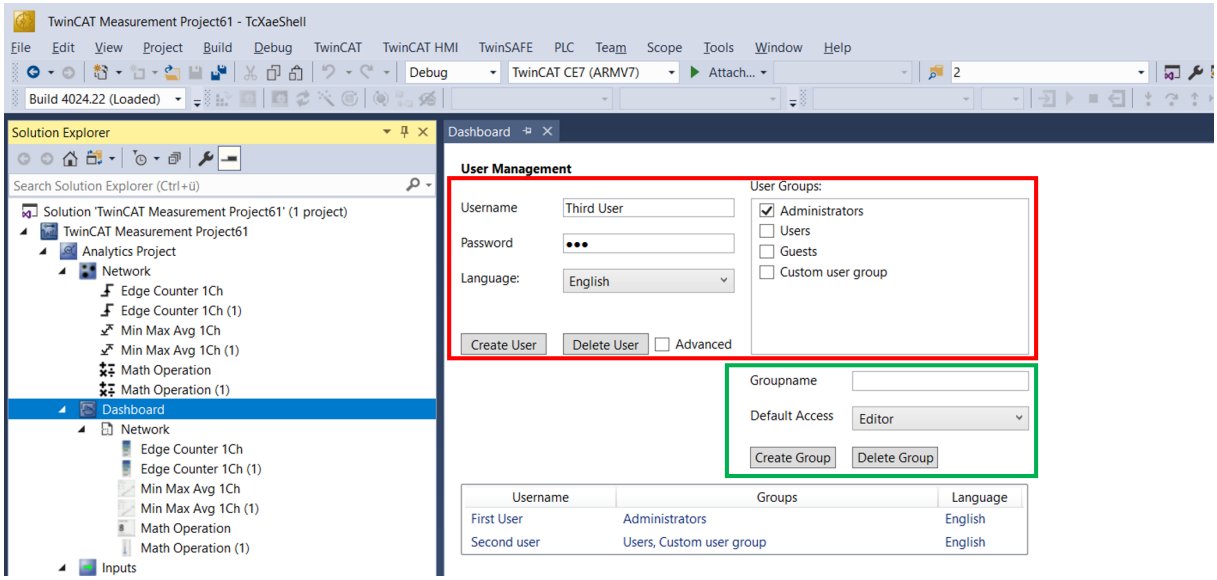
Users and user groups can be generated individually via the user management. Access rights to contents and controls can be defined for user groups.

Note that user management and access rights are only transferred to the HMI project from Dashboard version 2.0.

6.2.3.1 Definition in the Analytics Project

Defined users, user groups and access rights are automatically transferred to the HMI project during code generation. Changes to users and access rights can be made in the web browser after generation.

6.2.3.1.1 User management



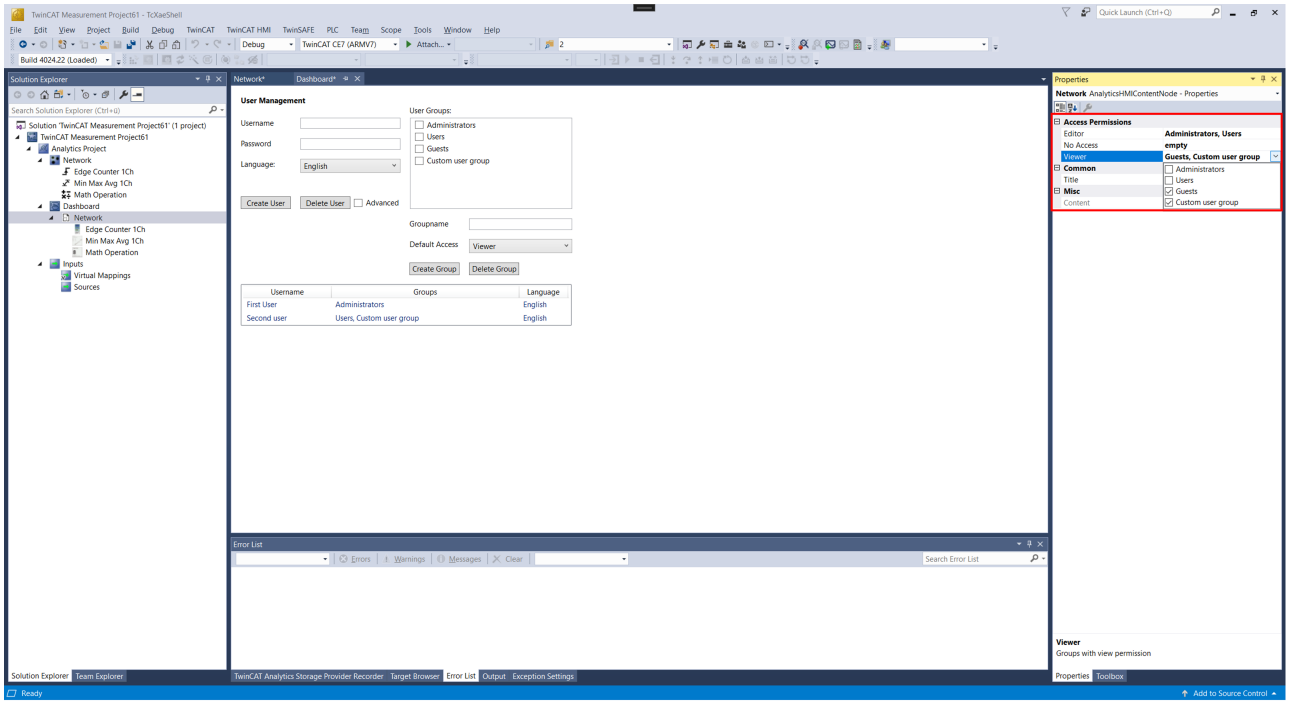
Users (outlined in red) can be created under the Analytics project dashboard node. For a new user, a username and password must be set, as well as a language. In addition, it is necessary to assign one or more user groups to the user, which gives the user the access rights of the respective group(s).

By default, there are the following user groups:

- Administrators
- Users
- Guests

For a more precise classification, additional user groups can be created (outlined in green). For this purpose, a name and the access authorization must be entered.

6.2.3.1.2 Configure access rights



Access rights can be configured for objects below the dashboard node (contents and controls). Access permissions (outlined in red) can be set for each user group.

The following access rights are available:

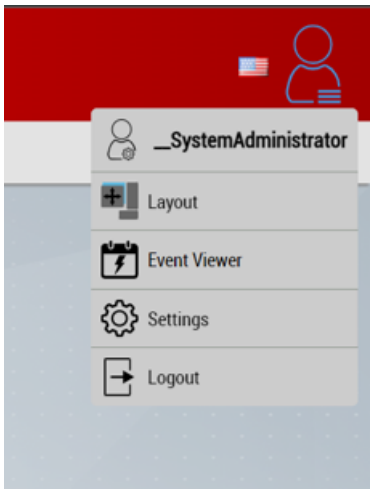
- Editor: Access to the object is permitted, and changes (such as layout adjustments for content objects) can also be made.
- Viewer: Access to the object allowed - but no changes can be made.
- No Access: No access allowed to the object and the objects subordinate to it in the Solution Explorer.

Access rights for content objects can be subsequently adjusted via the web browser.

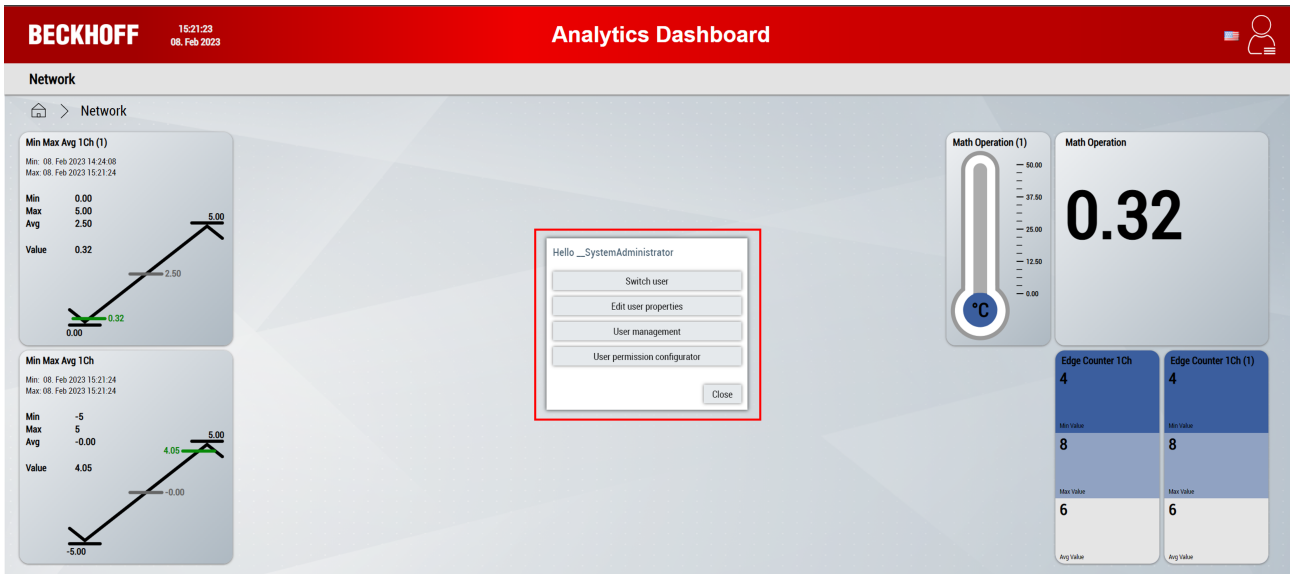
6.2.3.2 Customize users and access rights in the web browser

Note that the menu control and the configuration of users and access rights are only available from Dashboard version 2.0.

Configure user



The menu control can be used to make changes to the users and access rights. To do this, please click on your username (at the top of the menu).



The following options are available via the popup (outlined in red):

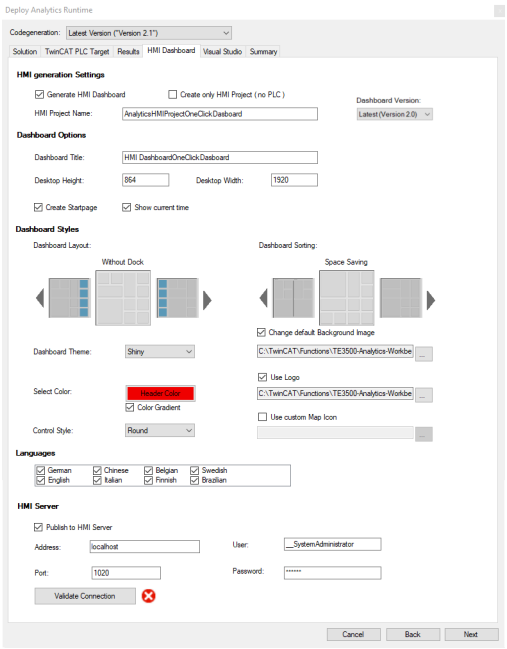
- **Switch user:** Change of the logged in user
- **Edit user properties:** Change the properties of the own user
- **User management:** Create new users, remove existing users and set user properties
- **User permission configurator:** Configuring the access rights of user groups to content objects

Note that when creating new users, individually configured layouts of the creating user are taken over.

6.2.4 Dashboard Configuration

HMI Dashboard tab

The **HMI Dashboard** tab contains all the configurations for the dashboard.



Configuration

HMI generation Settings

Generate HMI Dashboard	Enables the generation of the HMI Dashboard, if enabled. This automatically activates/deactivates the checkboxes **Create Bootproject** and **Activate PLC Runtime** on the **TwinCAT PLC Target** tab.
Create only HMI Project (No PLC)	Creates only one HMI project and not a PLC project, if enabled.
HMI Project Name	Name of the TwinCAT HMI project.
Dashboard version	Selection of the dashboard version.

Dashboard Options

Dashboard Title	Title of the HMI Dashboard, which is displayed in the dashboard header.
Desktop Height	Height of the target screen in pixels.
Desktop Width	Width of the target screen in pixels.
Create Startpage	Creates a start page for the dashboard that displays a map with all machine locations. The location data is adopted from the machine management data.
Show current time	Creates a clock in the dashboard header that shows the current local time.

Dashboard Styles

Dashboard Layout	Defines the layout of the dashboard. Dock requires the "Dock" property of a module to be TRUE. Dock Left: fixed left column Dock Right: fixed right column Without Dock: no fixed column
Dashboard Sorting	Defines the sorting of the dashboard. Space Saving: arranges the controls without free space. Control Type: arranges the controls by Control type. Control Size: arranges the controls by size from large to small. Filled: arranges the controls so that the entire screen is filled. Network Groups: groups the controls by network and summarizes them on a screen. Grouping begins at the defined level (0 = All, 1 = First Level...)
Dashboard Theme	Defines the topic of the HMI dashboard, affects controls and backgrounds: Light: bright skin, especially for day mode. Shiny: similar to the bright skin, color gradient in the controls. Dark: dark skin, especially for night mode.
Select Color	Defines the color of the dashboard header with color gradient, if enabled.
Control Style	Defines the style of the controls: Round: the controls have rounded corners. Flat: the controls have angular corners.
Change default background image	If enabled, a customer-specific background image can be set for the HMI Dashboard. If nothing is defined, the default image is used.
Use Logo	If enabled, a logo is added to the dashboard header. A customer-specific image can be defined for the logo.
Use custom Map Icon	If enabled, a customer-specific map icon can be defined for the map on the start page. If nothing is defined, the default icon is used.

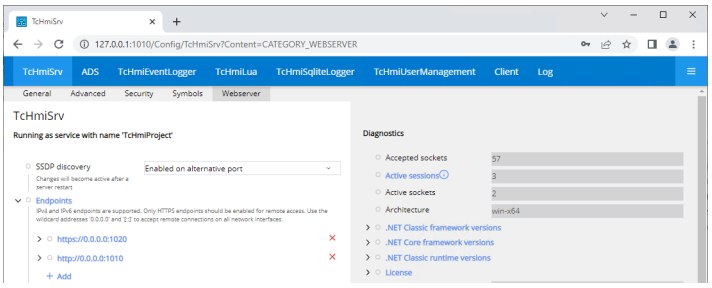
Languages

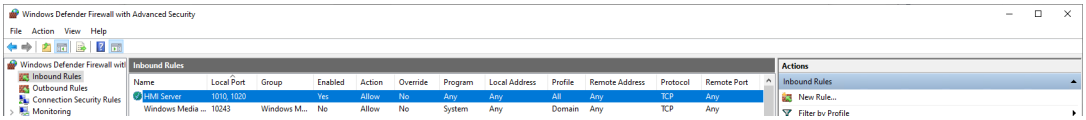
Languages	Enables the enabled language for the language change in the HMI Dashboard. (Only for standard text)

HMI Server

Publish to TwinCAT HMI Server	Publishes the dashboard to a TwinCAT HMI server, if enabled.
Address	Enter the IP address or host name of the TwinCAT HMI server.
Port	Enter the port of the TwinCAT HMI Server (by default 1010 without encryption and 1020 with encryption).
User	Enter the user name.
Password	Enter the password. (The administrator password must be set on the server via the configuration page).
Validate Connection	Press the button to validate your server connection.

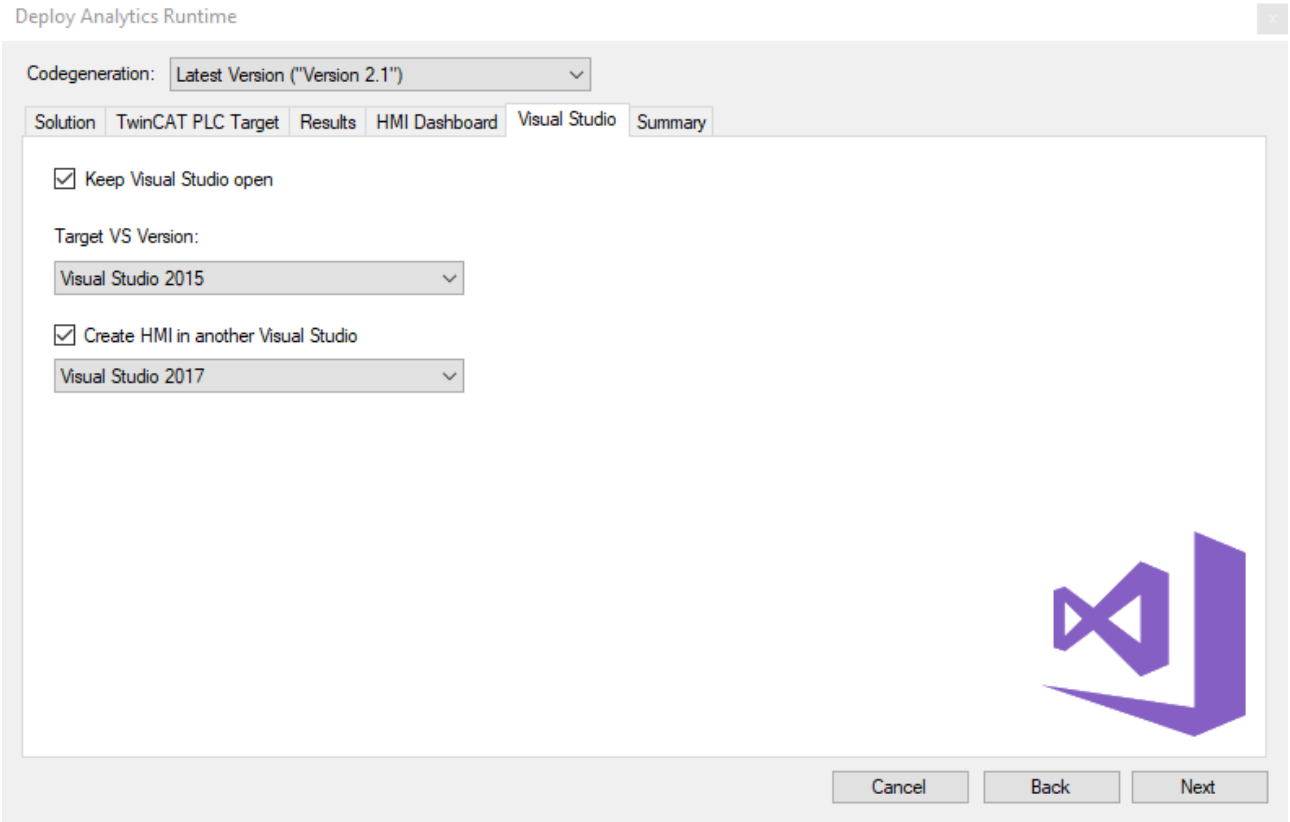
In order to publish to a remote HMI server, the HMI port and IP of the development computer must be shared in the HMI server. Also an inbound rule with the HMI ports must be set up in the Windows firewall from the remote PC.





Visual Studio tab

On the **Visual Studio** tab you can select which version of Visual Studio® or TwinCAT XAE shell is to be used to generate PLC and HMI. In addition, it is possible to generate HMI and PLC in two different solutions.

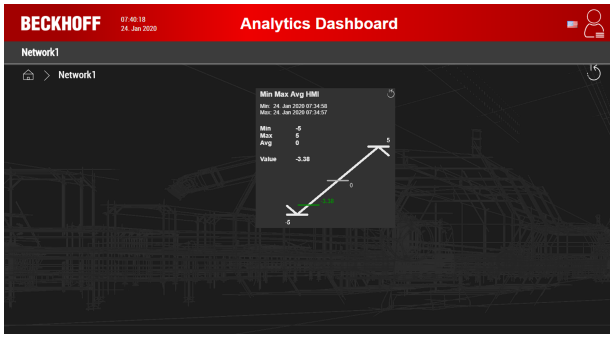


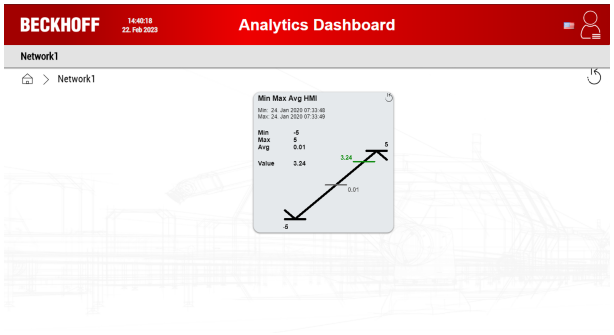
Keep Visual Studio open	Keeps Visual Studio® open after the generation, if checked.
Target VS Version	The target Visual Studio® version for PLC and HMI.
Create HMI in another Visual Studio	If enabled, the HMI is generated into a second solution. A different Visual Studio® version can be selected.
Requirement: TwinCAT or TwinCAT HMI must be installed in the selected Visual Studio® version.	

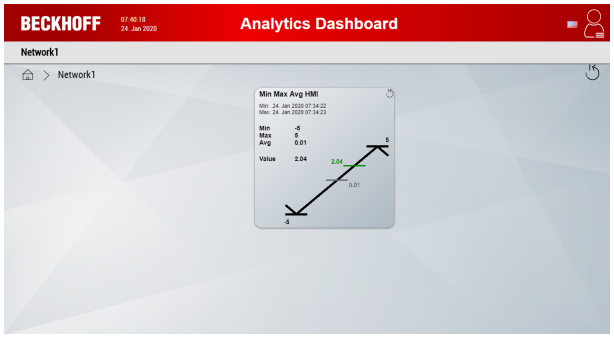
Click **Next** to display the **Summary** tab, then click **Deploy** to start generating the dashboard.

Impressions

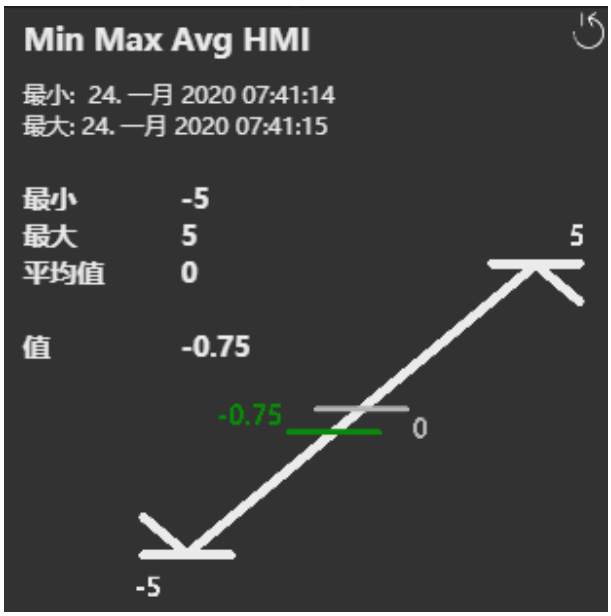
Topics

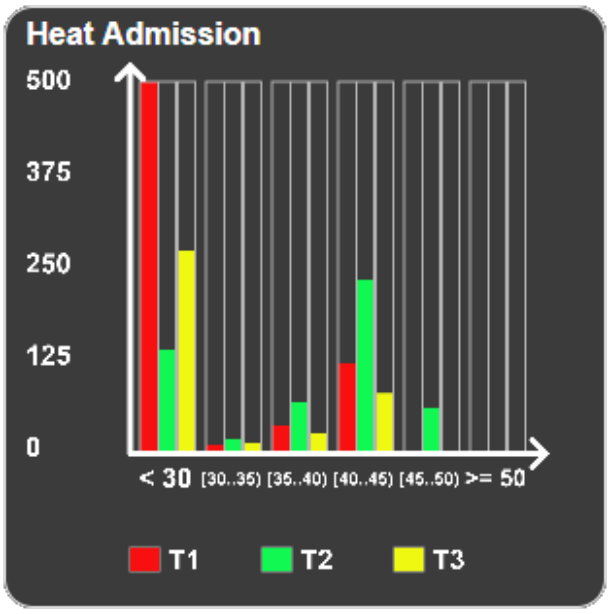




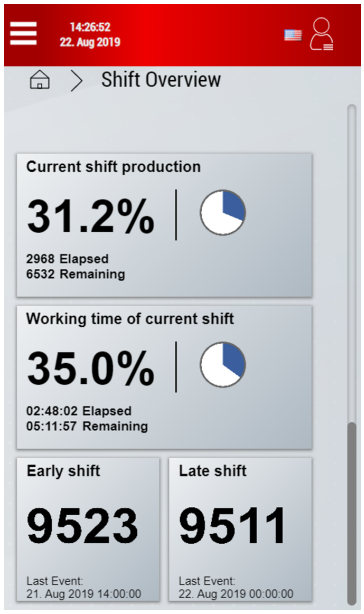


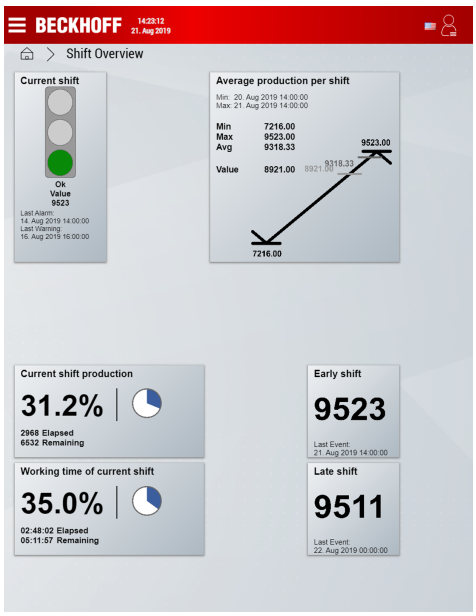
Styles



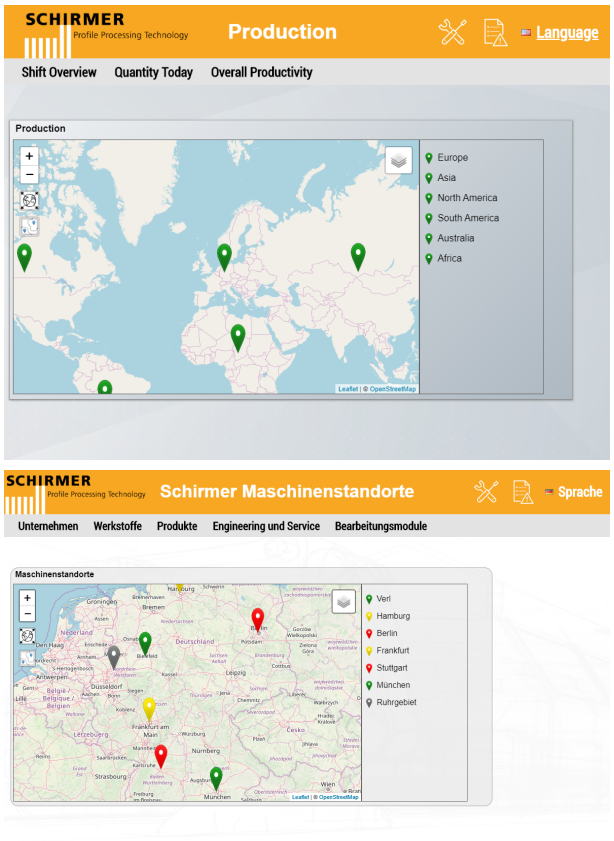


Views for mobile devices

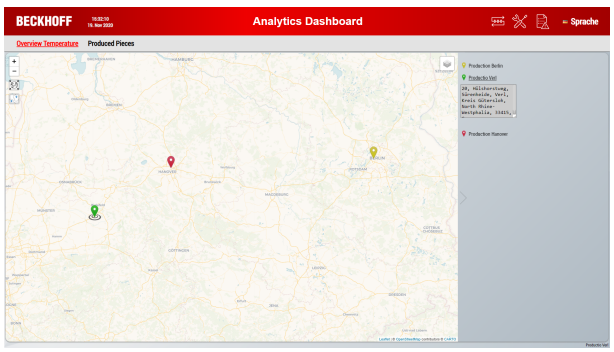


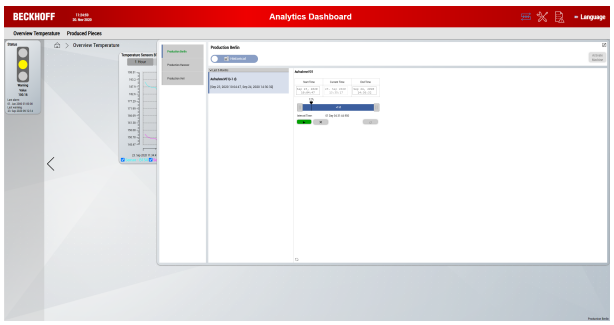


Map on customer-specific dashboard



Historical data and machine switching





6.2.5 Modifying a generated dashboard in HMI engineering

TwinCAT 3 HMI project

The result of dashboard generation is a complete TwinCAT 3 HMI project. Therefore, all options offered by TwinCAT 3 HMI Engineering can be used and included.

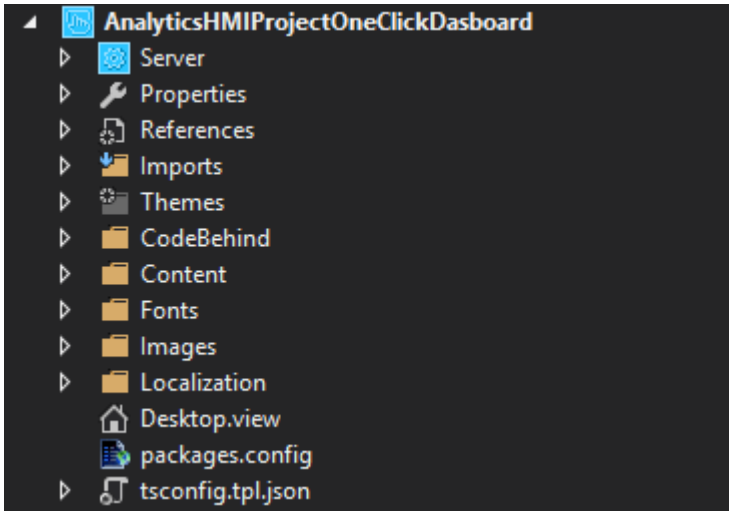


Fig. 1:

Note that many dashboard customizations can also be made in the web browser at runtime.

6.2.6 Configuration of the dashboard at runtime in the web browser

At runtime, different configurations can be made in the web browser:

- Create individual user-specific layouts
- Manage users (see Configuring user management and access rights)
- Customize access rights to contents (see Configuring user management and access rights)
- Changing control properties in the browser

- Customize parameters
- Changing global dashboard options

Note that not all configuration options are available with Dashboard version 1.0.

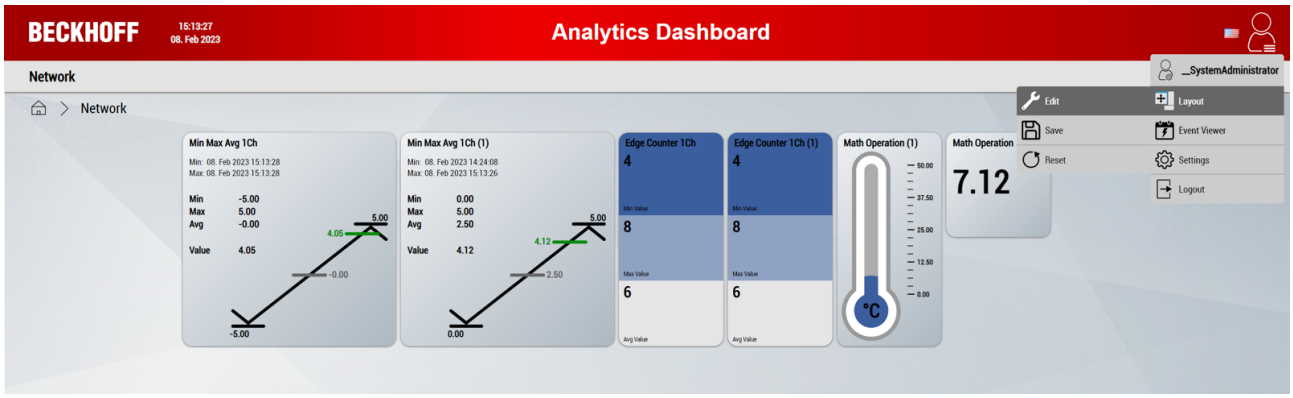
6.2.6.1 Customizing layouts with the Interactive function

With the interactive function, controls can be arranged within a content in a user-specific way. The customized layout is stored centrally in the HMI server extension "TcHmiAnalytics".

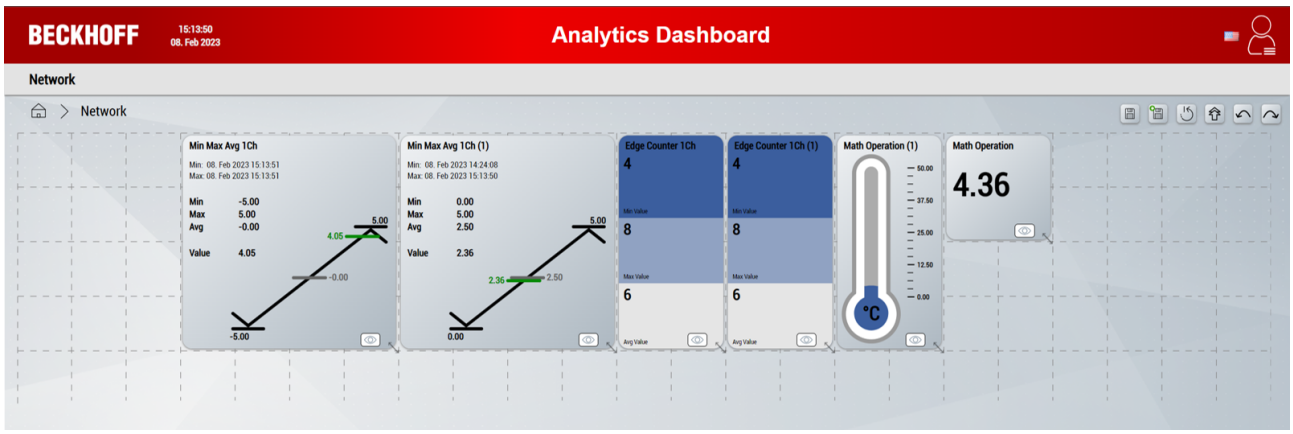
Note that layout editing is available only from Dashboard version 2.0. In addition, the layout can only be edited if the user has the necessary access rights.

Edit mode

Editing a layout is done in a special editing mode. The editing mode can be called via the menu control. Alternatively, the edit mode can also be started with the key combination "CTRL" + "E".



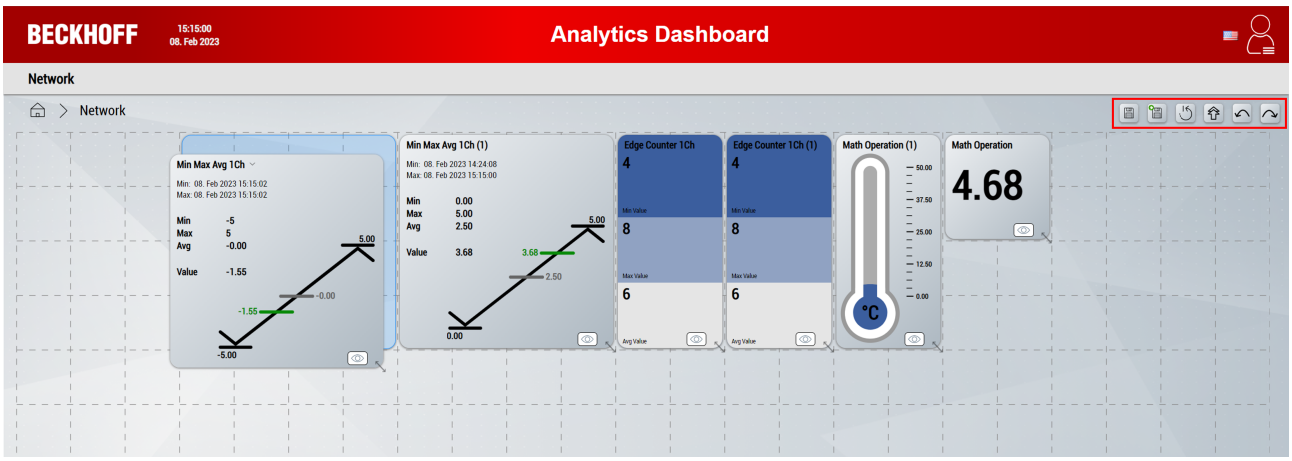
In edit mode, a grid is visible in the background, on which the controls can be arranged as desired.



Edit layouts

The following options are available:

- **Positioning:** For positioning, a control must be clicked or pressed. Thus, it is released from its anchorage and can be moved. The light blue frame indicates the new position within the layout. When the control is released, it anchors itself in its new position.
- **Scale:** Using the arrow in the lower right corner of a control, its size can be changed.
- **Hide controls:** By using the visibility button (bottom right inside a control) it can be shown or hidden. Hidden controls are displayed semi-transparently in edit mode.



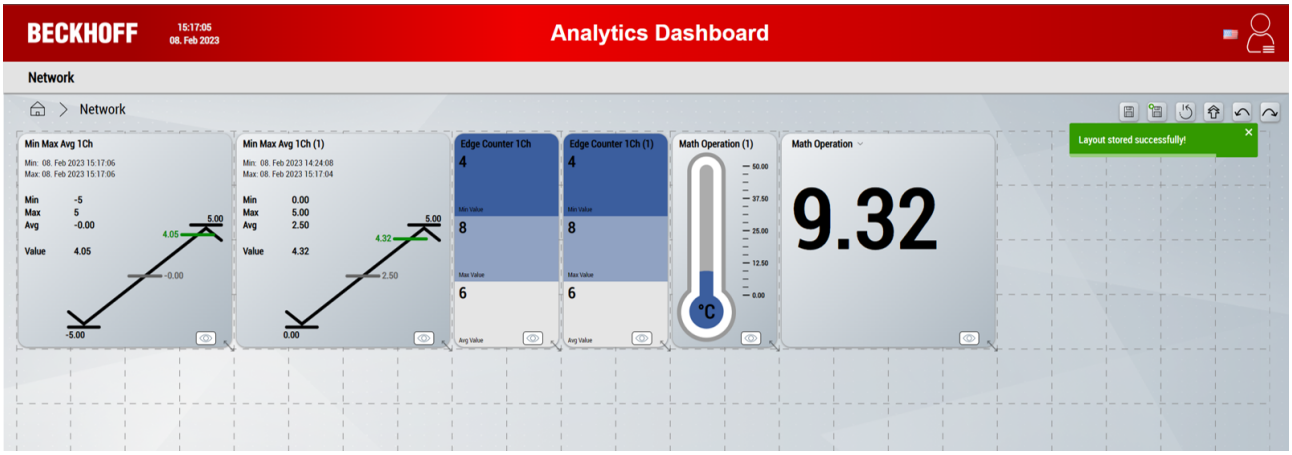
Editing menu

In edit mode, different actions can be performed via an additional menu (outlined in red):

- Save the layout (alternatively key combination "CTRL" + "S")
- Saving the layout and exiting the editing mode (alternatively key combination "CTRL" + "Q")
- Reset the changes made (alternatively key combination "CTRL" + "R")
- Move controls upwards so that free lines are removed (alternatively key combination "CTRL" + "U")
- Undo last action (alternatively key combination "CTRL" + "Z")
- Repeat last action (alternatively key combination "CTRL" + "Y")

Save layouts

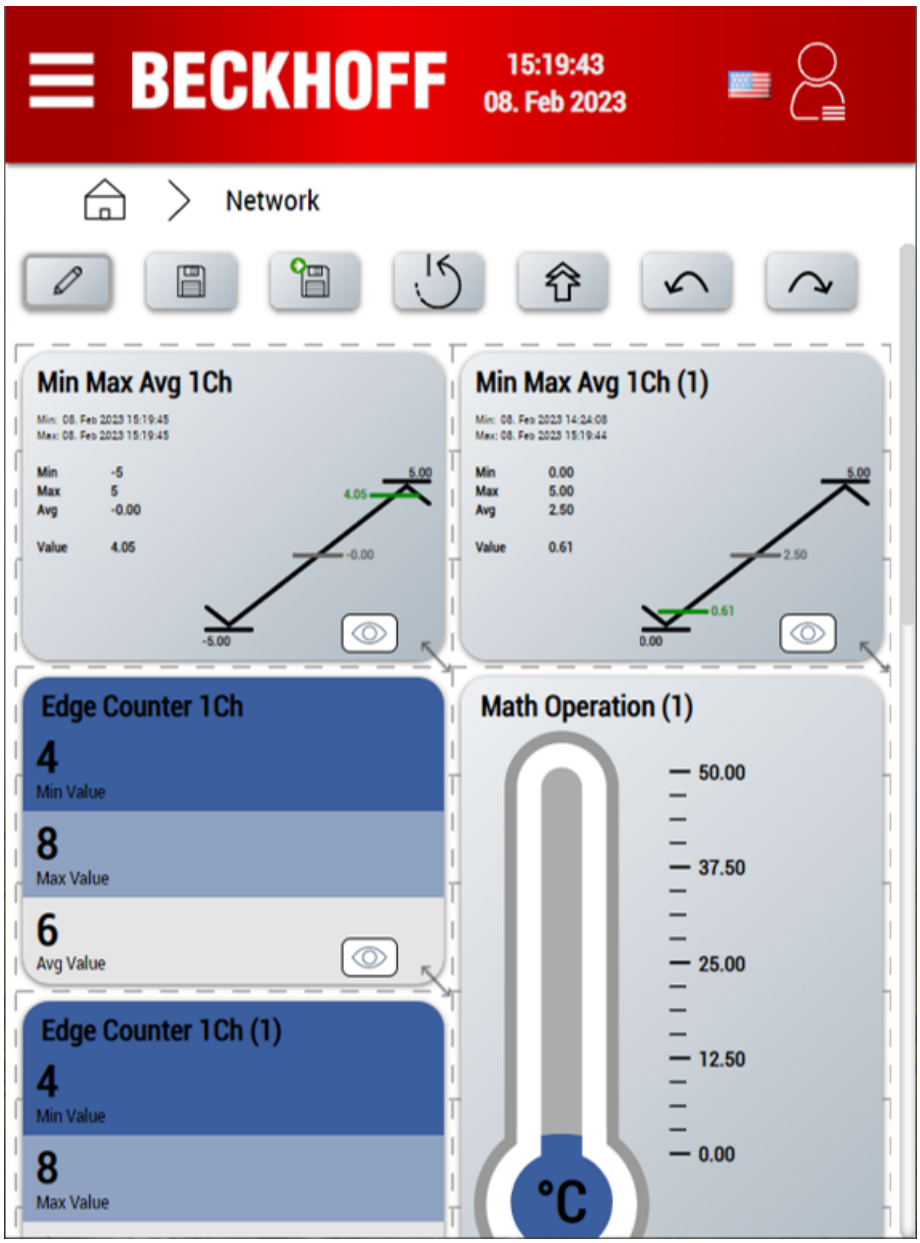
Customized layouts can be saved via the edit menu or via the menu control. A popup shows the result of the save operation.



If an error occurs during saving, check in the server configuration whether the "TcHmiAnalytics" extension is activated.

Mobile layout

For mobile devices, a user-specific mobile layout can additionally be generated. This is independent from the normal layout and has only 2 columns. Editing and saving is done in the same way.



6.2.6.2 Changing control properties in the browser

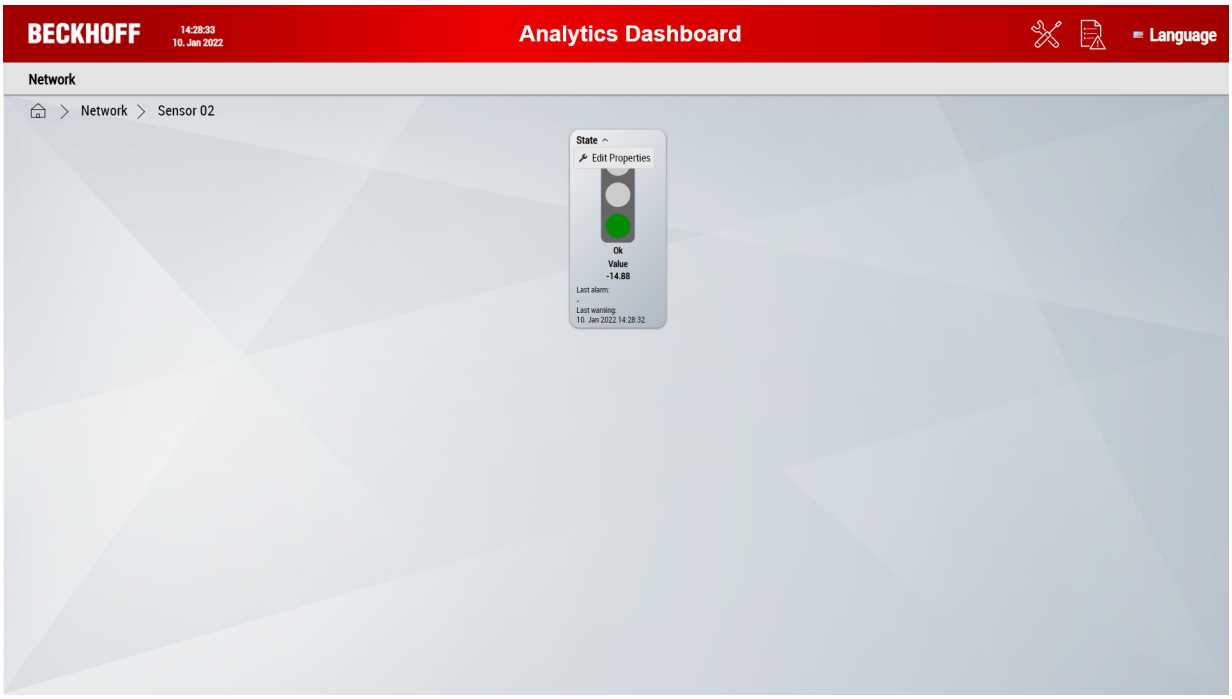
Recommendation: make changes in the Analytics Workbench project

Usually, the changes made here can already be made in the Analytics Workbench before you generate the dashboard (see Mapping Wizard or also Dashboard node). This has the advantage that the changes are saved when the dashboard is generated again.

Each Analytics Control has selected properties that can be changed dynamically in the browser. This option is available from Analytics version 3.4.3145.0 or with Control Package version 1.1.31. You can use it to easily update the NuGet package for older HMI projects.

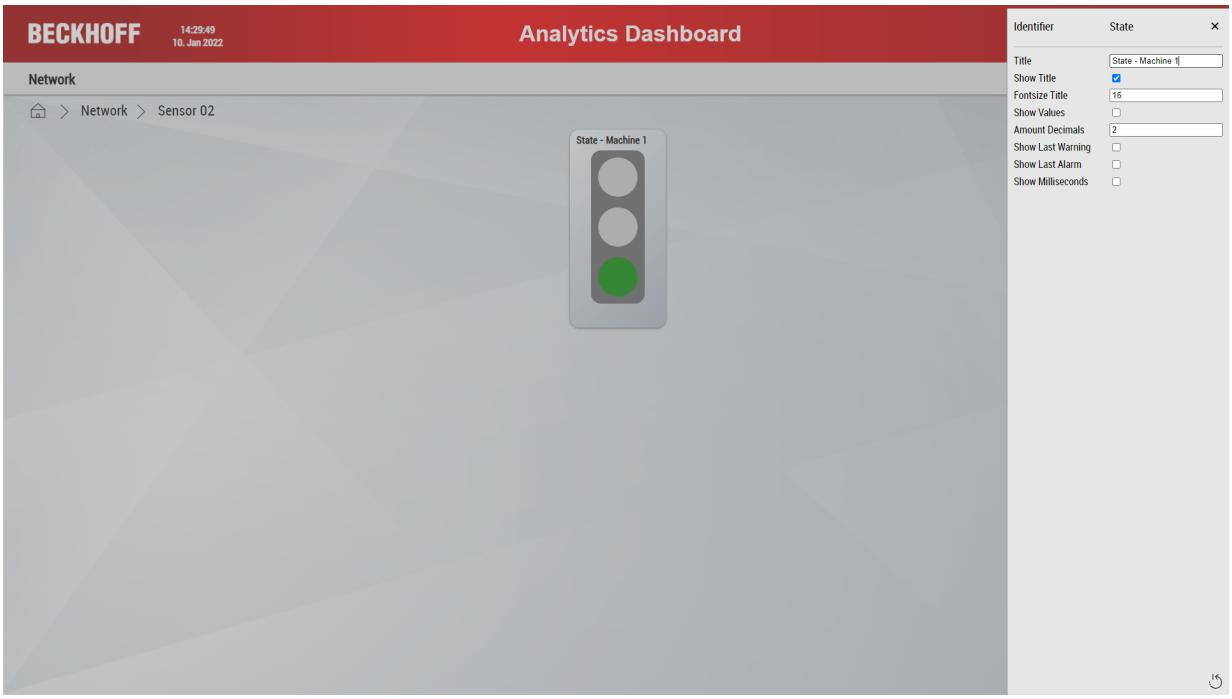
1. Open a generated HMI project and click on the title of a control where you want to change a property.

2. There a menu opens where you can click on **Edit Properties**.



⇨ Each control has different properties that can be customized. Here, for demonstration purposes, the properties of the *Traffic Light* control are changed

3. Change the title and uncheck the bottom three selection boxes to display only the traffic light with the changed title.

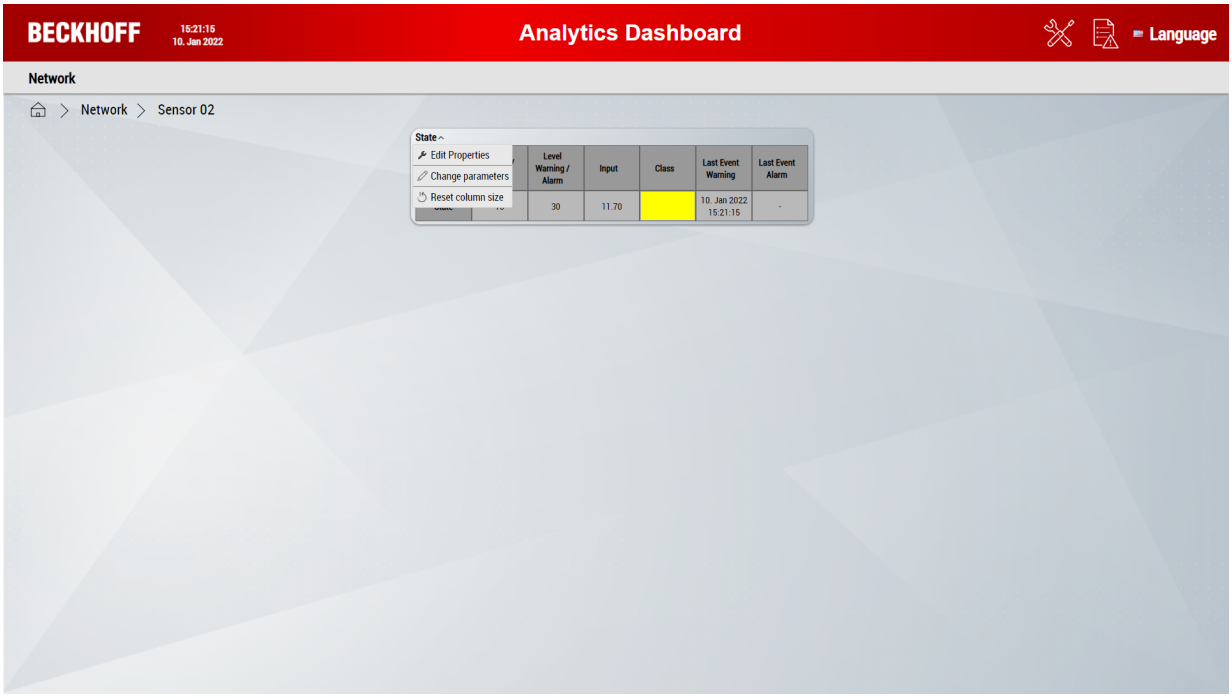


⇨ The changes made are saved client-side (on the device where you open the dashboard) and are only displayed there. In the future, storage will be performed via the HMI server, which means that changes will be made globally on each device.

6.2.6.3 Changing parameters in the browser

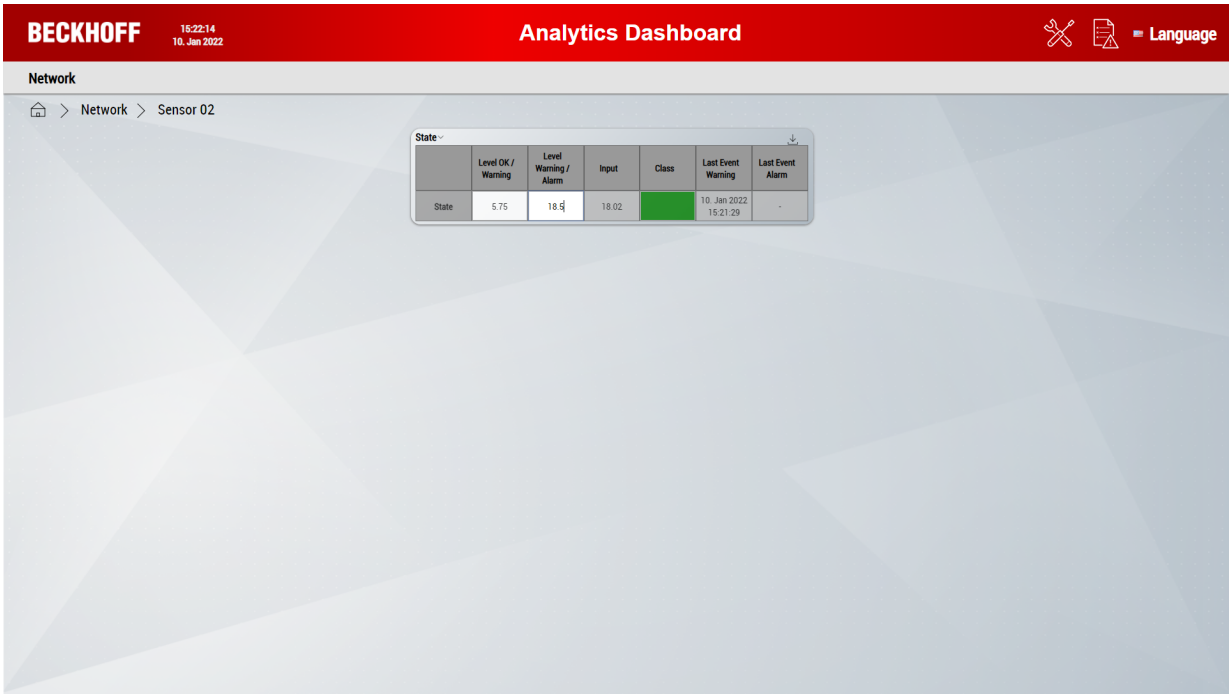
Numerical parameters such as limit values can be changed dynamically in the dashboard. This is possible with the Data Table Control. The values are persistently written back to the PLC there and are thus also available after a restart.

1. Open a generated HMI project and click on the title of the table control where you want to change a parameter.
2. There a menu opens where you click on **Change parameters**. A module without parameters does not have this menu item.

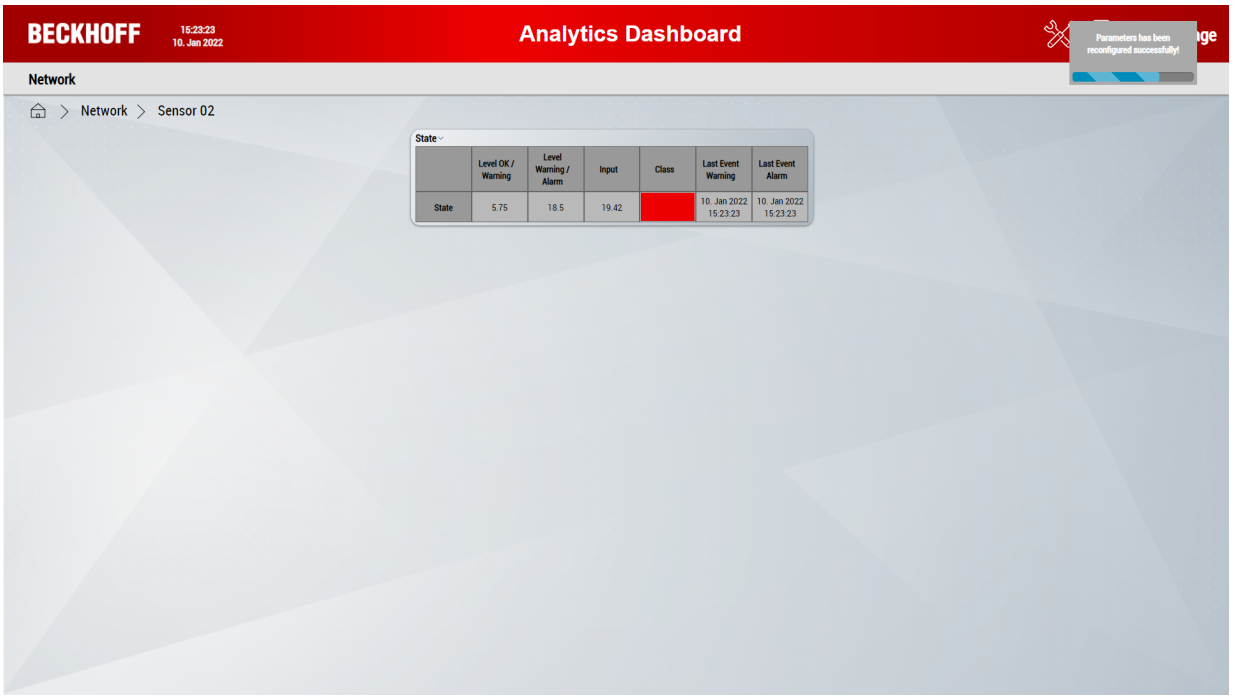


⇨ The changeable fields change to an input field with a white background.

3. Change the values there and then click the save icon in the top right corner.

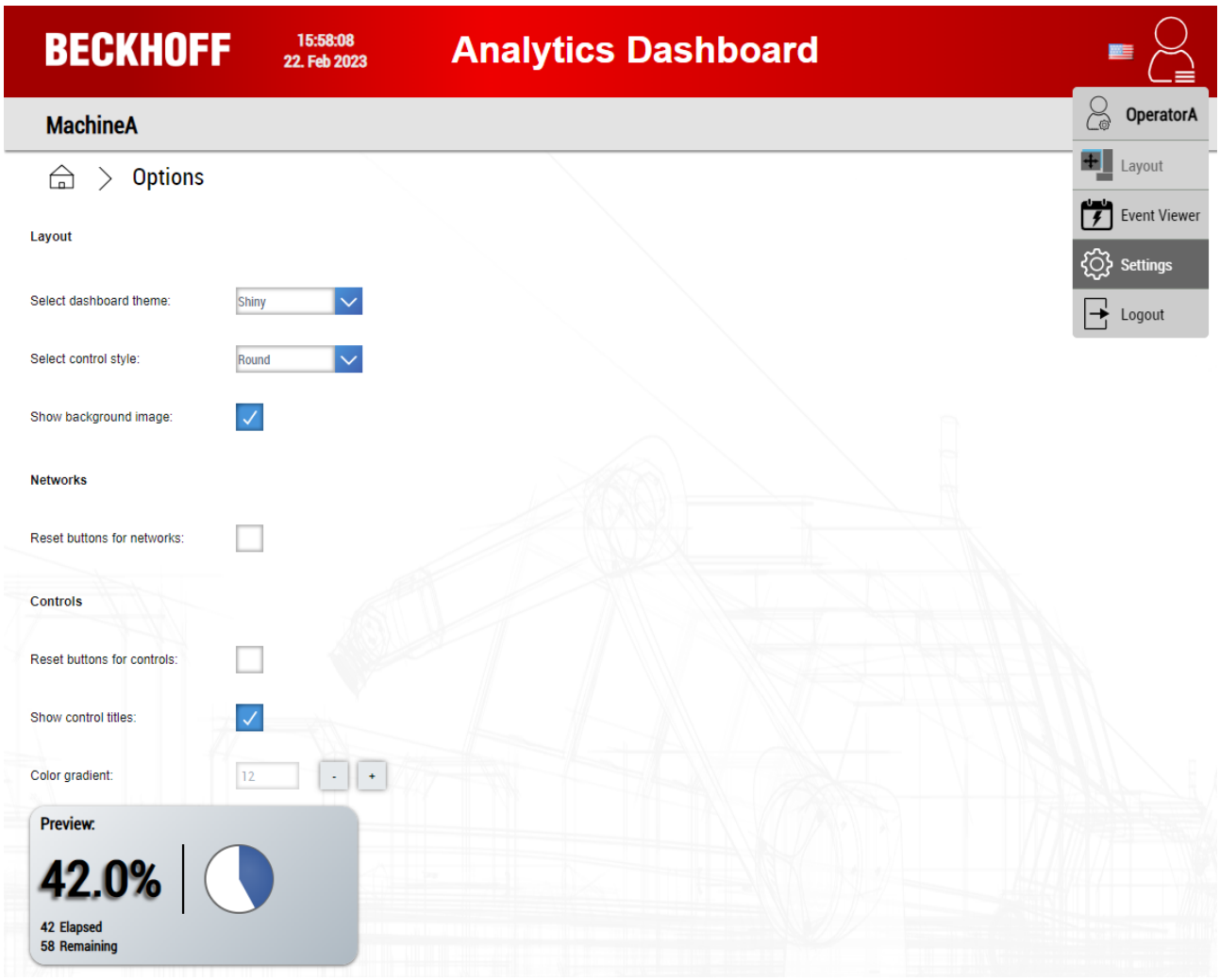


⇨ After saving the parameters, you will see a message in the upper right area of the dashboard. There you will be told if the reconfiguration worked.



6.2.6.4 Customizing global dashboard options

On the Options page, general settings for the dashboard can be edited.



Layout

Select dashboard theme	Change the dashboard theme between shiny, dark and light.
Select control style	Change the control style between flat and round.
Show background image	Displays the standard or customer-specific background image.

Networks

Reset buttons for networks	Enables reset buttons for entire networks. Reset all controls within the selected network.

Controls

Reset buttons for controls	Enables reset buttons for controls. Reset a single control.
Show control titles	Enables control titles for all controls.

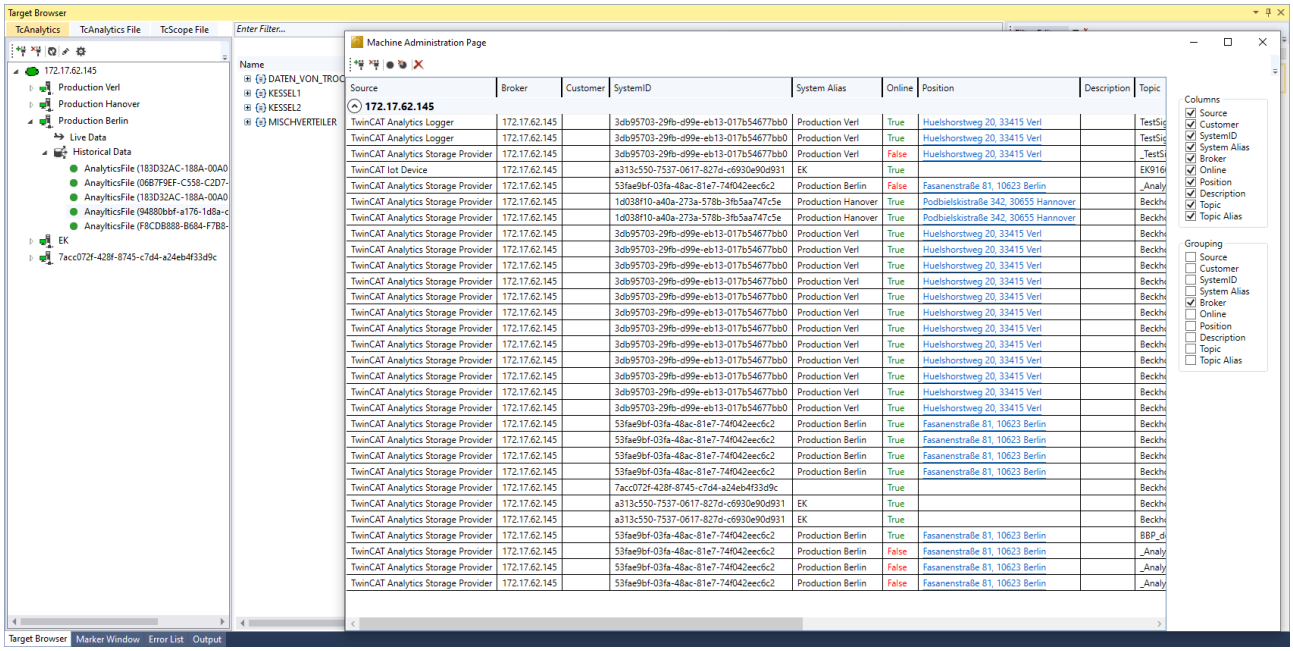
6.2.7 Switching multiple machines in the HMI Dashboard

In TwinCAT Analytics you can use different data streams from several machines, which you can switch in an analysis. This is possible both in the Analytics Workbench via the Virtual Input Source and in the fully generated PLC and HMI. Both live and historical data can be used. For each data stream, you can add a brief description and the location either in the Analytics Logger or in the Machine Administration [▶ 27].

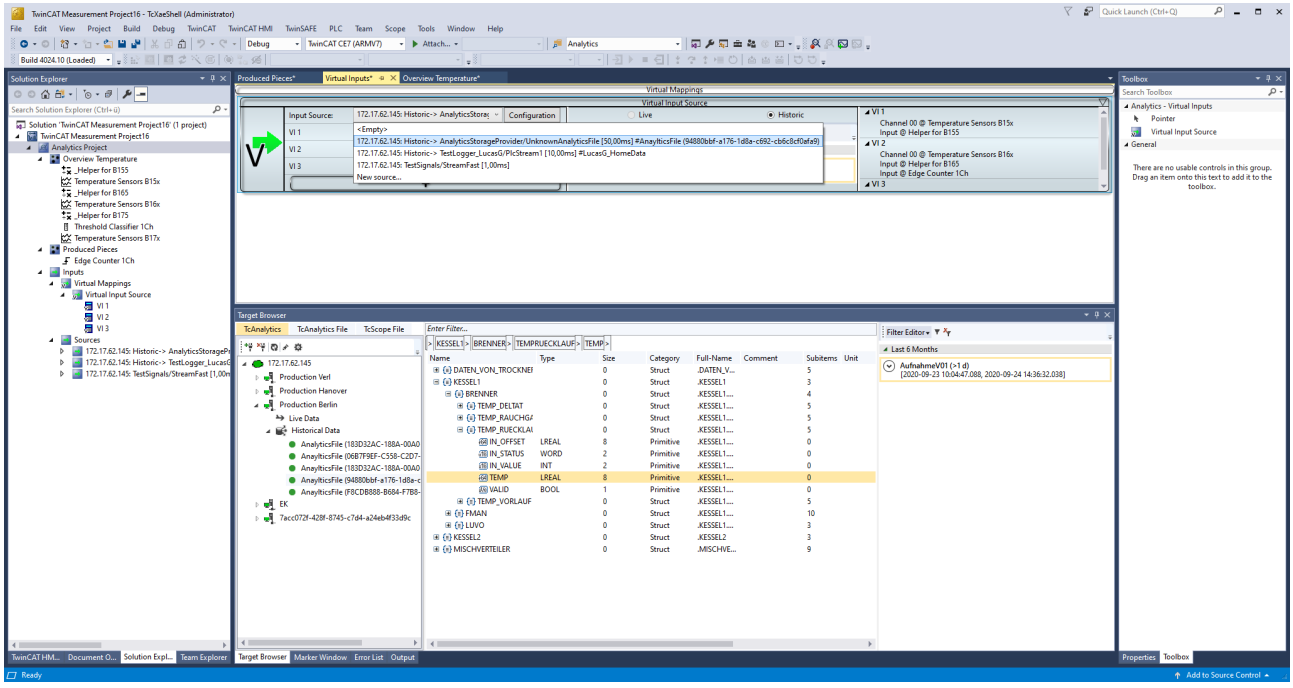
Machine Administration page

If not already done in the Overview, set the metadata of your machine on the Analyse data [▶ 27] page. Open the TwinCAT Target Browser (**TwinCAT > Target Browser > Target Browser**) and click the gear icon. Now you can enter the location of your machine, a short description and the name of your machine. Note that existing data in an Analytics project is not assigned this meta information, since the information is only transferred from the Target Browser during drag and drop. In this case, you can delete the data stream and recreate it.

For each data stream you use in your Analytics configuration, the Runtime deployment [▶ 43] creates a map entry. These map entries are used as input variables for the general map on the start page of your HMI dashboard.

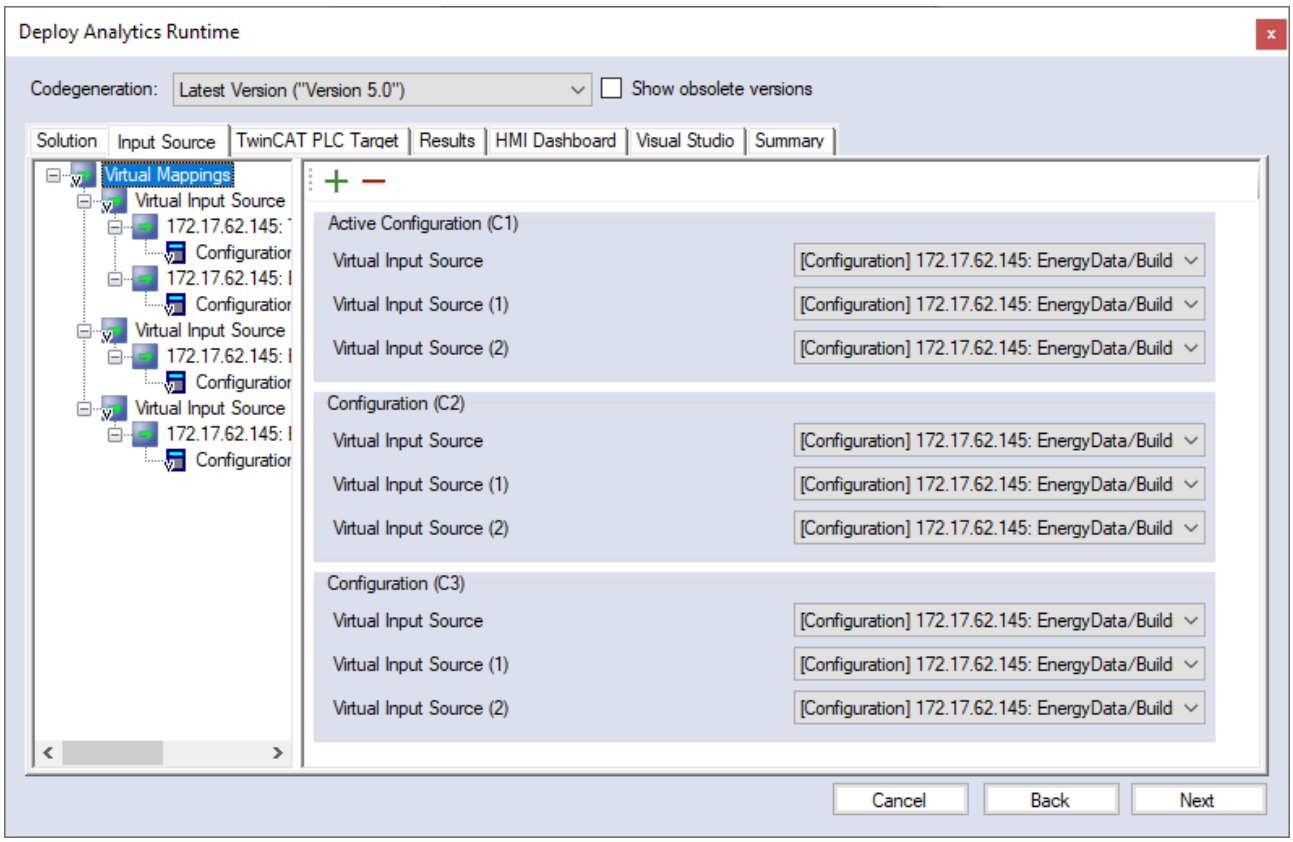


In the Analyse data [▶ 27] the data streams are listed under **Sources**. These are created by using data from various sources from the TwinCAT Target Browser. These sources are listed in the Virtual Input Source. It is possible to switch between the sources. Check that all individual inputs are linked and that none is set to **Empty** by clicking each source once (which corresponds to a switch in the workbench). You can then generate an HMI with PLC.

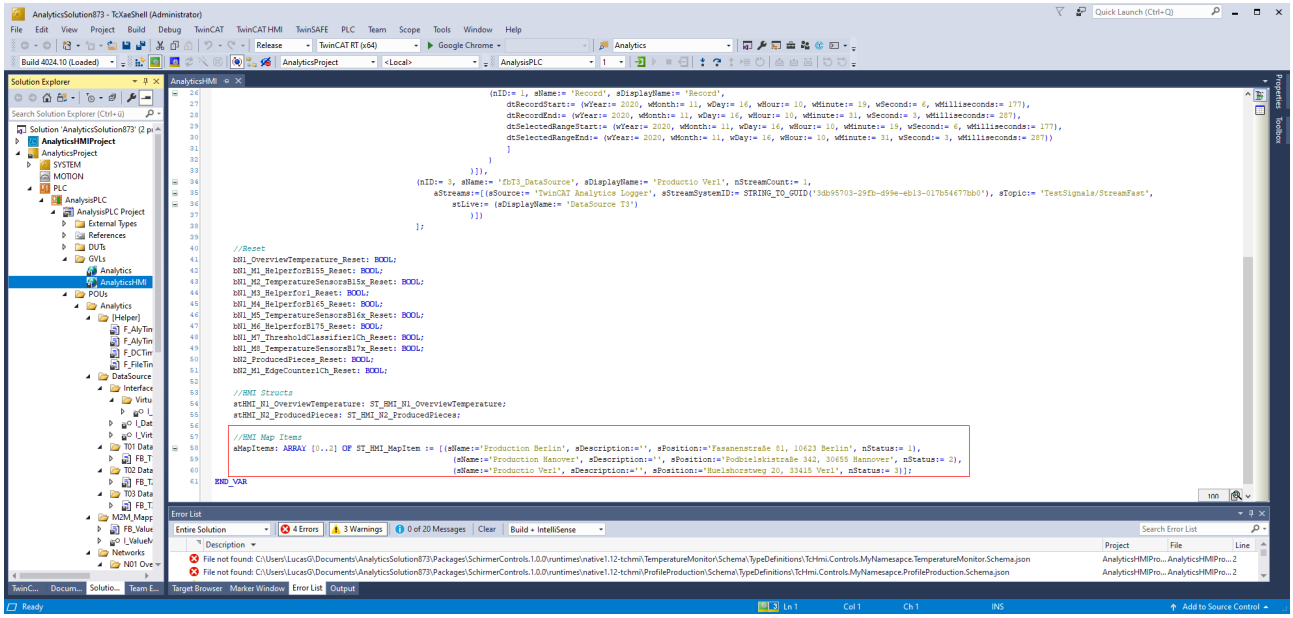


Deploy Wizard

In Runtime deployment [▶ 43], any number of configurations can be created at **Input Source**, which can be switched in the HMI. This makes it possible in the analysis HMI to analyze and switch live as well as historical data from possibly different machines. Each of the listed configurations can be analyzed in parallel. A configuration always has as many Virtual Input Sources as are configured in the Analytics Workbench.

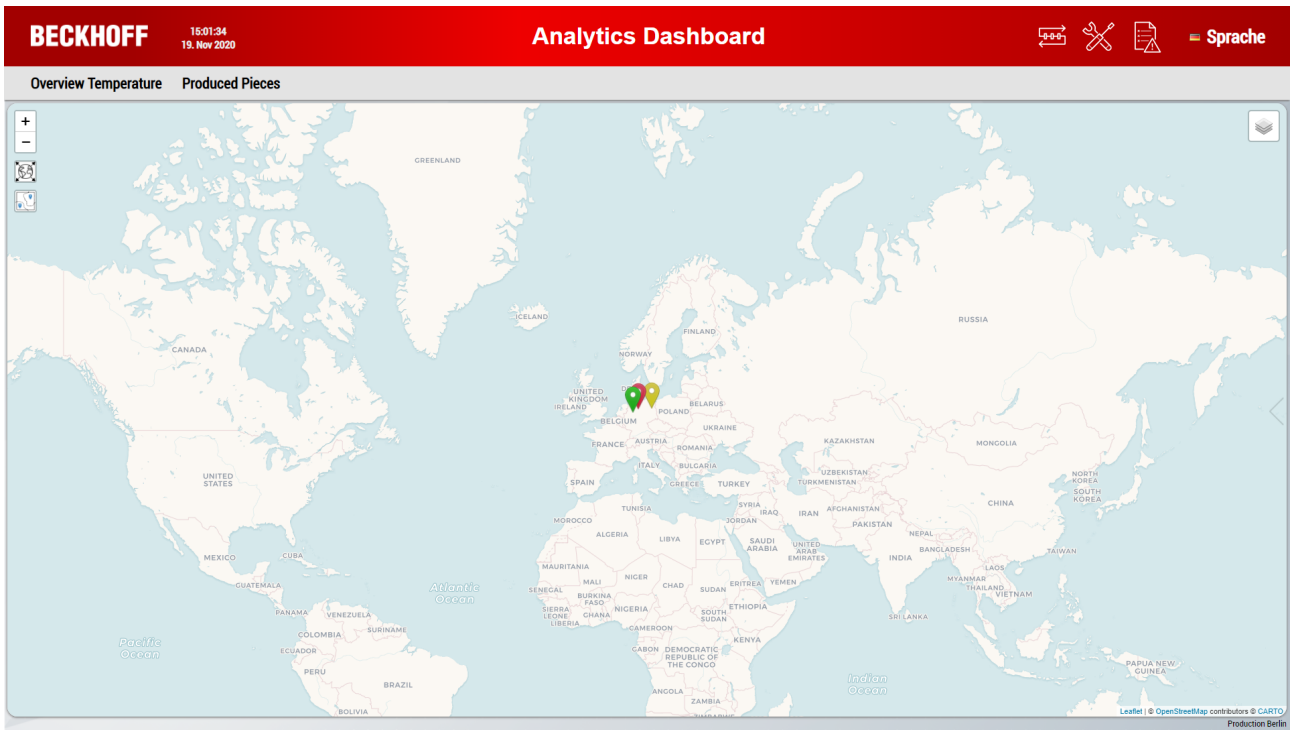


After a successful HMI and PLC generation, the names, positions, and descriptions of the input sources are entered into the global variable list of the PLC. If something is not right or you want to change a value afterwards, you can do this directly in the PLC.

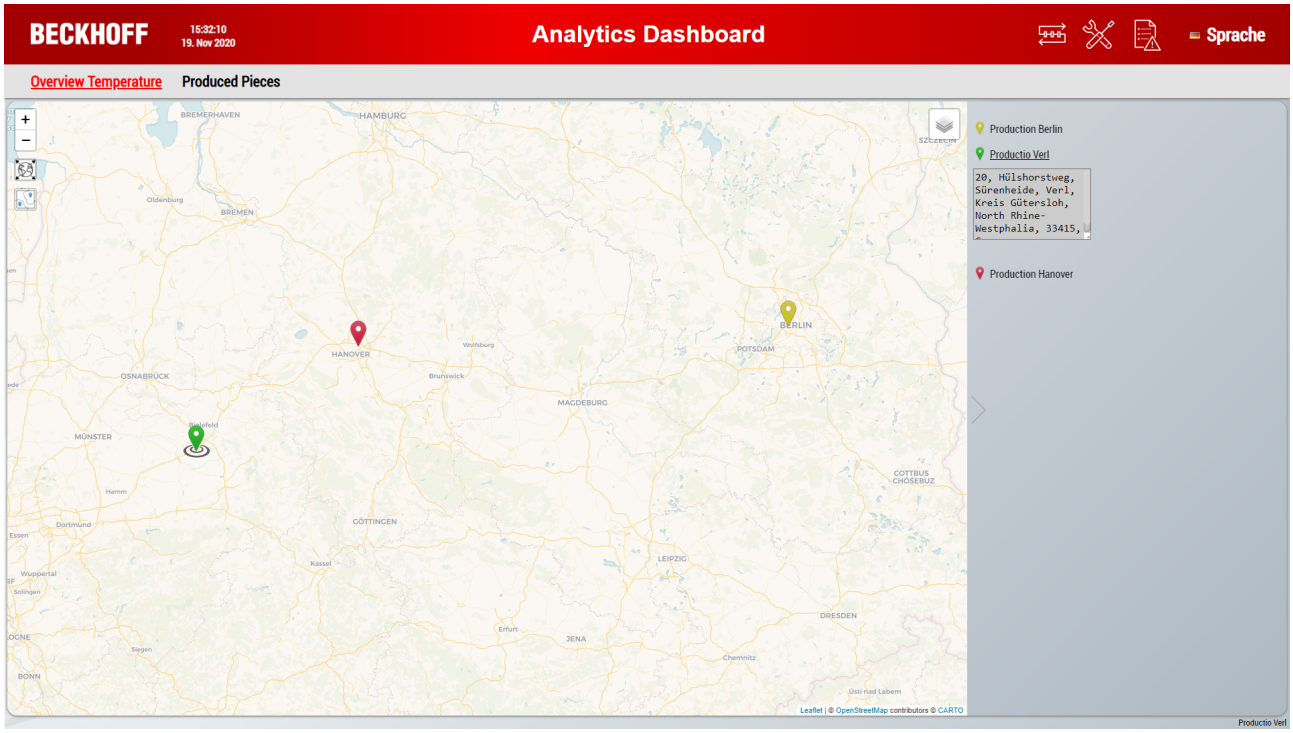


Map with machines

If you selected to create a start page, the *Map Control* will be created on the homepage of your HMI Dashboard. The map shows all machine locations and lists the names of the machines (system alias) in the legend. The legend can be opened and closed using the arrow on the right. The icon color indicates the current machine status: green = OK, yellow = Warning, red = Alarm.

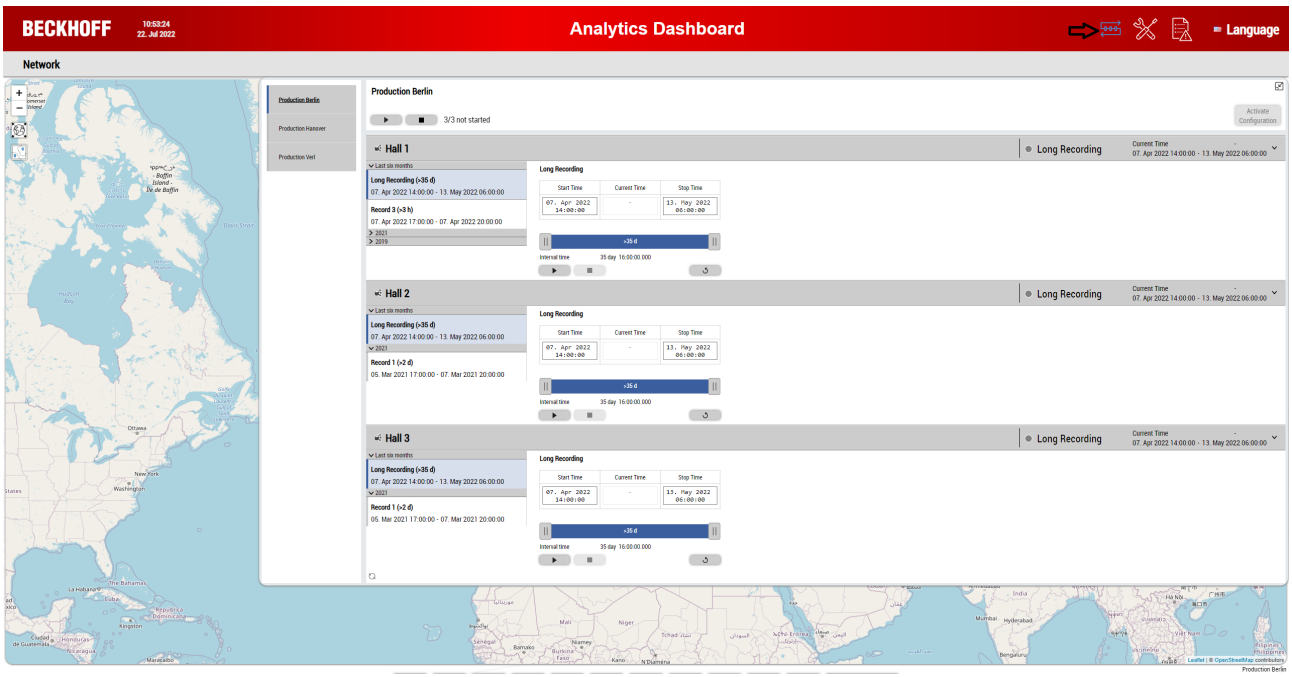


You can zoom in on a single icon by double-clicking it and zoom out to an overview of all machines using the icons on the left. It is also possible to click on an icon on the map. The corresponding legend entry is automatically highlighted. This also works the other way around, i.e. when you click on a legend entry, the corresponding icon is automatically highlighted.



Switching machine data

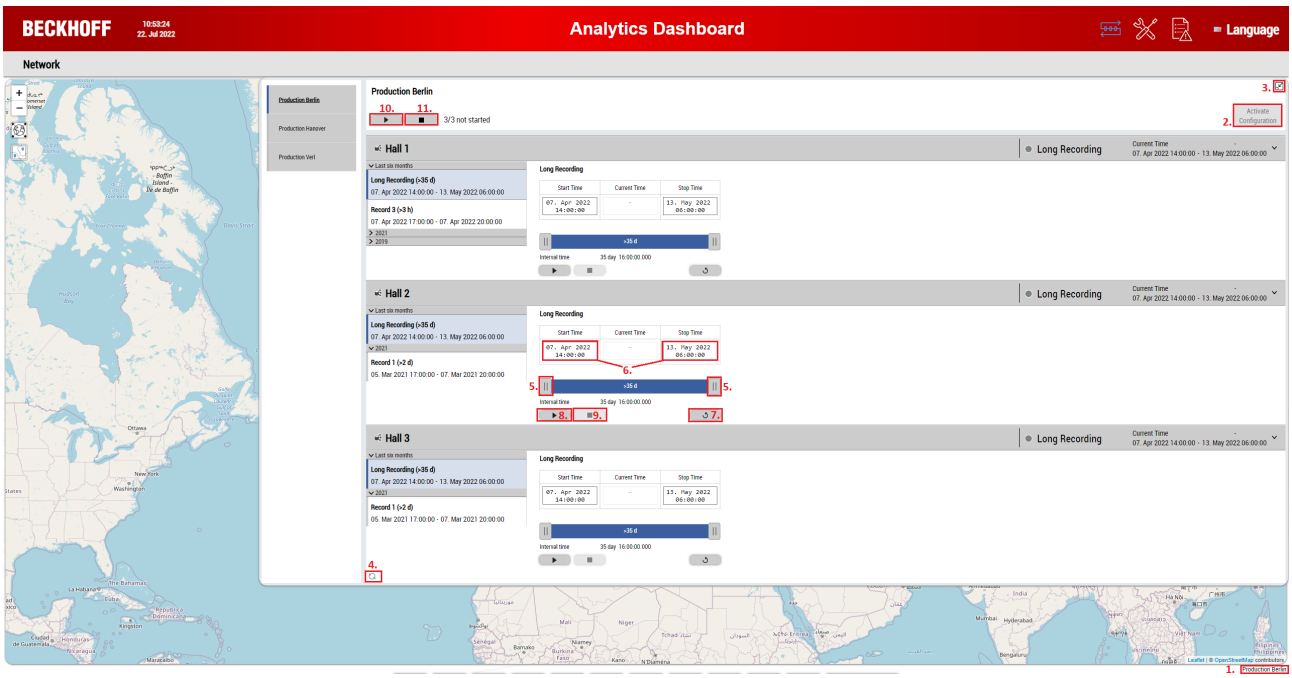
In addition to the map with the individual machine configurations, the analyses for the individual configurations can be switched. In the PLC all analyses run in separate tasks, which means that all analysis data is available at the same time. In the generated HMI, the various machines can be switched over in a specially developed control. The control can be opened and closed via the icon highlighted in the image below (black arrow).



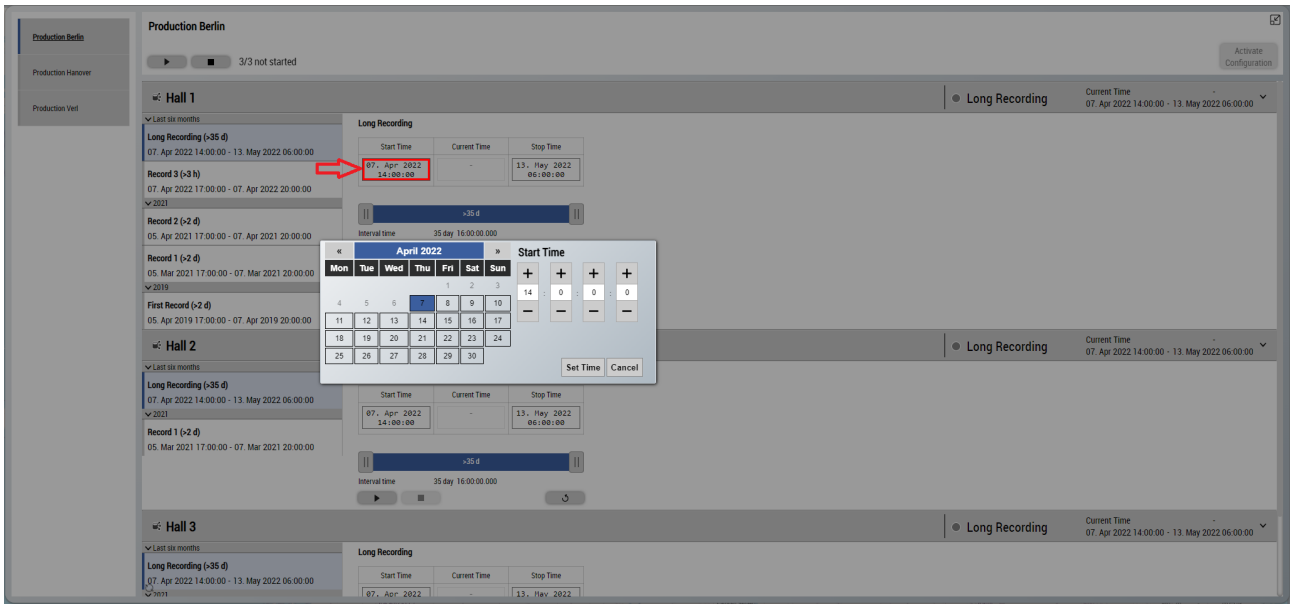
The configurations that were previously shown on the map with the locations can be selected in the control. A configuration can consist of live and historical data, whereby only historical data is configured for the **Production Berlin**. Historical data can be analyzed equivalently as in Working with Historical Data in the HMI.

When clicking on another machine, this configuration is only displayed in the control. To activate, click **Activate Configuration**.
The control is explained below.

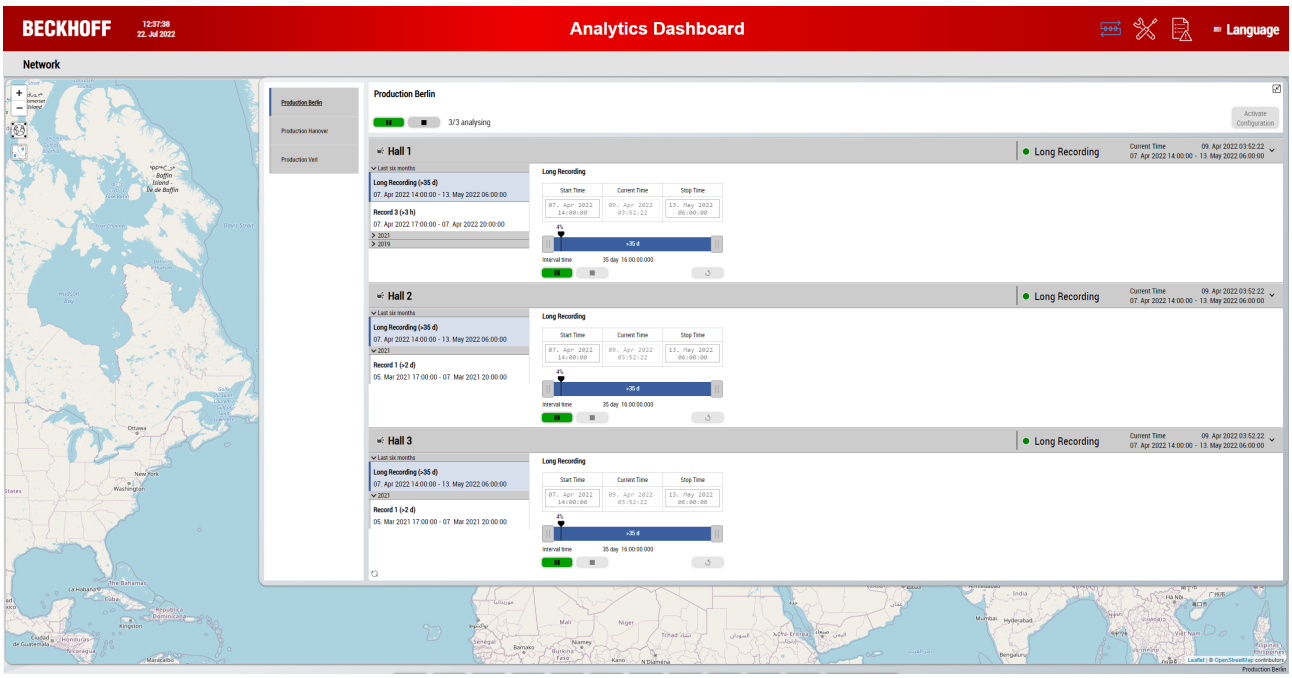
1. The currently active configuration.
2. Button to activate the selected configuration.
3. Selection window to minimize (and automatically activate the selected configuration).
4. Reloading the record list. This allows data recorded subsequently to be dynamically loaded and analyzed in the dashboard.
5. Selection of the start and end times via a slider.
6. Selection of the start and end date via a selection window with calendar.
7. Resetting the start and end times.
8. Starting the analysis of the selected **Input Source**.
9. Cancel the analysis of the selected **Input Source**.
10. Starting the entire analysis
11. Cancel the entire analysis



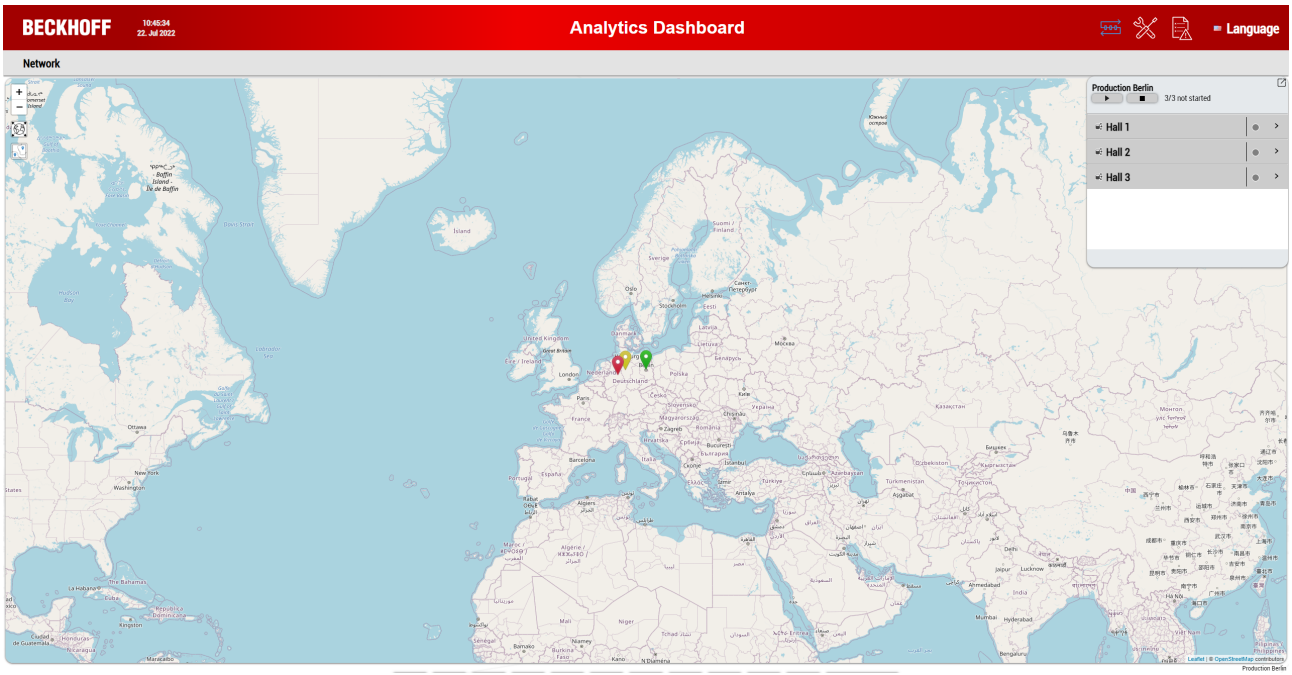
The slider is ideal for quick tests and approximate setting of the start and end times. For an exact setting down to the millisecond, click on the **text field**. This opens a selection window with calendar for selecting the date and setting the exact time.

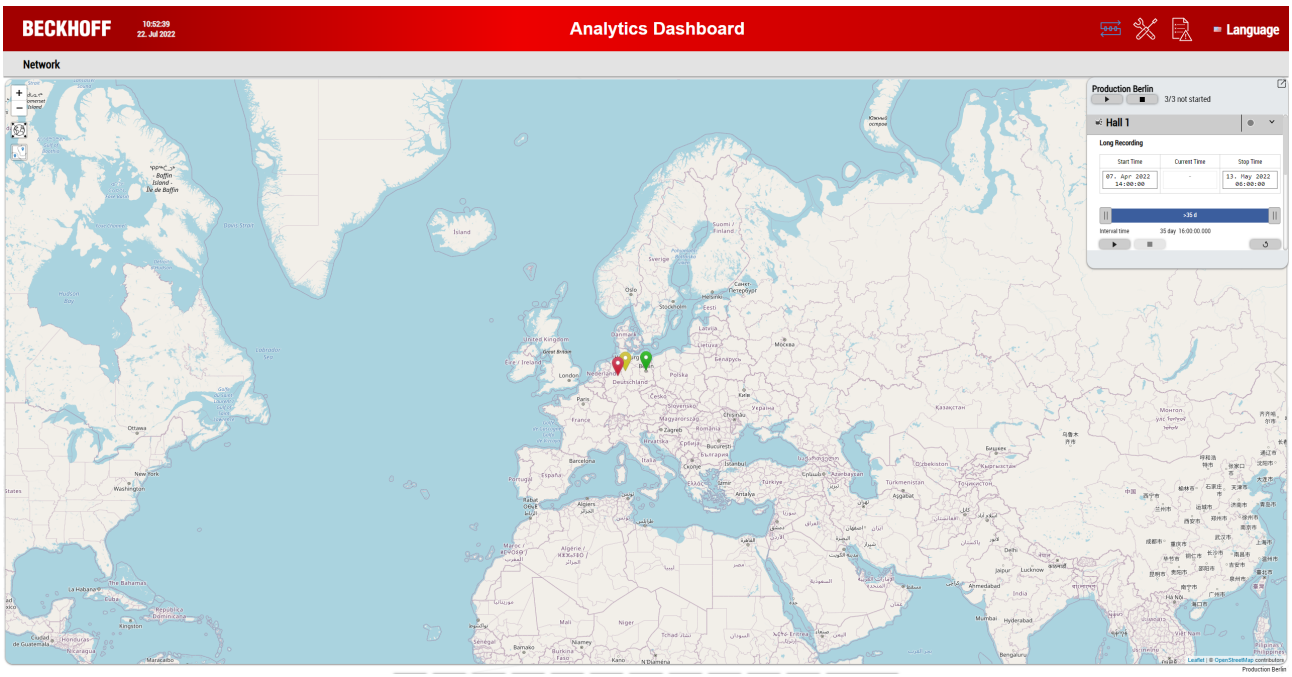


As soon as you click the button **Start**, the analysis is triggered with all input sources in the PLC. A loading symbol appears for the first time until the analysis starts. It turns green when the first data appears. Likewise, a marker shows the current time of the historical analysis in the slider. The process speed depends on the bandwidth, the number of data and the recorded cycle time. It is possible to analyze several historical recordings at the same time by simply switching machines. Internally, all analyses continue in parallel. This makes it easy to switch between analyses in the dashboard.



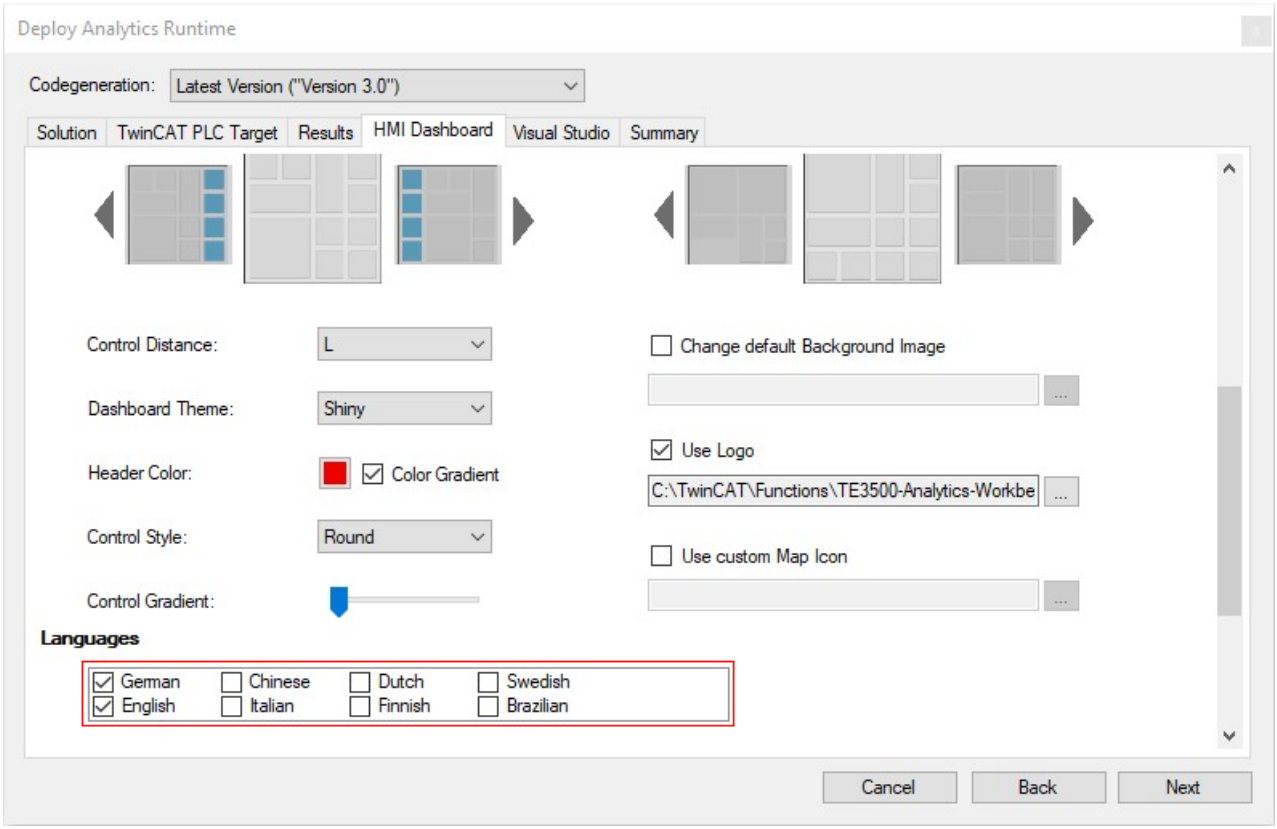
The button **Minimize** in the upper right corner hides the selection of configurations and records. This allows you to work with the records from the input sources and view the data at the same time. You can hide the window completely by clicking the blue icon at the top. Only the configuration name is always in the bottom right corner of the dashboard.





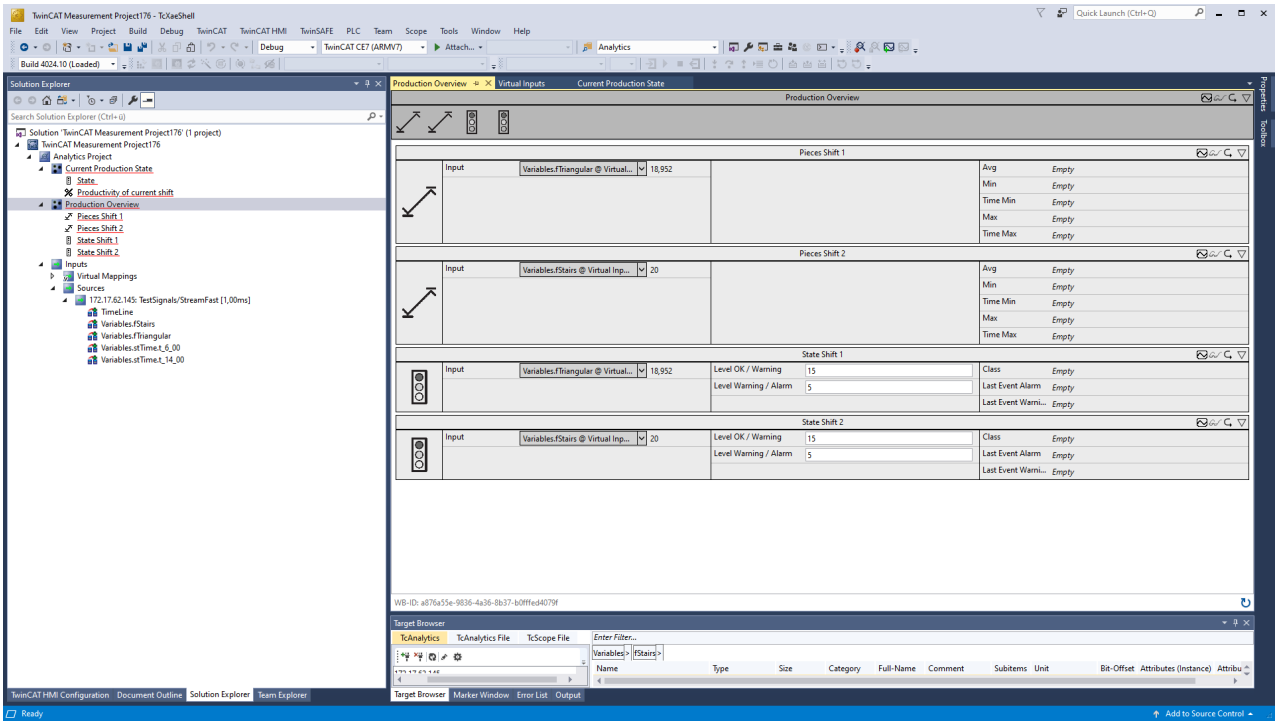
6.2.8 Integration of a language switch

In the Deploy wizard configuration window, up to 8 languages can be selected for language switching in the HMI area.

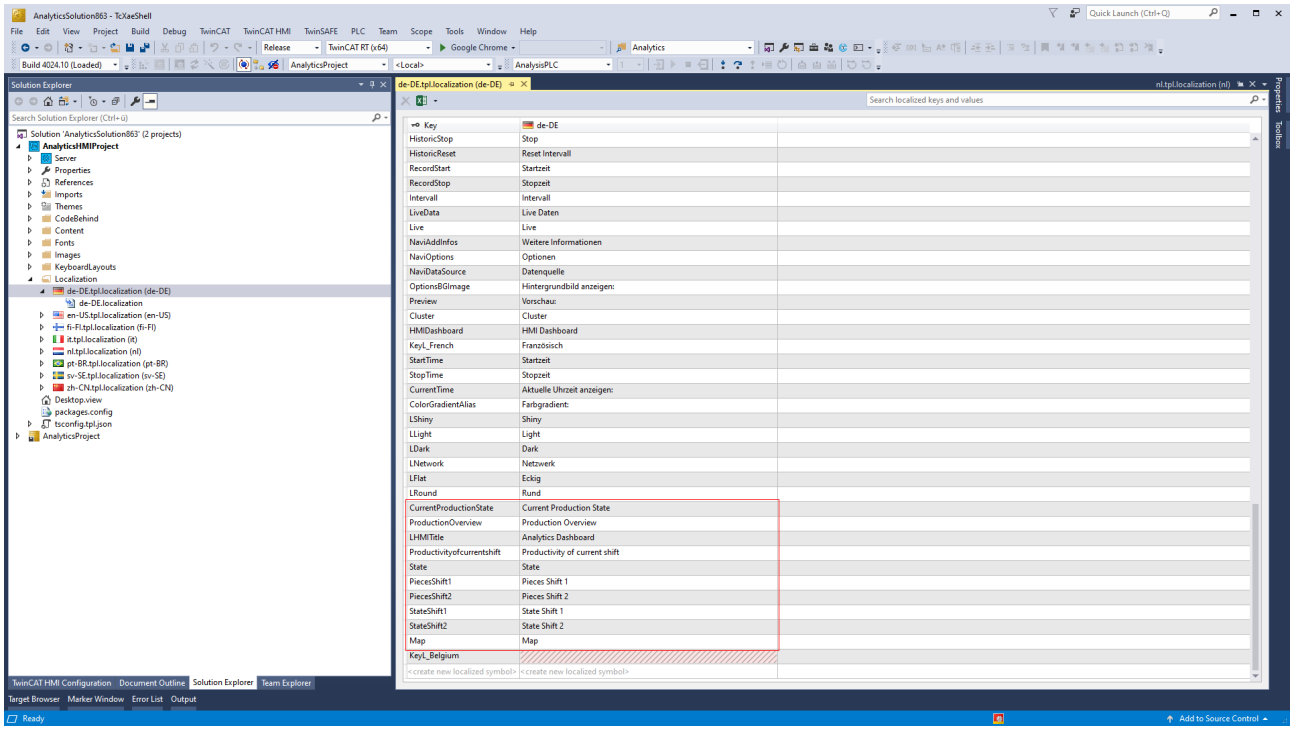


If you use the user administration, the languages of the created users are already selected. These cannot be deselected.

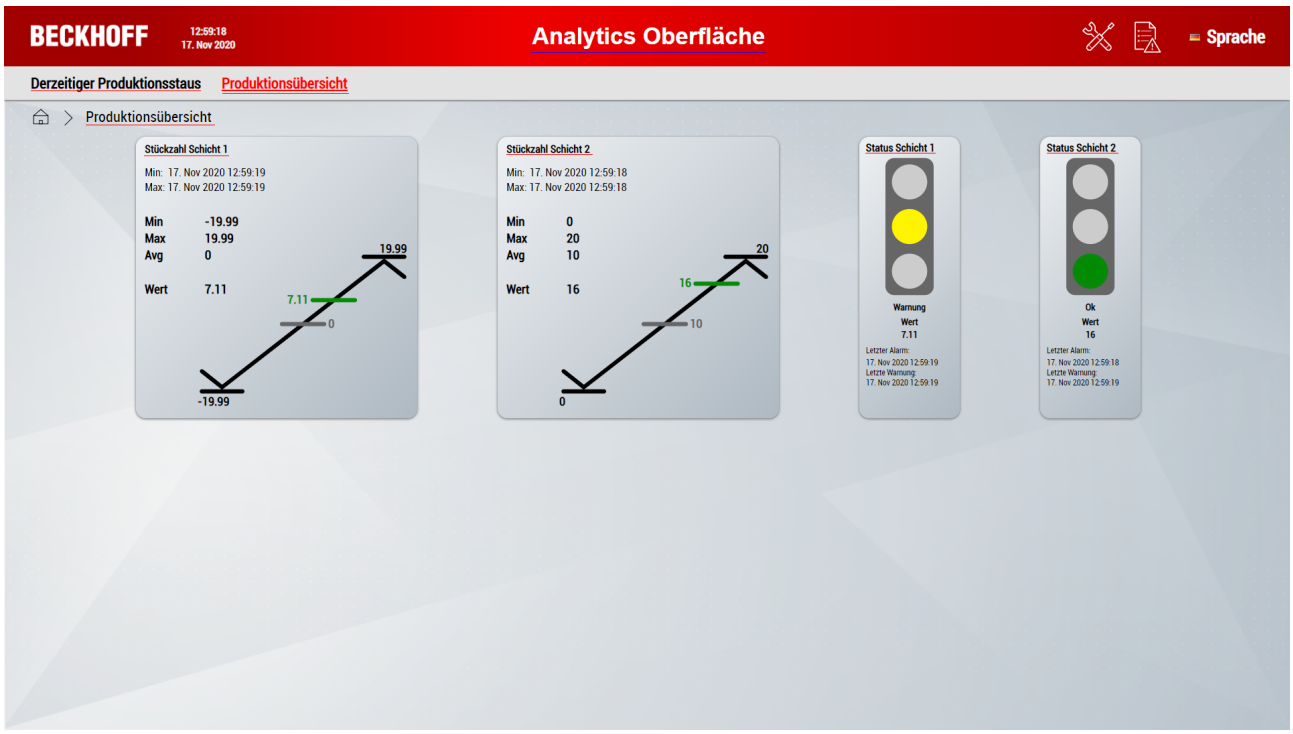
The texts in the supplied Analytics Controls and all other texts can be switched automatically. Only your network and module names have to be translated, if you want them to be included in language switching. In the following screenshot, the main affected names are marked in the **Solution Explorer**.



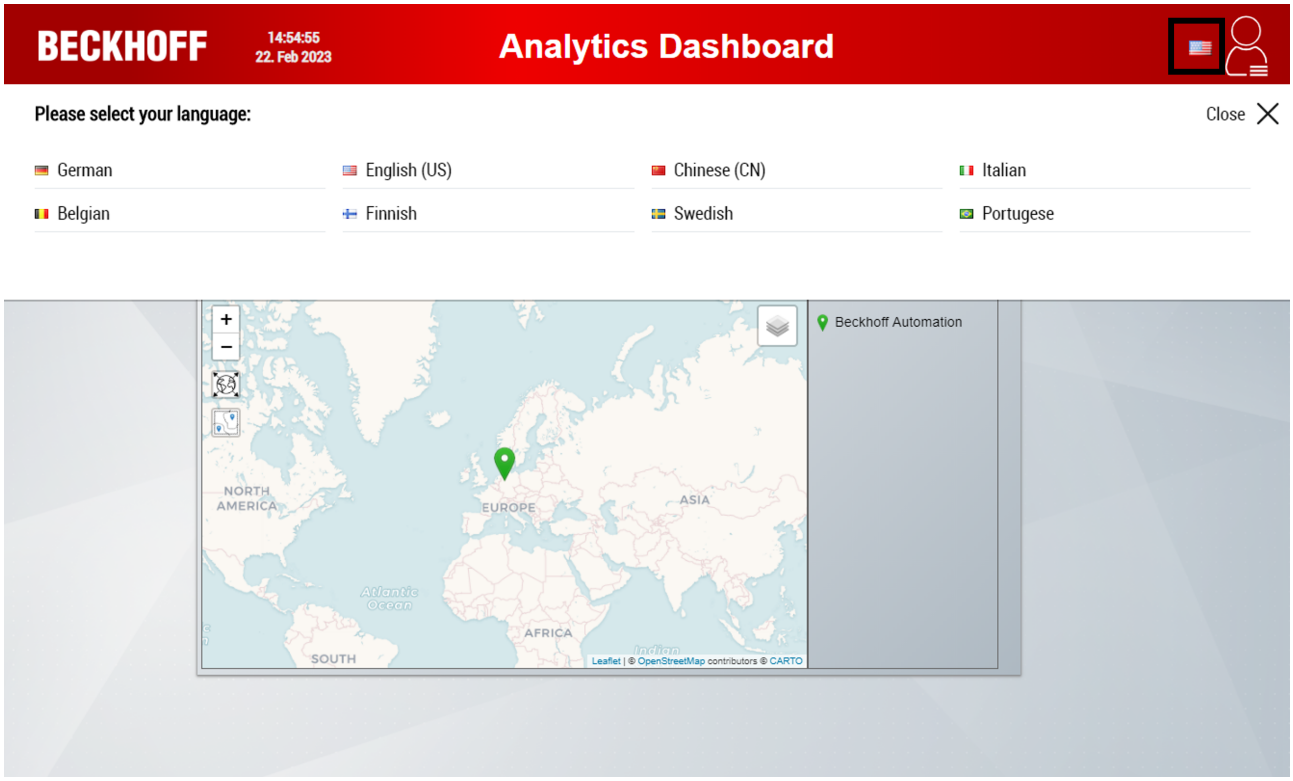
In the generated TwinCAT HMI Engineering project this is easily possible, since the translation entries are already prepared. The texts for the respective languages are stored in the **Localization** files. The names of your networks and modules are automatically entered there and only have to be translated. To do this, open all the files of the languages for which you need a translation. The following screenshot lists the entries that need to be translated for German. The number of entries to be processed varies depending on the complexity of the Analytics project.



Once completed, you can open the dashboard by clicking the **Google Chrome** button (or the name of your default browser). Simply reloading in the browser is not sufficient, because the project has to be rebuilt internally. In the following image the texts which are now also switched with the language switch are highlighted.



The language can be changed via the flag in the upper right corner.



The language can be set and saved user-specifically from Dashboard version 2.0.

6.3 Analytics Reporting

24/7 reporting is also possible in Analytics Runtime. The TwinCAT Analytics Workbench Configurator can be used to configure the Reporting modules provided for this purpose (see 24/7 Reporting). The Reporting Collectors collect the data and send it to the Reporting Server. The Reporting Triggers trigger the creation of a report in the Reporting Server.

PLC function blocks are also generated for the Reporting modules via Runtime deployment [▶ 43]. This PLC code can then be downloaded to a TwinCAT Analytics Runtime.
In order for the Reporting Server to be available on the TwinCAT Analytics Runtime system, the Measurement Setup must be executed on this system. After successful setup, the Reporting Sever is started as a Windows service.
Provided that the TwinCAT Analytics Runtime is on the same system, the parameter for the AmsNetId in the constructor of the PLC function blocks can be left empty. If this is not the case, an Ads connection must be established between the two systems and the AmsNetId of the system with the Reporting Server must be entered in the constructor of the PLC function blocks.

7 Appendix

7.1 FAQ - frequently asked questions and answers

In this section frequently asked questions are answered in order to make your work with TwinCAT Analytics Runtime easier. If you have further questions, please contact our support team support@beckhoff.com.

It is possible to extend the number of connected HMI clients on the Analytics Runtime device? [▶ 167]

?It is possible to extend the number of connected HMI clients on the Analytics Runtime device?

!Yes. Four HMI Clients are already included in the Analytics Runtime. But it is a standard TwinCAT system. So you can book additional licenses like TF2030 Client Pack 10 or TF2040 Client Pack 25.

More Information:
www.beckhoff.com/tf3550

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