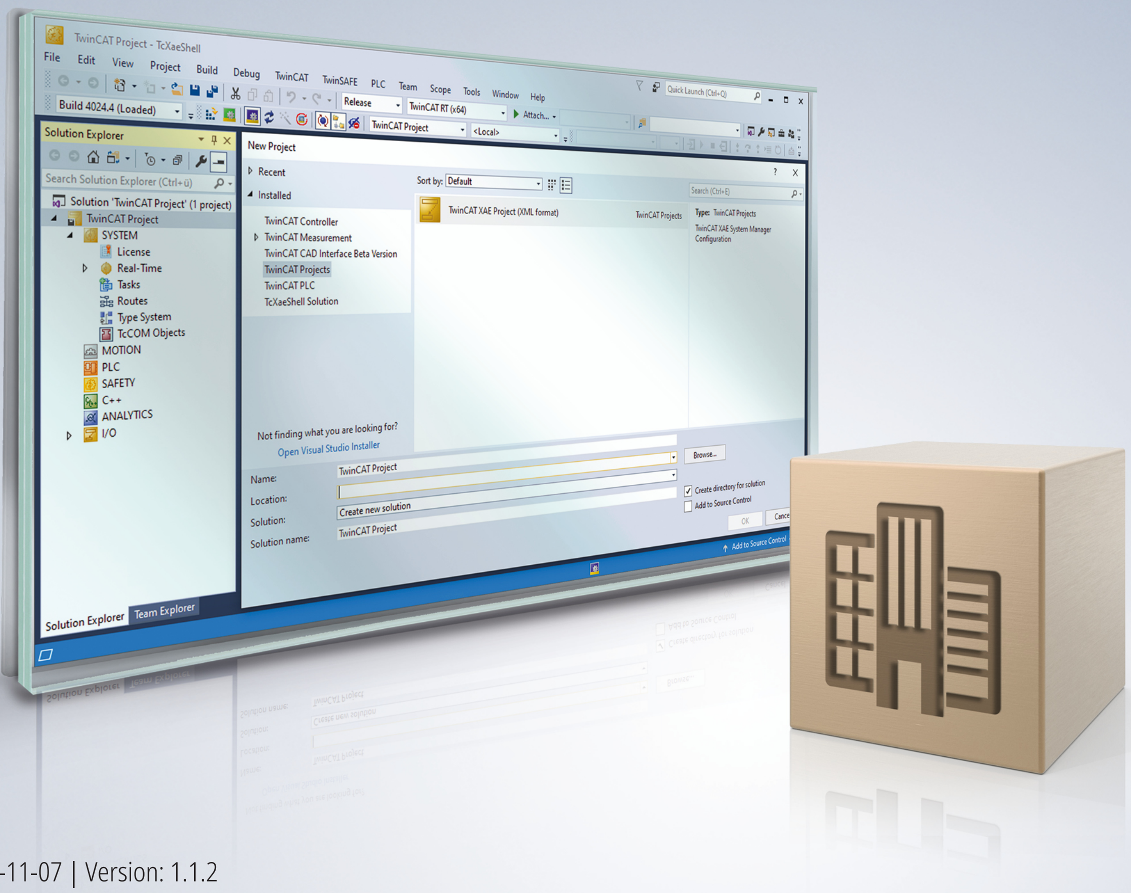
BECKHOFF New Automation Technology

Manual | EN

TF8020

TwinCAT 3 | BACnet



2023-11-07 | Version: 1.1.2

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® 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 Introduction

The newly developed TwinCAT library *Tc3_BACnetRev14* implements a fully object-oriented engineering and configuration process to provide a convenient project planning of Beckhoff BACnet controllers.

Using the library does not require any OOP (Object-Oriented Programming) skills. However, for advanced users, this architecture offers flexible extendable options.

So far, using the older library Tc2_BACnetRev12, the connection between the PLC variables and the BACnet supplement was established using the automapping process. The BACnet function blocks were complemented by comments describing the object properties which were parsed by the map or remap function in the System Manager. This process is still available in TwinCAT 4024 and can be used by adding the library Tc2_BacnetRev12. Please note that this function may no longer be supported in future TwinCAT versions!

Using both libraries Tc3_BACnetRev14 and Tc2_BACnetRev12 within one project is not supported.

BACnet® is a registered trademark of ASHRAE.

2.1 Overview

The components are provided by two libraries:

Tc3_BACnetRev14: This library provides function blocks, data types, global variables and parameters required for the engineering of BACnet objects. This library is provided with the TwinCAT installation (4024.11 or higher) as compiled library (without source-code). The uncompiled source-code is provided upon request (e-mail: buildingautomation@beckhoff.com).

Please note, loading uncompiled libraries takes some time (the library is marked with a clock symbol in this case). Wait until the clock symbols disappears before using the library.

Tc3_BA2_Common**:** This library contains all function blocks, data types, global variables and parameters commonly used by BACnet and TwinCAT 3 Building Automation (Tc3BA). E. g. engineering units and a PID controller are available in this library.

Please add this library in addition to *Tc3_BACnet_Rev14* if you want to use these options.

Some function blocks may require additional libraries:

Tc2_Utilities: Contains general helper functions, e. g. to access the operating system, create csv files or to convert and format data (like UTF-8 character set support).

Tc2_SUPS: Contains functions to use an uninterruptable power supply (UPS).

2.2 System Requirements

Target System: TwinCAT XAR 4024.17 or higher

Engineering-PC: TwinCAT XAE 4024.17 or higher.

3 Overview about BACnet properties

BACnet objects are composed using properties which are either specified in the BACnet standard or may be proprietary. Some properties are required, where others may be optional. A few properties are required to be writable, others may be made writable if needed. The conformance code in the PICS document describes which properties belong to which group:

R = required

O = optional

W = writable

The column "CC" describes the requirements by the BACnet standard.
"WA" describes the write access options in TwinCAT.

The following table shows an example object (analog input, not all properties are shown) from the PICS document (the document can be downloaded here: www.beckhoff.com/bacnet).

Analog Input

Property	Data Type	CC	WA
Object_Identifier	BACnetObjectIdentifier	R	R
Object_Name	CharacterString	R	W
Object_Type	BACnetObjectType	R	R
Present_Value	REAL	R	R
Description	CharacterString	O	W
Device_Type	CharacterString	O	W
Status_Flags	BACnetStatusFlags	R	R
Event_State	BACnetEventState	R	R
Reliability	BACnetReliability	O	R

3.1 Input, Output, Value object types

Analog, Binary and Multistate objects are the most commonly used objects. Analog objects represent a floating point (REAL) information, Binary objects a digital (inactive / active) information and Multistate objects a set of multiple states.

Input object types are typically used to represent a physical hardware input connected to the device, e. g. a temperature or brightness sensor.

Output object types typically represent physical outputs like a 0-10V output or a percentage of a valve actuator.

Value object types are used to represent virtual information like a setpoint or a control parameter.

3.2 Most commonly used BACnet properties

This chapter explains the meaning of the most commonly used BACnet object properties. For a complete list of objects supported by TwinCAT, the supported properties of these objects and the implemented conformance code refer to the PICS document.

3.2.1 Object_Identifier

This property consists of two elements, a 10-bit `Object_Type` and a 22-bit `Object_Instance`. The `Object_Type` is defined in the BACnet standard:

```
AnalogInput :=0
AnalogOutput :=1
AnalogValue :=2
```

```
BinaryInput :=3
BinaryOutput :=4
BinaryValue :=5
Calendar :=6
Command :=7
Device :=8
EventEnrollment :=9
File :=10
Group :=11
Loop :=12
MultiStateInput :=13
MultiStateOutput :=14
NotificationClass :=15
Program :=16
Schedule :=17
Averaging :=18
MultiStateValue :=19
TrendLog :=20
LifeSafetyPoint :=21
LifeSafetyZone :=22
Accumulator :=23
PulseConverter :=24
EventLog := 25
GlobalGroup := 26
TrendLogMultiple := 27
LoadControl := 28
StructuredView := 29
AccessDoor := 30
unassigend31 := 31
AccessCredential := 32
AccessPoint := 33
AccessRights := 34
AccessUser := 35
AccessZone := 36
CredentialDataInput := 37
NetworkSecurity := 38
BitStringValue := 39
CharacterStringValue := 40
DatePatternValue := 41
DateValue := 42
DateTimePatternValue := 43
DateTimeValue := 44
IntegerValue := 45
LargeAnalogValue := 46
OctetStringValue := 47
PositiveIntegerValue := 48
TimePatternValue := 49
TimeValue := 50
NetworkPort := 51
AlertEnrollment := 52
Channel := 53
```

The `Object_Instance` is specified when the object is created and must be unique per object type. In the TwinCAT 3 library **Tc3_BACnetRev14** the instance numbers start at 10000 by default. The highest 22-bit value = 4194303 cannot be assigned to object instances. This value is reserved by the BACnet standard as a wildcard in case the actual object instance number is not known.

In the case of the `Device_Objects`, the `Object_Instance` must be unique within the entire BACnet internetwork (across all BACnet data link layers).

3.2.2 Object_Name

This property contains the name of the object. In many cases this property contains the Data Point Addressing Description (DPAD), i.e. it is a project or customer-specific naming convention for data points.

The `Object_Name` must contain at least one printable character that must be unique within the device. In the case of the `Object_Name` of the device object, this property must be unique within the entire BACnet internetwork (across all BACnet data link layers). The default character set in TwinCAT is UTF-8. The maximum length of this string in TwinCAT is 255.

3.2.3 Object_Type

This property represents the 10 bit type information of the BACnet Object Identifier. If this property is requested, the response only contains the object type. If the property ObjectIdentifier is requested, the 32 bit Identifier (10 bit type and 22 bit instance number) is returned in the response.

3.2.4 Present_Value

This property is very likely the most important property. It represents the current process value of the object.

The data-type of this property is:

Analog objects:	`REAL` (32-bit single accuracy floating point number according to ANSI-IEEE 754)
Binary objects:	`ENUMERATED` `0`=inactive, `1`=active *
Multistate objects:	`UNSIGNED INT`

* Please note that the library uses a BOOL data type and provides automatic conversion to the BACnet enumeration.

3.2.5 Description

This optional property may contain an additional text information describing the object. The default character set is UTF-8. The maximum length in TwinCAT is 255 characters.

3.2.6 Device_Type

(Available in input and output object types)

This optional text property may be used to specify the hardware type, e. g. a PT1000 sensor. It may contain other information like the order number or similar information about the hardware connected. The default character set is UTF-8. The maximum length in TwinCAT is 255 characters.

3.2.7 Status_Flags

This property represents four boolean status flags:

`IN_ALARM`: in case of an active alarm, this flag is set to `TRUE`.

`FAULT`: if the Reliability property is not equal to `NO_FAULT_DETECTED`, this flag is set to `TRUE`. Error states include defect sensors, links, etc.

`OVERRIDDEN`: If the `Present_Value` property of the object is overridden locally by the application, this flag is set to `TRUE`. The values of the properties `Present_Value` and `Reliability` are no longer taken from the hardware. The definition of the overwritten state is done locally by the application.

`OUT_OF_SERVICE`: this value indicates the state of the property `Out_of_Service`.

3.2.8 Event_State

This property represents the event state in case of an active alarm, e.g. `LOW_LIMIT`. In the normal state or in the absence of an alarm associated with the object, the value must be `NORMAL`.

3.2.9 Reliability

This property indicates whether the value of the property `Present_Value` is reliable (`NO_FAULT_DETECTED`) and if not, *why* is not (e.g. short circuit, missing sensor, etc.).

3.2.10 Out_of_Service

This Boolean property indicates whether the object is in service or not. In the case of `Out_of_Service = TRUE`, `Present_Value` is no longer tracked by hardware and `Present_Value` becomes writable, even for input object types. This allows to set a value in case of defective hardware.

3.2.11 Update_Interval

This optional property specifies the maximum time in hundredths of a second between updates of the `Present_Value` if the input is not out of service and not overwritten.

3.2.12 Units

This property specifies the technical unit of the object. As these units are used for BACnet and for general building automation purposes, the listed values can be retrieved from the library Tc3_BA2_Common. The list can be found in the section DUTs/Enumerations/Units/E_BA_Unit.

3.2.13 Min_Pres_Value

This property specifies the lowest value in technical units that can be reliably determined for the `Present_Value` property of the object. Currently not available in Analog Value objects.

3.2.14 Max_Pres_Value

This property specifies the highest value in technical units that can be reliably determined for the `Present_Value` property of the object. Currently not available in Analog Value objects.

3.2.15 Resolution

This property specifies the lowest detectable change in the `Present_Value` property in technical units.

3.2.16 COV_Increment

This property specifies the hysteresis for Change-of-Value notifications. If the new value of the `Present_Value` property exceeds the old value by `COV_Increment`, the recipients subscribed to this object are informed about the value change.

Please note: a `COV_Increment` of zero means that every (even small) change is reported, which can lead to a message flood in the network!

3.2.17 Time_Delay

This property specifies the minimum amount of time in seconds that `Present_Value` must be outside the range defined by the `High_Limit` and `Low_Limit` properties before a `TO-OFFNORMAL` event is generated, or within the same range, including the Deadband property, before a `TO-NORMAL` event is generated.

3.2.18 Notification_Class

This property specifies the instance number of the Notification Class object for the event-notification-distribution.

3.2.19 High_Limit

This property specifies the limit value that `Present_Value` must exceed before an event is generated.

3.2.20 Low_Limit

This property specifies the limit value below which `Present_Value` must fall before an event is generated.

3.2.21 Deadband

This property specifies a range between the `High_Limit` and `Low_Limit` properties in which `Present_Value` must lie for a `TO-NORMAL` event to be generated.

The following conditions apply to the event generation:

- `Present_Value` must be below the `High_Limit` minus deadband and
- `Present_Value` must exceed the `Low_Limit` plus deadband and
- `Present_Value` must remain within this range for a minimum period specified in the `Time_Delay` property, and
- either the `HighLimitEnable` or the `LowLimitEnable` flag must be set in the `Limit_Enable` property, and
- the TO-NORMAL flag must be set in the `Event_Enable` property.

3.2.22 Limit_Enable

This property specifies two flags that enable and disable compliance with the upper alarm limit and lower alarm limit separately.

Please note:
up to BACnet revision 12 there was an error in the description of the Property Description in the BACnet standard. Only event reporting was enabled / disabled up to version 12. As of revision 13, this specification was changed in the BACnet standard. Now, observation of these properties is enabled / disabled.

3.2.23 Event_Enable

This property specifies three flags that separately enable and disable the reporting of `TO-OFFNORMAL`, TO-FAULT and TO-NORMAL events.

3.2.24 Acked_Transitions

This property specifies three flags that indicate the receipt of acknowledgements for the `TO-OFFNORMAL`, `TO-FAULT` and `TO-NORMAL` events separately.

3.2.25 Notify_Type

This property specifies whether the notifications generated by the object should be Events or Alarms.

3.2.26 Event_Time_Stamps

This property specifies the times of the last event notifications for the `TO-OFFNORMAL`, `TO-FAULT` and `TO-NORMAL` events, respectively.

Please note: the property value can be a choice of `DateTime` (mostly used), `Time` (rarely used) or `Sequence Number` (if the device does not support a clock).

3.2.27 Event_Message_Texts

This property specifies the message text of the last event notifications for the `TO-OFFNORMAL`, `TO-FAULT` and `TO-NORMAL` events, respectively.

3.2.28 Profile_Name

This property is the name of an object profile to which this object conforms. To ensure uniqueness, a profile name must begin with a vendor identifier code (415 for Beckhoff Automation) in base-10 integer format, followed by a dash.

3.2.29 Event_Message_Texts_Config

This property contains the strings that form the basis of the "Message Text" parameter for the event notifications of the `TO_OFFNORMAL`, `TO_FAULT` and `TO_NORMAL` events.

3.2.30 Event_Detection_Enable

This property indicates whether or not intrinsic reporting is enabled in the object and controls whether or not the object will be considered by event summarization services.

3.2.31 Event_Algorithm_Inhibit_Ref

This reference specifies the property that controls the value of the `Event_Algorithm_Inhibit` property.

3.2.32 Event_Algorithm_Inhibit

This property indicates whether or not the event algorithm has been disabled for the object.

3.2.33 TimeDelay_Normal

This property specifies the minimum time in seconds that `Present_Value` must remain within the range defined by the properties `High_Limit` and `Low_Limit`, including the Deadband property, before a `TO-NORMAL` event is generated.

Please note: up to BACnet revision 12, only Time_Delay was available in the BACnet standard for the two transitions `TO_OFFNORMAL` and `TO_NORMAL`. From revision 13, this optional TimeDelay_Normal property is used to specify a separate hysteresis for `TO_NORMAL` events.

3.2.34 Reliability_Evaluation_Inhibit

This property indicates whether or not reliability-evaluation is disabled in the object.

3.2.35 Property_List

This property is a BACnetARRAY of property identifiers. A property identifier for each property that exists within the object.

3.2.36 Priority_Array

This property is a read-only array of prioritized values. See chapter Command prioritization [▶ 62].

3.2.37 Relinquish_Default

This property is the default value used for the Present_Value property when all command priority values in the `Priority_Array` property have the value NULL. See chapter Command prioritization [▶ 62].

3.2.38 Inactive_Text

(Binary objects) This property represents a human-readable description of the `INAKTIV` state. For example, if the binary object represents a switch, the inactive state can be represented as `AUS`.

3.2.39 Active_Text

(Binary objects) This property represents a human-readable description of the `AKTIV` state. For example, if the Binary object represents a switch, the active state may be represented as ON.

3.2.40 Change_Of_State_Time

(Binary objects) This property represents the date and time at which the most recent change of state occurred.

3.2.41 Change_Of_State_Count

(Binary objects) This property indicates how often the state of property `Present_Value` has changed since property `Change_Of_State_Count` was last set to a null value.

3.2.42 Time_Of_State_Count_Reset

(Binary objects) This property represents the date and time when the `Change_Of_State_Count` property was last set to a null value.

3.2.43 Elapsed_Active_Time

(Binary objects) This property represents the cumulative number of seconds `Present_Value` has had the value `ACTIVE` since the `Elapsed_Active_Time` property was last set to a null value.

3.2.44 Time_Of_Active_Time_Reset

(Binary objects) This property represents the date and time when the `Elapsed_Active_Time` property was last set to a null value.

3.2.45 Alarm_Value

(Binary objects) This property indicates the value `Present_Value` must have before an event is generated.

3.2.46 Minimum_Off_Time

(Binary objects) This property specifies the minimum number of seconds that `Present_Value` remains in the `INACTIVE` state after a write operation in `Present_Value` has set this property to the `INACTIVE` state. See chapter Command prioritization [▶ 62].

3.2.47 Minimum_On_Time

(Binary objects) This property specifies the minimum number of seconds that `Present_Value` remains in the `ACTIVE` state after a write operation in `Present_Value` has set this property to the `ACTIVE` state. See chapter Command prioritization [▶ 62].

3.2.48 Feedback_Value

This property indicates the actual value of the unit controlled by `Present_Value`.

3.2.49 Number_Of_States

(Multistate objects) This property defines the number of states `Present_Value` can have.

3.2.50 State_Text

(Multistate objects) This property is a BACnetARRAY of strings representing descriptions of all possible states of `Present_Value`. The number of descriptions corresponds to the number of states defined in the `Number_Of_States` property.

3.2.51 Alarm_Values

(Multistate objects) This property specifies all states to which `Present_Value` must correspond before a `TO-OFFNORMAL` event is generated.

3.2.52 Fault_Values

(Multistate objects) This property specifies all states to which `Present_Value` must correspond before a `TO-FAULT-` event is generated.

3.2.53 Date_List

(Calendar object) This property is a BACnetLIST of BACnetCalendarEntry, each of which is either a specific date or date pattern (Date), range of dates (BACnetDateRange), or month/week-of-month/day-of-week specification (BACnetWeekNDay).

3.2.54 Weekly_Schedule

(Schedule object) This property is a BACnetARRAY containing exactly seven elements. Each of the elements 1-7 contains a BACnetDailySchedule. A BACnetDailySchedule consists of a list of BACnetTimeValues which describe the sequence of schedule actions on one day of the week when no Exception_Schedule is in effect. Monday equals the first element (day 1) where Sunday equals the last element (day 7).

3.2.55 Exception_Schedule

(Schedule object) This property is a BACnetARRAY of BACnetSpecialEvents. Each BACnetSpecialEvent describes a sequence of schedule actions that takes precedence over the normal day's behavior on a specific day or days. The special event may contain a Date, Date-range, WeekNDay information or a Calendar reference (within the same BACnet device) to specify the exception period.

3.3 Understanding the PICS document

A PICS (**P**rotocol **I**mplementation **C**onformance **S**tatement) is the "BACnet data sheet". This document describes the scope of the BACnet device implementation.

The PICS document for Beckhoff controllers can be downloaded here: https://download.beckhoff.com/download/Document/certificates/beckhoff_bacnet_ip_pics_en_rev14_ver4.0.pdf

The PICS describes the supported object types and for each object type, which properties are supported, the read or write access (so-called conformance code) and possible property limitations.

In addition, the supported Data Link Layer, character sets, segmentation support, etc. is described as well.

The third element of the PICS document is the list of supported BIBBs (**B**ACnet **I**nteroperability **B**uilding **B**lock**s**). The standard describes A-side and B-side BIBBs.
A-side BIBBs are those issued by a client requesting something from or to a B-side device (server).

To use a specific functionality, such as Change-of-Value, both sides, client and server, must support the corresponding BIBB.

Example:

DS-COV-A: Client requests data using the subscribed Change-of-Value.

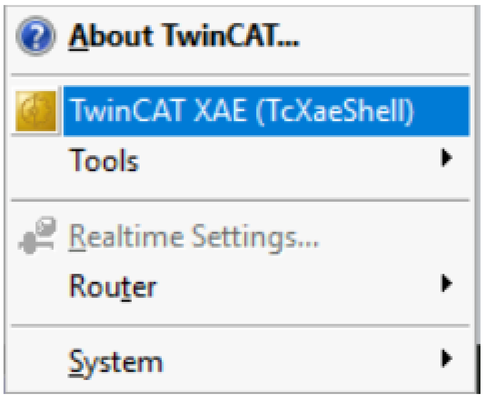
DS-COV-B: The server responds to COV subscriptions from the client and provides the data to the client when the value has changed by sending a COV notification.

4 Quickstart

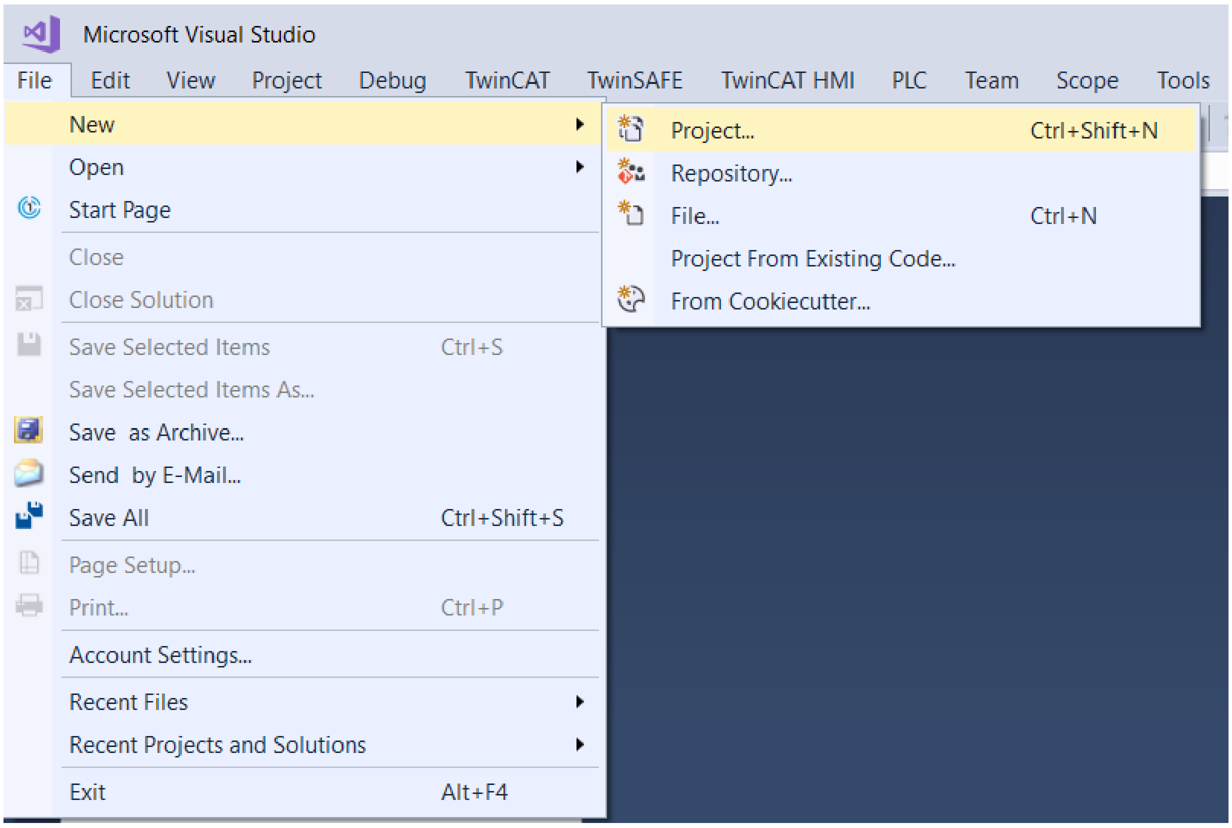
This example shows how to create a BACnet server containing a simple analog value object on a CX controller.

4.1 Creating the TwinCAT project

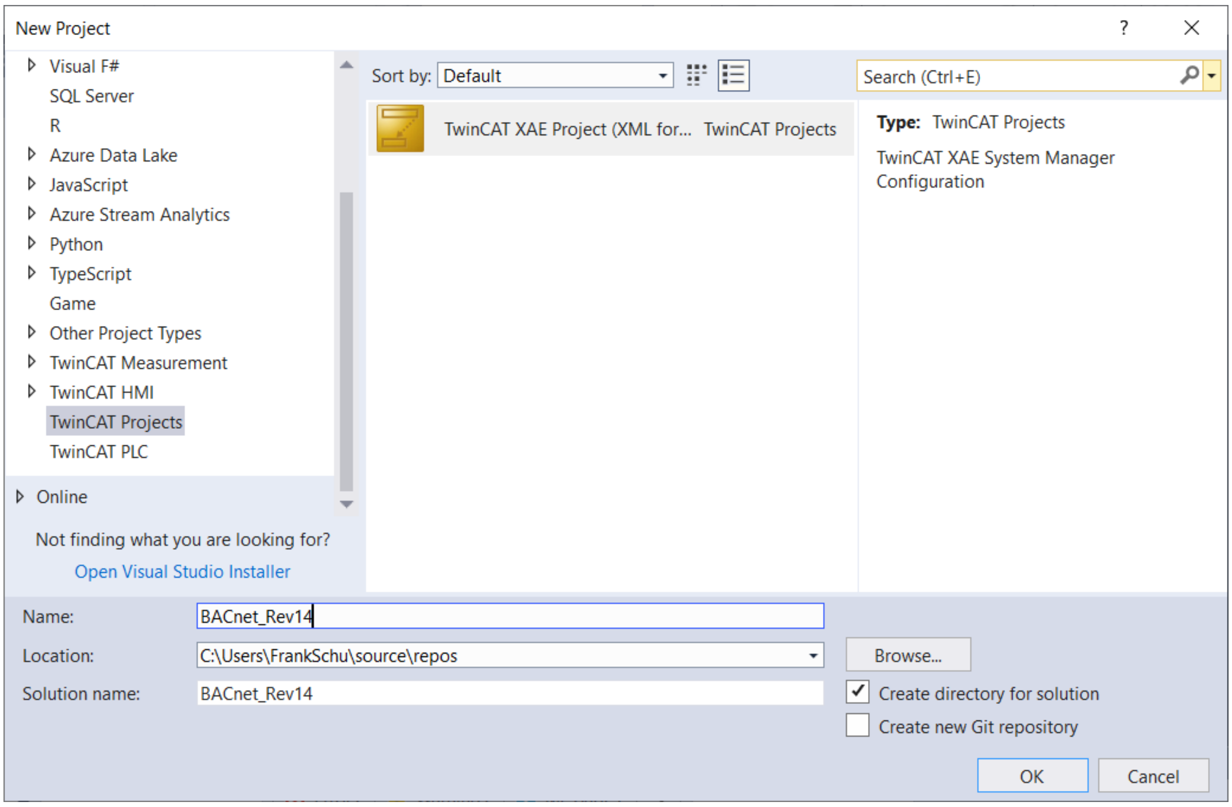
1. Start TwinCAT.



2. Create a new, empty project.



3. Select *TwinCAT Projects* in the list of project types, select *TwinCAT XAE Project*, name the project and change the directory if necessary.



Type: TwinCAT Projects

TwinCAT XAE System Manager Configuration

Name: BACnet_Rev14

Location: C:\Users\FrankSchu\source\repos

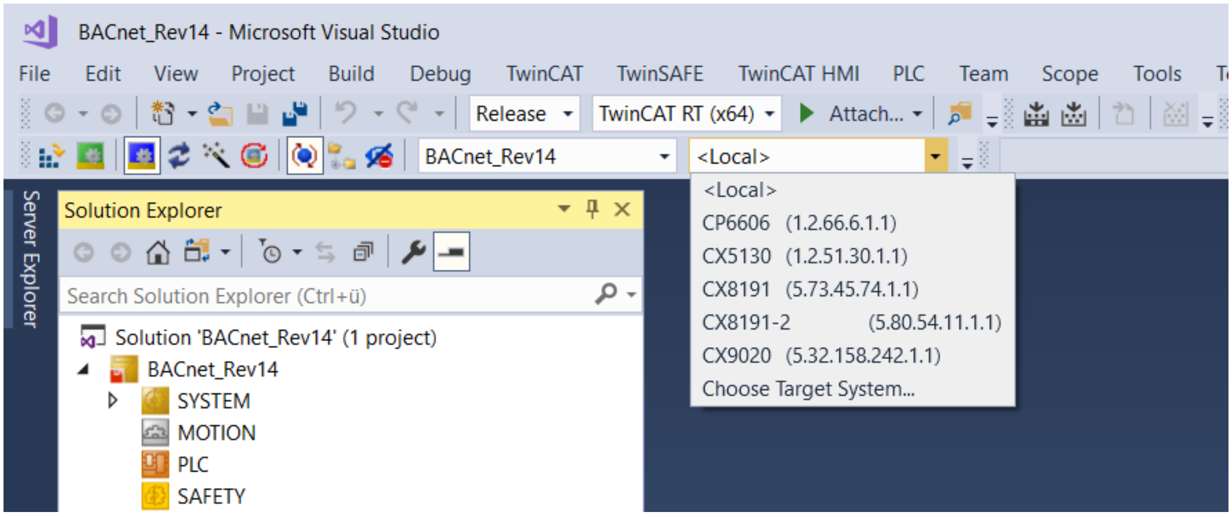
Solution name: BACnet_Rev14

[✓] Create directory for solution

[ ] Create new Git repository

4.1.1 Selecting the target system

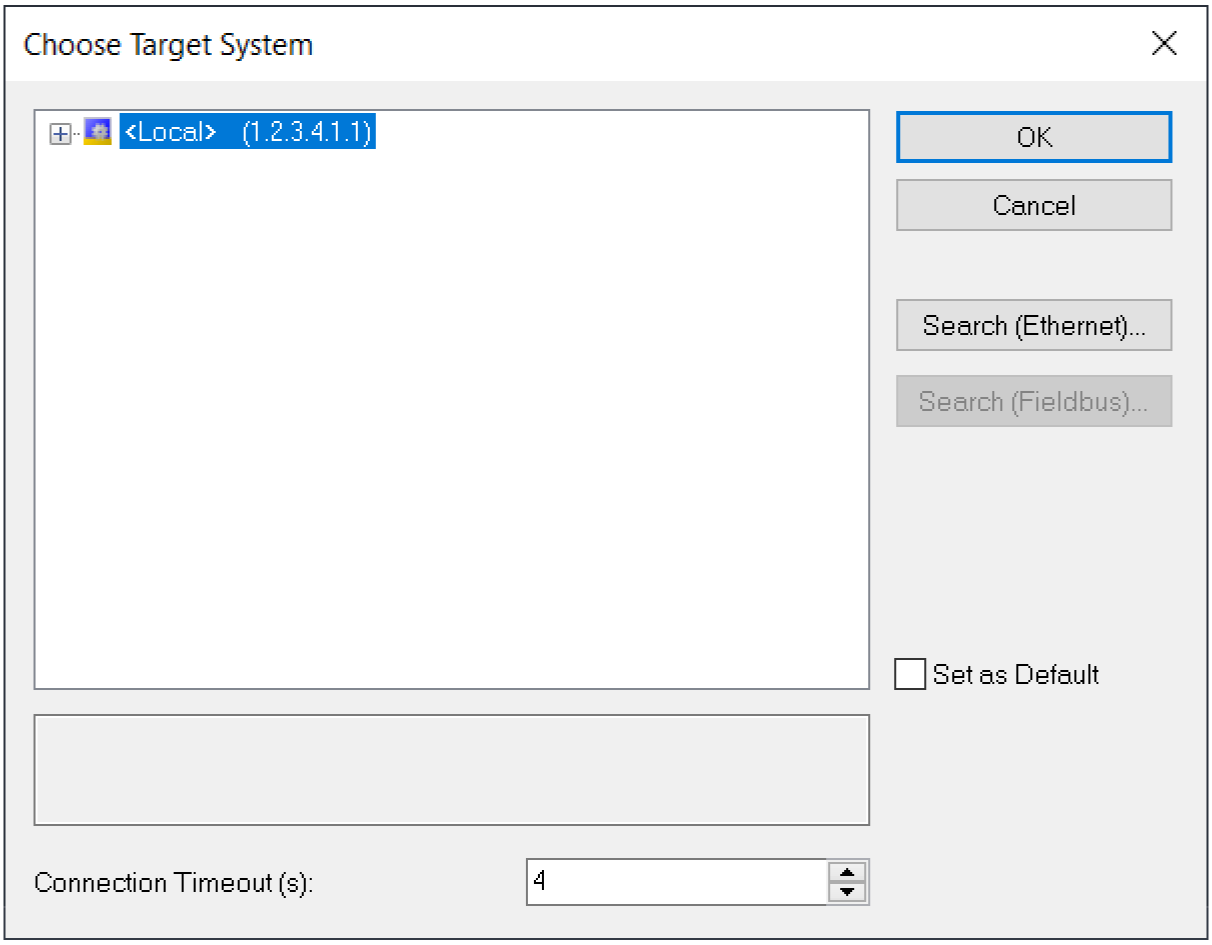
1. Select the target system from the list of controllers already connected via an ADS route.



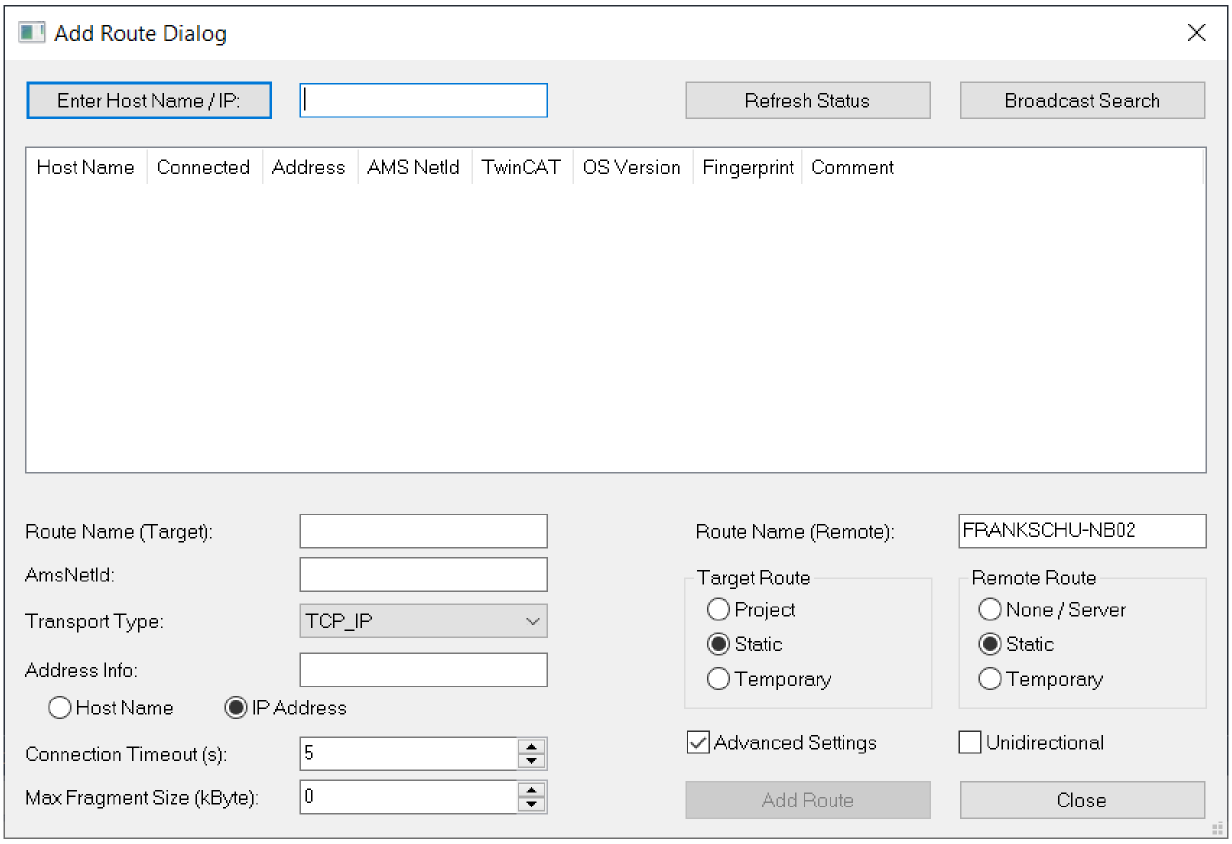
<Local>
CP6606 (1.2.66.6.1.1)
CX5130 (1.2.51.30.1.1)
CX8191 (5.73.45.74.1.1)
CX8191-2 (5.80.54.11.1.1)
CX9020 (5.32.158.242.1.1)
Choose Target System...

2. If no ADS route to the controller exists, select the function *Choose Target System*.

3. Select *Search (Ethernet)* in this dialog.



4. Select *Broadcast Search*.



Route Name (Target):

AmsNetId:

Transport Type: TCP_IP

Address Info:

Host Name [✓ IP Address]

Connection Timeout (s): 5

Max Fragment Size (kByte): 0

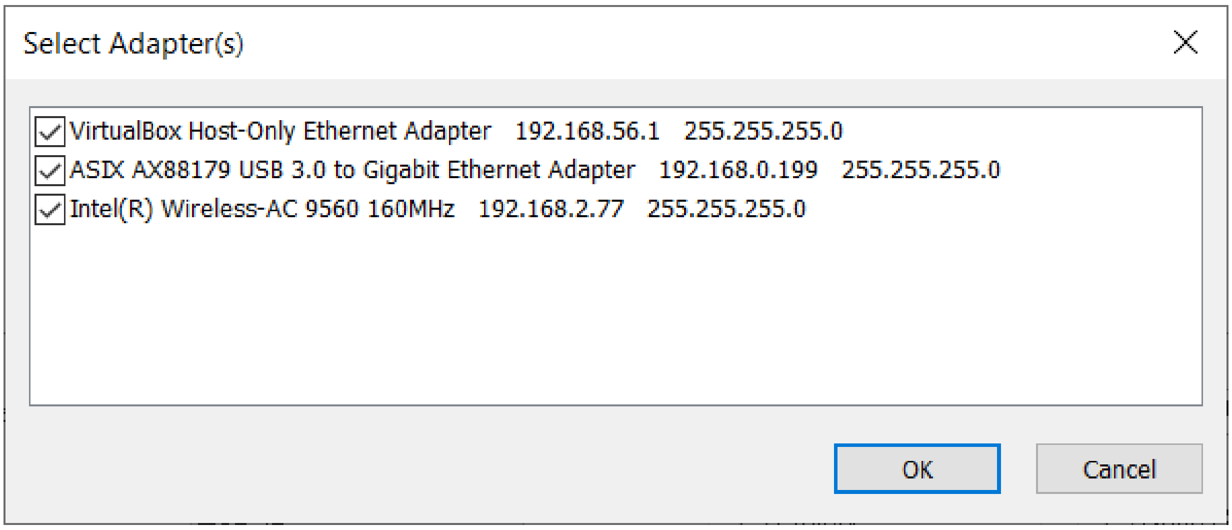
Route Name (Remote): FRANKSCHU-NB02

Target Route: Project [✓ Static] Temporary

Remote Route: None / Server [✓ Static] Temporary

[✓] Advanced Settings

[ ] Unidirectional



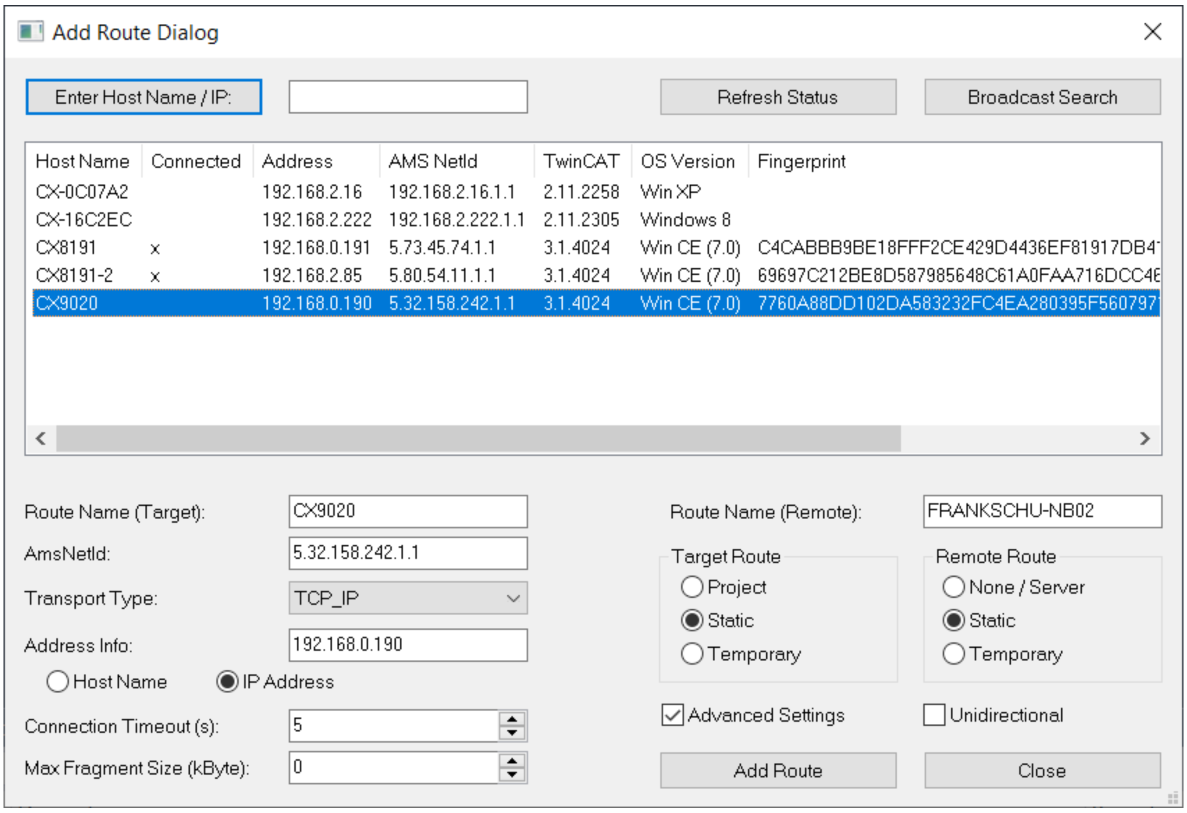
✓ VirtualBox Host-Only Ethernet Adapter 192.168.56.1 255.255.255.0

✓ ASIX AX88179 USB 3.0 to Gigabit Ethernet Adapter 192.168.0.199 255.255.255.0

✓ Intel(R) Wireless-AC 9560 160MHz 192.168.2.77 255.255.255.0

5. Select the network adapters which are used to search for controllers.

⇨ All available controllers in the network are shown in this list. The letter "x" in the *Connected* column means that an ADS route has already been set up.



Add Route Dialog

Enter Host Name / IP: | Refresh Status | Broadcast Search

Host Name	Connected	Address	AMS NetId	TwinCAT	OS Version	Fingerprint
CX-0C07A2		192.168.2.16	192.168.2.16.1.1	2.11.2258	Win XP	
CX-16C2EC		192.168.2.222	192.168.2.222.1.1	2.11.2305	Windows 8	
CX8191	x	192.168.0.191	5.73.45.74.1.1	3.1.4024	Win CE (7.0)	C4CABBB9BE18FFF2CE429D4436EF81917DB4
CX8191-2	x	192.168.2.85	5.80.54.11.1.1	3.1.4024	Win CE (7.0)	69697C212BE8D587985648C61A0FAA716DCC4
CX9020		192.168.0.190	5.32.158.242.1.1	3.1.4024	Win CE (7.0)	7760A88DD102DA583232FC4EA280395F560797

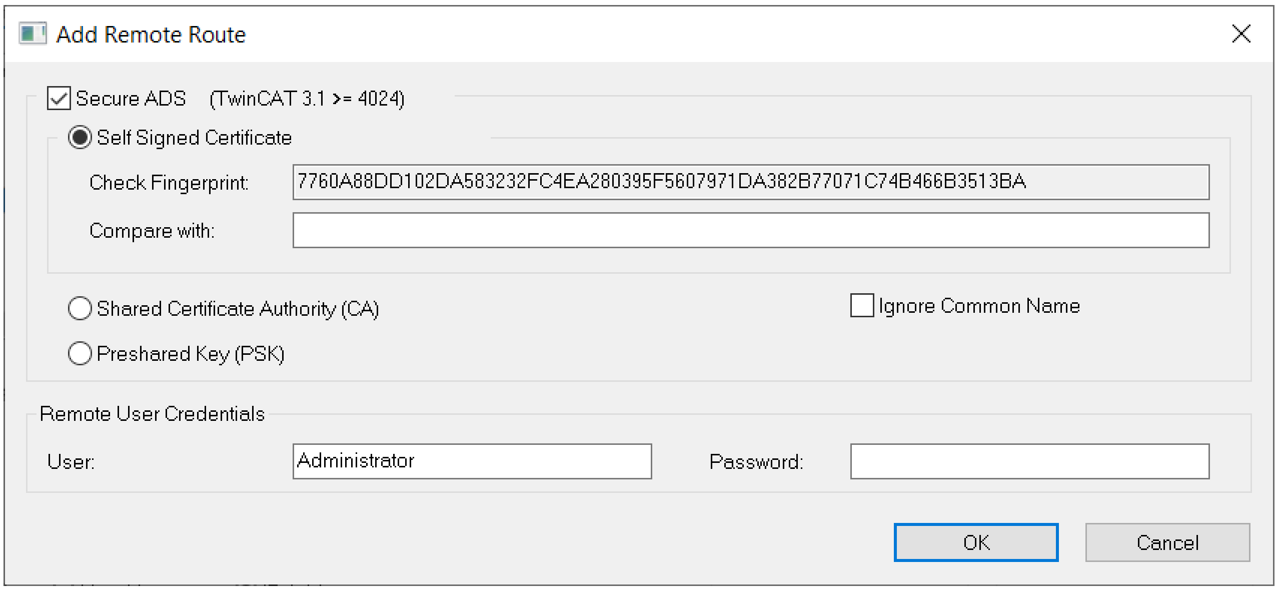
Route Name (Target): CX9020
AmsNetId: 5.32.158.242.1.1
Transport Type: TCP_IP
Address Info: 192.168.0.190
Host Name [✓ IP Address]
Connection Timeout (s): 5
Max Fragment Size (kByte): 0

Route Name (Remote): FRANKSCHU-NB02
Target Route: Project [✓ Static] Temporary
Remote Route: None / Server [✓ Static] Temporary
[✓] Advanced Settings
[ ] Unidirectional

Add Route | Close

6. Select the controller to be used for the BACnet project, select *IP Address* in the section *Address Info* and select *Add Route*.

It is optionally possible to create a secure ADS route. Please refer to the Beckhoff Information System for further information how to use Secure ADS.



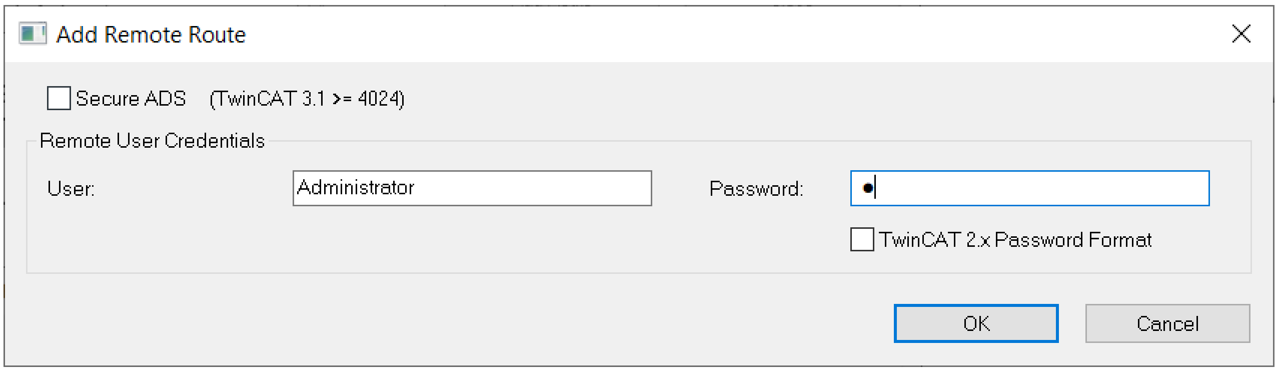
Add Remote Route

[✓] Secure ADS (TwinCAT 3.1 >= 4024)
[✓ Self Signed Certificate]
Check Fingerprint: 7760A88DD102DA583232FC4EA280395F5607971DA382B77071C74B466B3513BA
Compare with:
Shared Certificate Authority (CA)
[ ] Ignore Common Name
Preshared Key (PSK)

Remote User Credentials
User: Administrator
Password:

OK | Cancel

This example uses a regular ADS route. The standard password for the user *Administrator* is the number *1*. We strongly recommend changing the default to a more secure password and document it in the revision documents of the project, at the latest when finishing the project.



Secure ADS (TwinCAT 3.1 >= 4024) [ ]

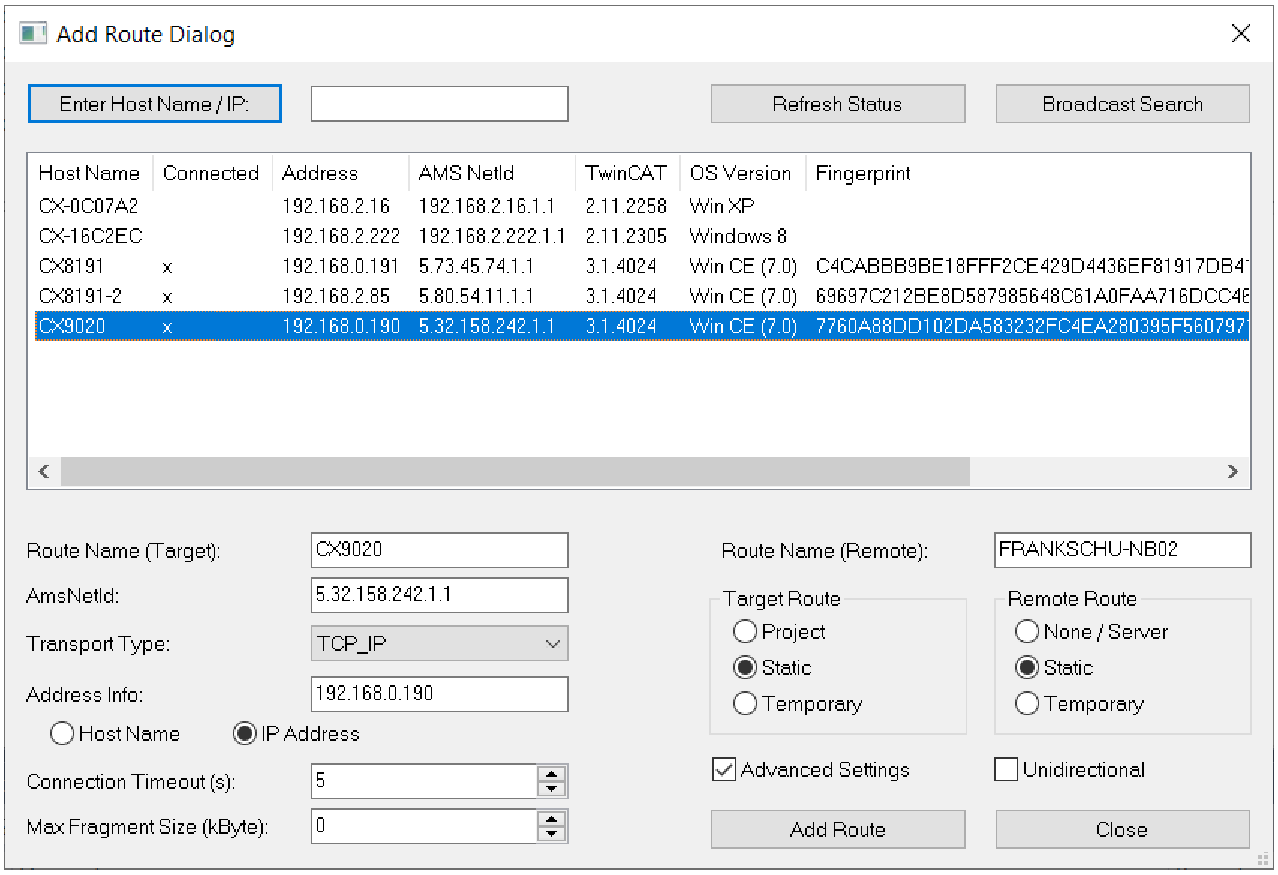
Remote User Credentials

User: Administrator

Password: •

TwinCAT 2.x Password Format [ ]

The controller is now connected through ADS.



Host Name	Connected	Address	AMS NetId	TwinCAT	OS Version	Fingerprint
CX-0C07A2		192.168.2.16	192.168.2.16.1.1	2.11.2258	Win XP	
CX-16C2EC		192.168.2.222	192.168.2.222.1.1	2.11.2305	Windows 8	
CX8191	x	192.168.0.191	5.73.45.74.1.1	3.1.4024	Win CE (7.0)	C4CABBB9BE18FFF2CE429D4436EF81917DB4
CX8191-2	x	192.168.2.85	5.80.54.11.1.1	3.1.4024	Win CE (7.0)	69697C212BE8D587985648C61A0FAA716DCC4
CX9020	x	192.168.0.190	5.32.158.242.1.1	3.1.4024	Win CE (7.0)	7760A88DD102DA583232FC4EA280395F56079

Route Name (Target): CX9020

AmsNetId: 5.32.158.242.1.1

Transport Type: TCP_IP

Address Info: 192.168.0.190

Host Name [✓ IP Address]

Connection Timeout (s): 5

Max Fragment Size (kByte): 0

Route Name (Remote): FRANKSCHU-NB02

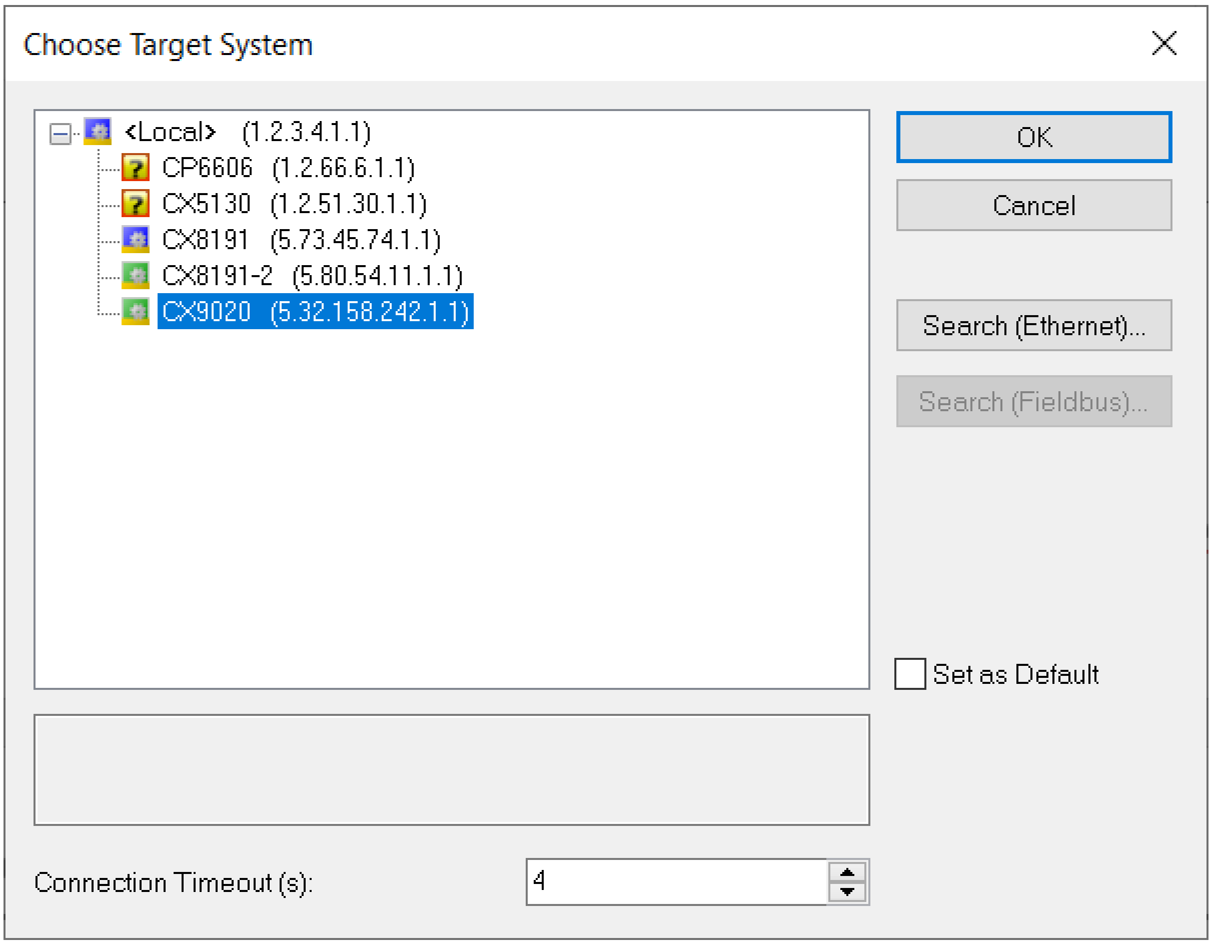
Target Route: Project [✓ Static] Temporary

Remote Route: None / Server [✓ Static] Temporary

Advanced Settings [✓]

Unidirectional [ ]

7. Close the dialog and select the controller from the list of available target systems.



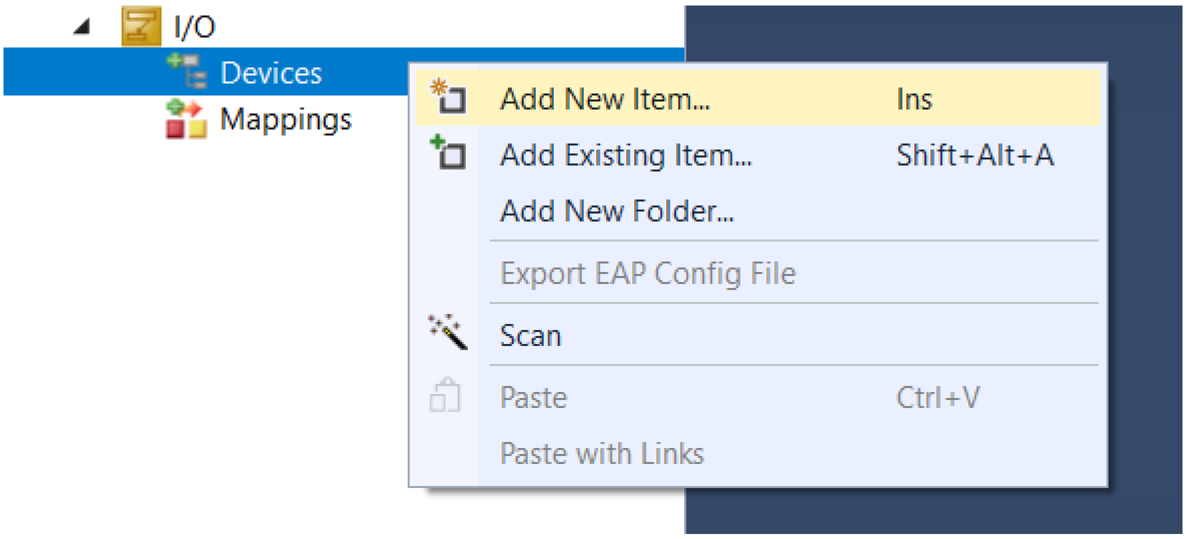
Choose Target System

<Local> (1.2.3.4.1.1)
CP6606 (1.2.66.6.1.1)
CX5130 (1.2.51.30.1.1)
CX8191 (5.73.45.74.1.1)
CX8191-2 (5.80.54.11.1.1)
CX9020 (5.32.158.242.1.1)

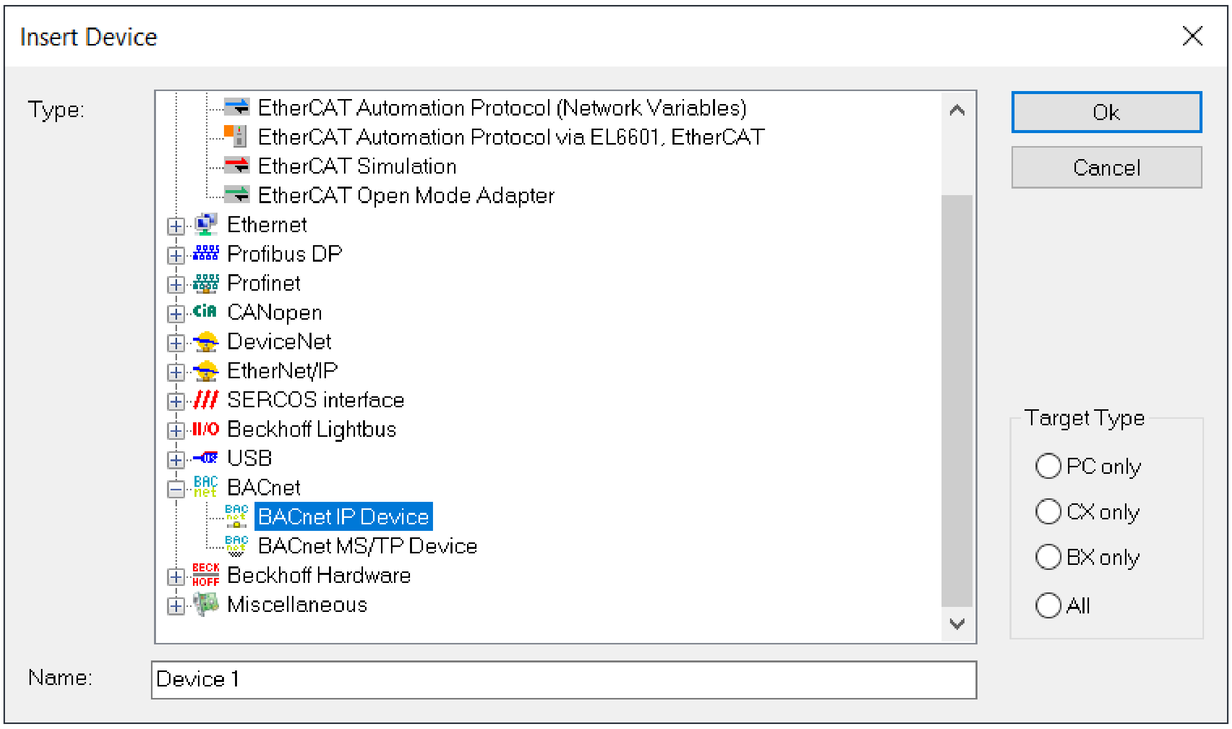
Connection Timeout (s): 4

4.1.2 Create a BACnet Adapter and Server

1. Right click to *Devices* and select Add New Item.

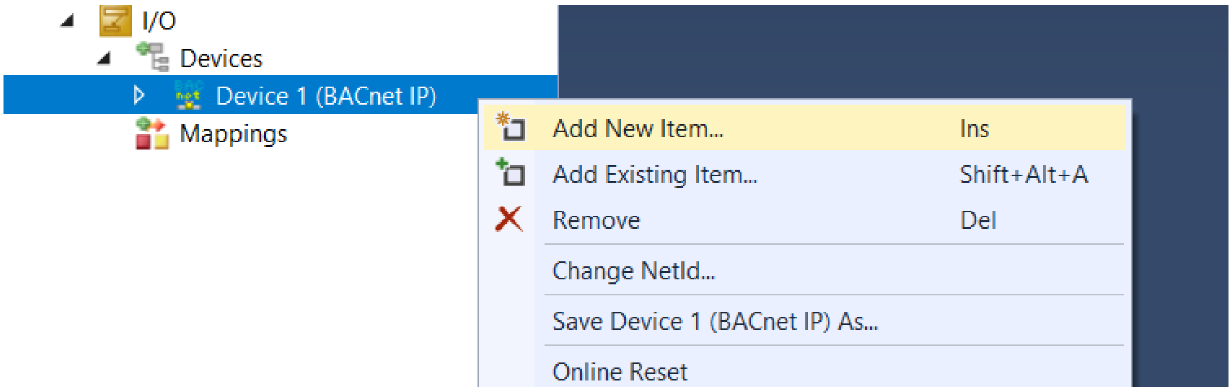


2. Select *BACnet IP Device* from the list of available supplements.

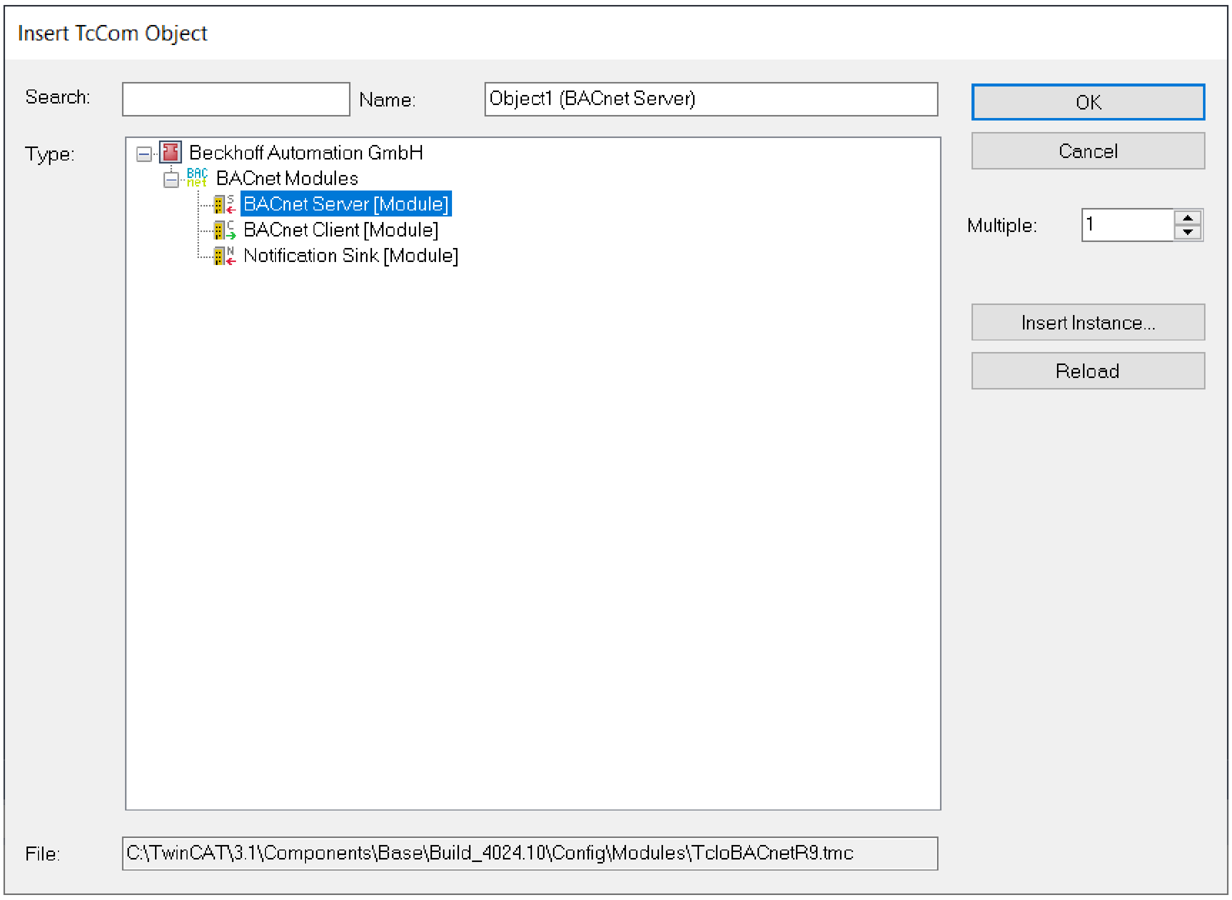


Name: Device 1

3. Right click to menu item *Device 1 (BACnet IP)* and select *Add New Item*.



4. From the list of available TCOM objects select *BACnet Server (Module)*.


Insert TcCom Object

Search:
Name: Object1 (BACnet Server)
Type:
- Beckhoff Automation GmbH
  - BACnet Modules
    - BACnet Server [Module]
    - BACnet Client [Module]
    - Notification Sink [Module]

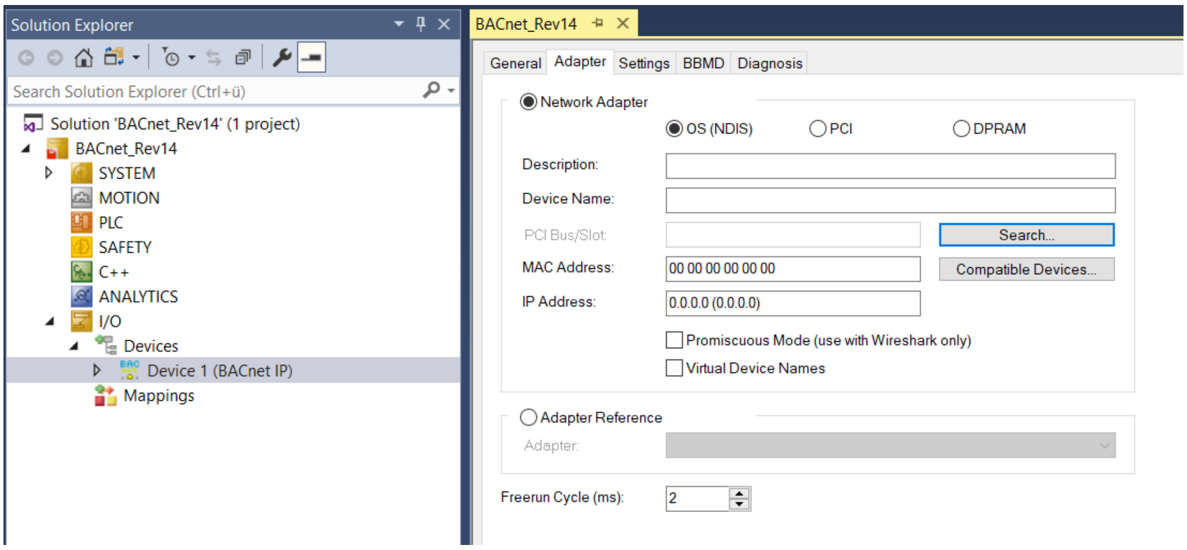
Multiple: 1

File: C:\TwinCAT\3.1\Components\Base\Build_4024.10\Config\Modules\TcIoBACnetR9.tmc

4.1.3 IP address settings

1. Double click to BACnet Adapter *Device 1 (BACnet IP)*, select the *Adapter* tab and check, if the IP address was correctly set.

   ⇨ In the figure below the MAC address and IP address are set to zero, no adapter was selected.


Solution Explorer

Search Solution Explorer (Ctrl+ü)

- Solution 'BACnet_Rev14' (1 project)
  - BACnet_Rev14
    - SYSTEM
    - MOTION
    - PLC
    - SAFETY
    - C++
    - ANALYTICS
    - I/O
      - Devices
        - Device 1 (BACnet IP)
      - Mappings

BACnet_Rev14

General | Adapter | Settings | BBMD | Diagnosis

(●) Network Adapter

(●) OS (NDIS) ( ) PCI ( ) DPRAM

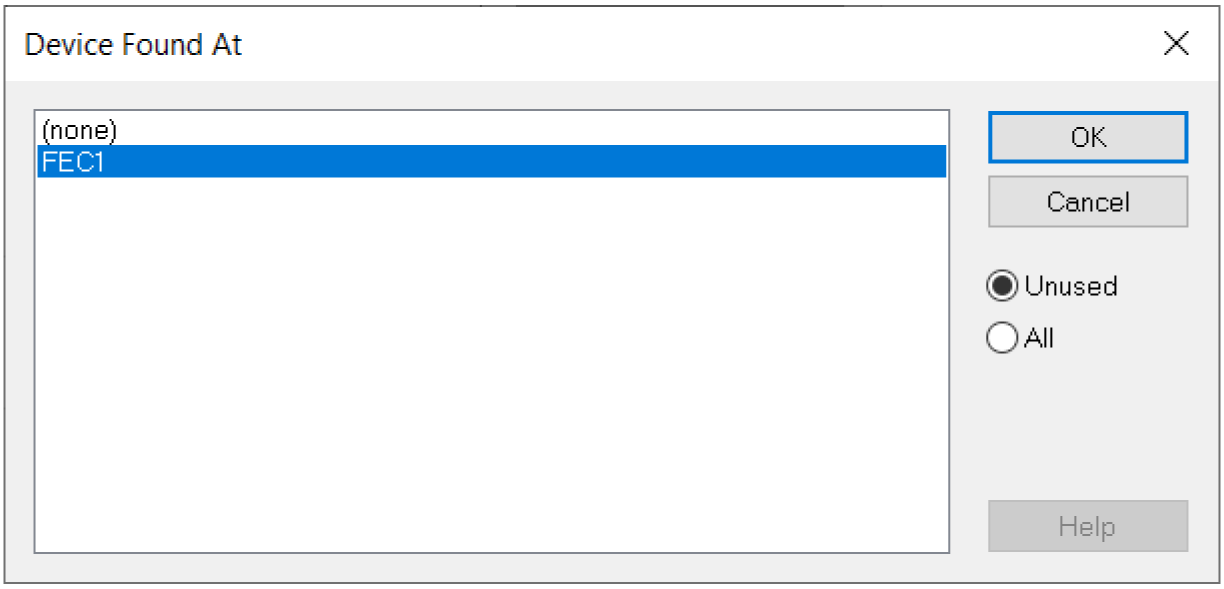
Description:
Device Name:
PCI Bus/Slot:
MAC Address: 00 00 00 00 00 00
IP Address: 0.0.0.0 (0.0.0.0)

☐ Promiscuous Mode (use with Wireshark only)
☐ Virtual Device Names

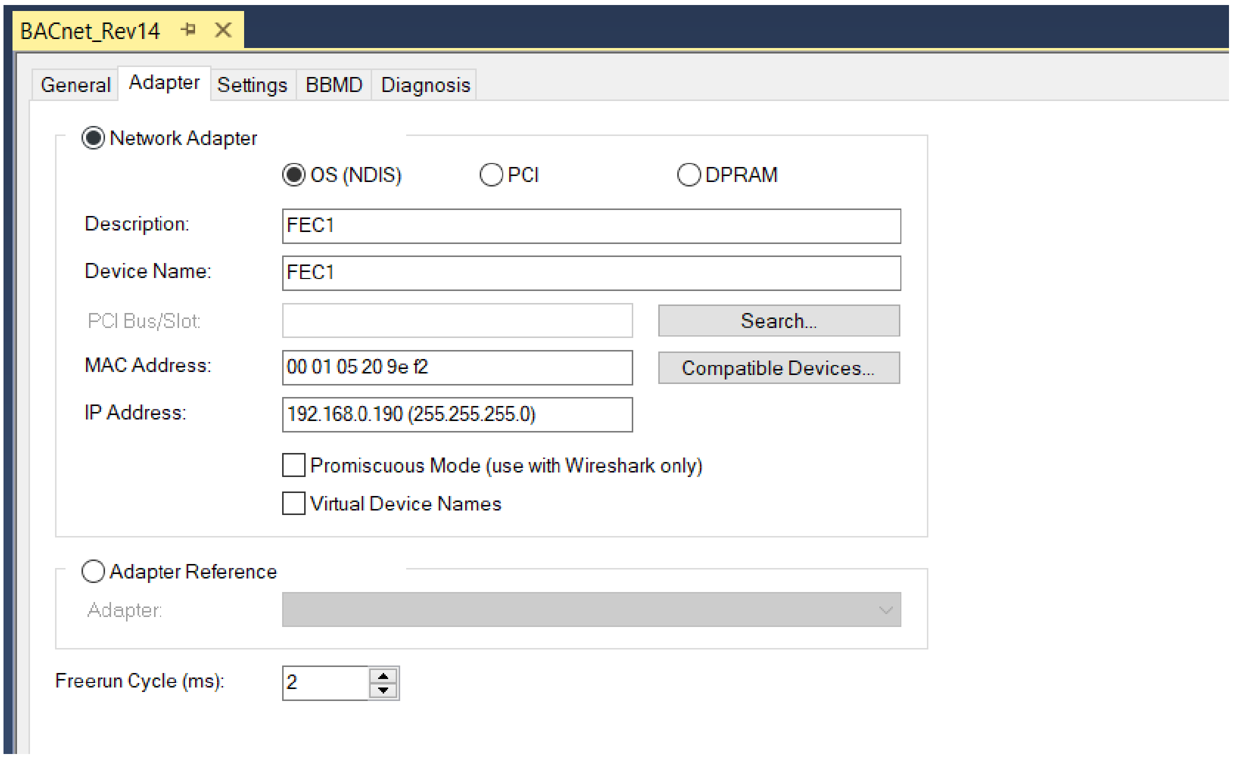
( ) Adapter Reference
Adapter:

Freerun Cycle (ms): 2

2. Select *Search* and select the network interface. Depending on the controller type the names and number of network interfaces may vary.



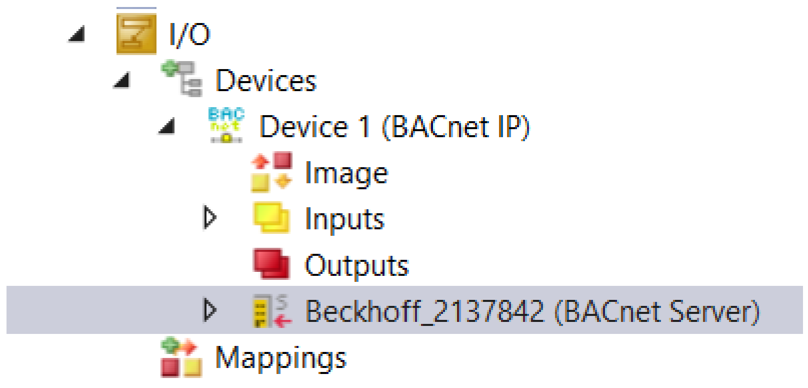
3. Check if the IP address was set correctly.



Description: FEC1
Device Name: FEC1
PCI Bus/Slot:
MAC Address: 00 01 05 20 9e f2
IP Address: 192.168.0.190 (255.255.255.0)
Promiscuous Mode (use with Wireshark only)
Virtual Device Names
Adapter:
Freerun Cycle (ms): 2

4.1.4 Adjusting the BACnet Server settings

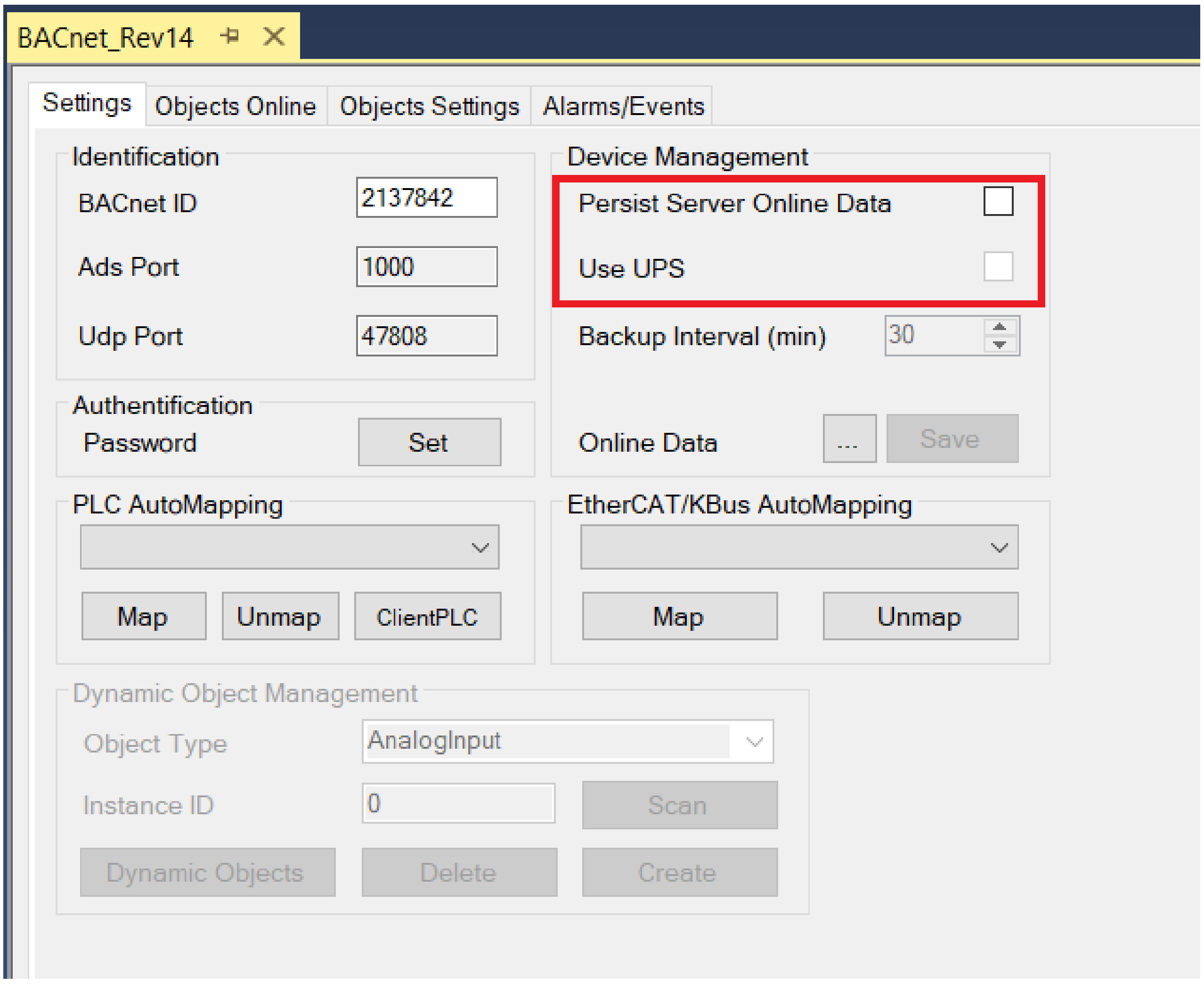
1. Double click to the BACnet server which was created in the Device 1 tree item.


I/O
Devices
Device 1 (BACnet IP)
Image
Inputs
Outputs
Beckhoff_2137842 (BACnet Server)
Mappings

2. Use the input box *BACnet ID* to set the instance number of the BACnet server.

The BACnet instance number is a 22-bit value (valid range from 0 – 4194302, 4194303 is reserved as wildcard and cannot be used). Combined with the object type (10-bit), this information creates the unique address of the device (the Device Object Identifier).

Every device in the network requires a unique Device Object Identifier. The uniqueness is required across the entire BACnet internetwork, this means across all BACnet networks connected by BACnet routers. Talk to the specialist planner or other parties involved in the BACnet installation to avoid duplicate addressing!



Identification

BACnet ID	2137842
Ads Port	1000
Udp Port	47808

Authentification
Password

Device Management

Persist Server Online Data	☐
Use UPS	☐
Backup Interval (min)	30
Online Data	...

PLC AutoMapping

EtherCAT/KBus AutoMapping

Dynamic Object Management

Object Type	AnalogInput
Instance ID	0

4.1.5 Persistence

Persistent storage of values modified by BACnet clients (e. g. from the BMS) is provided as a file in the TwinCAT boot directory (BACnetOnline_1010010.bootdata). To enable persistent storage, please activate “Persist Server Online Data in the BACnet Server settings (as shown above).

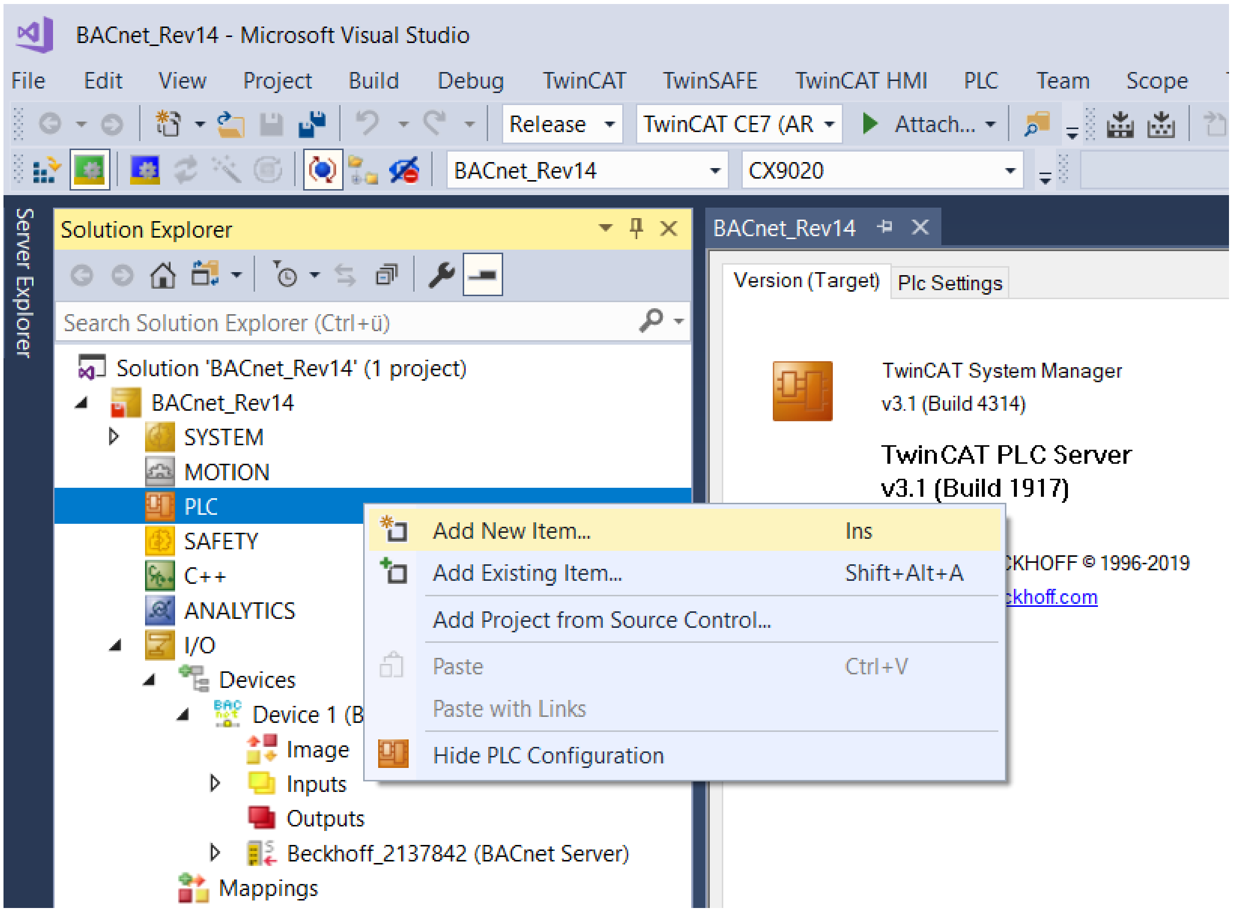
In case a UPS is installed in the controller, the flag Use UPS may be activated. In this case, the UPS function block must assure that persistent data is written in case of a power loss.

4.2 Creating the BACnet PLC project

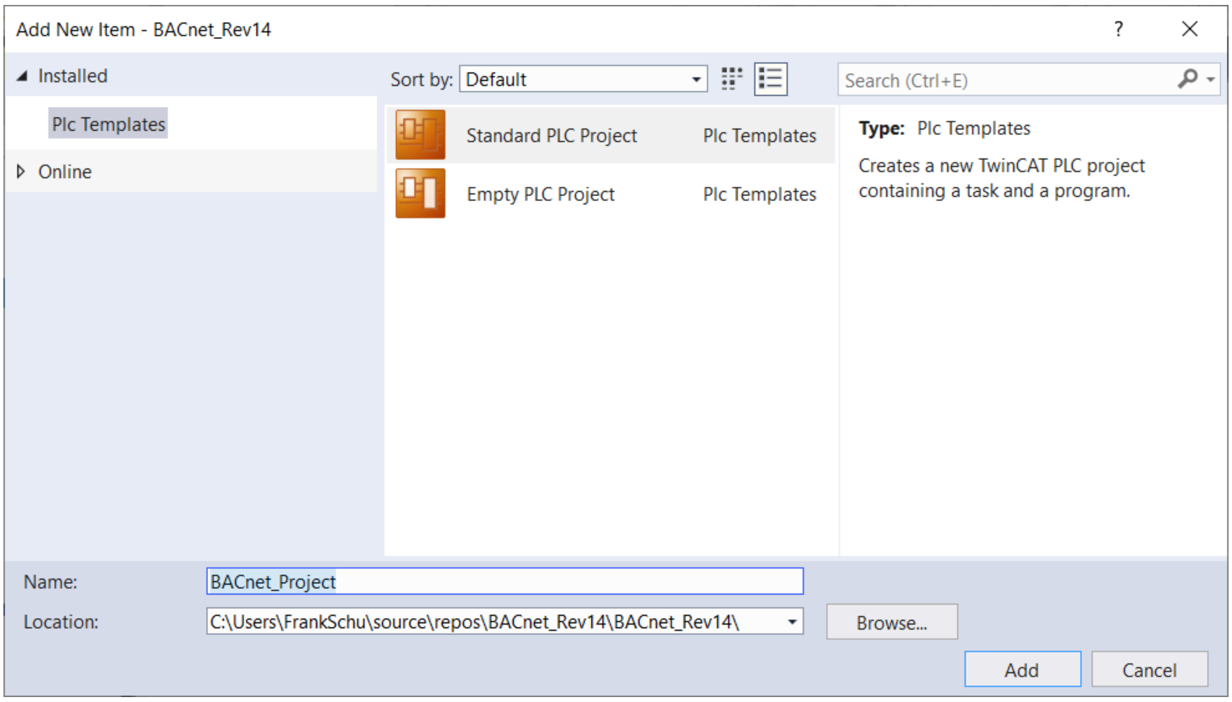
This example shows how to create a first BACnet server project.

4.2.1 Creating the PLC project

1. Right click to *PLC* and select *Add New Item*.



2. Select *Standard PLC Project* and name the project.



Add New Item - BACnet_Rev14

Standard PLC Project — Plc Templates

Empty PLC Project — Plc Templates

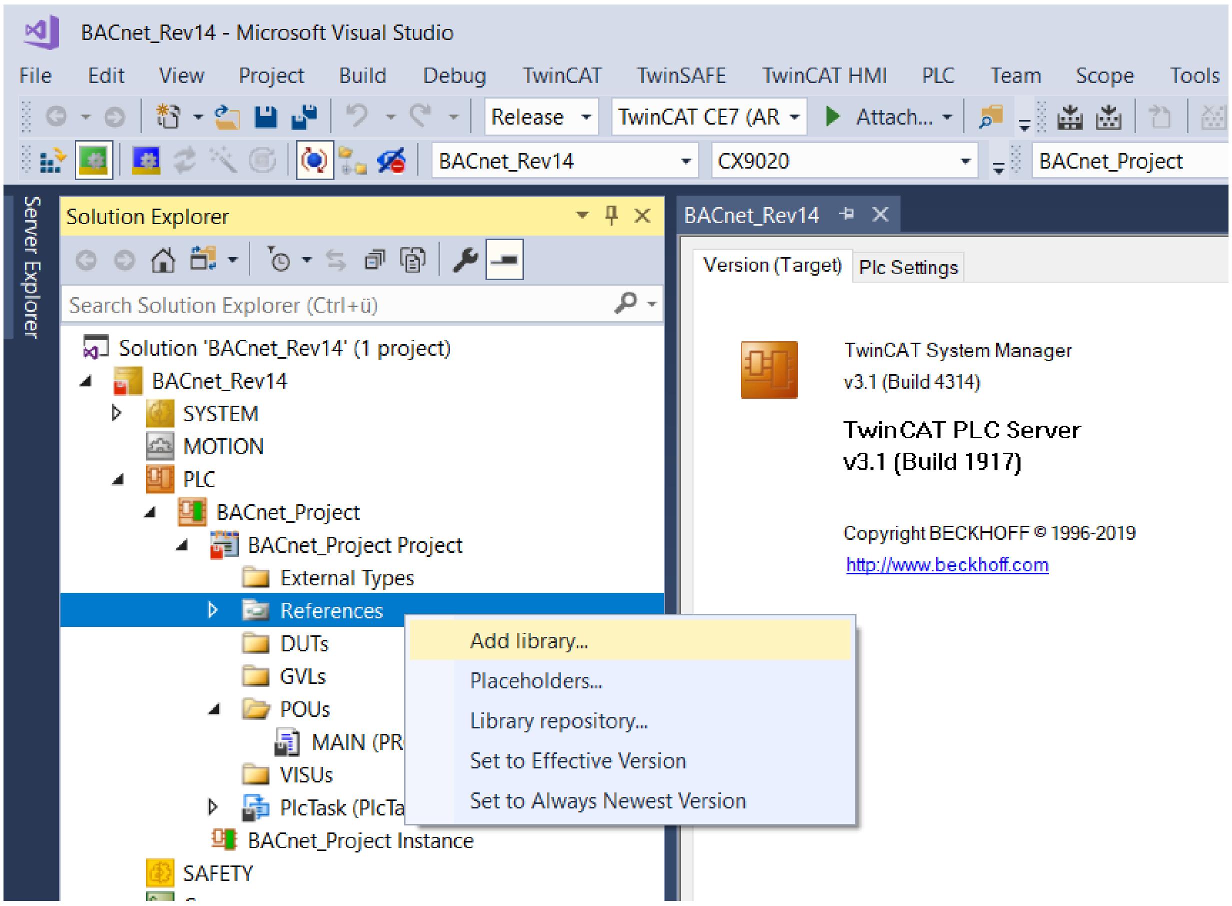
Type: Plc Templates

Creates a new TwinCAT PLC project containing a task and a program.

Name: BACnet_Project

Location: C:\Users\FrankSchu\source\repos\BACnet_Rev14\BACnet_Rev14\

3. Right click to *References* and select *Add Library* to add libraries to a PLC project.



BACnet_Rev14 - Microsoft Visual Studio

TwinCAT System Manager v3.1 (Build 4314)

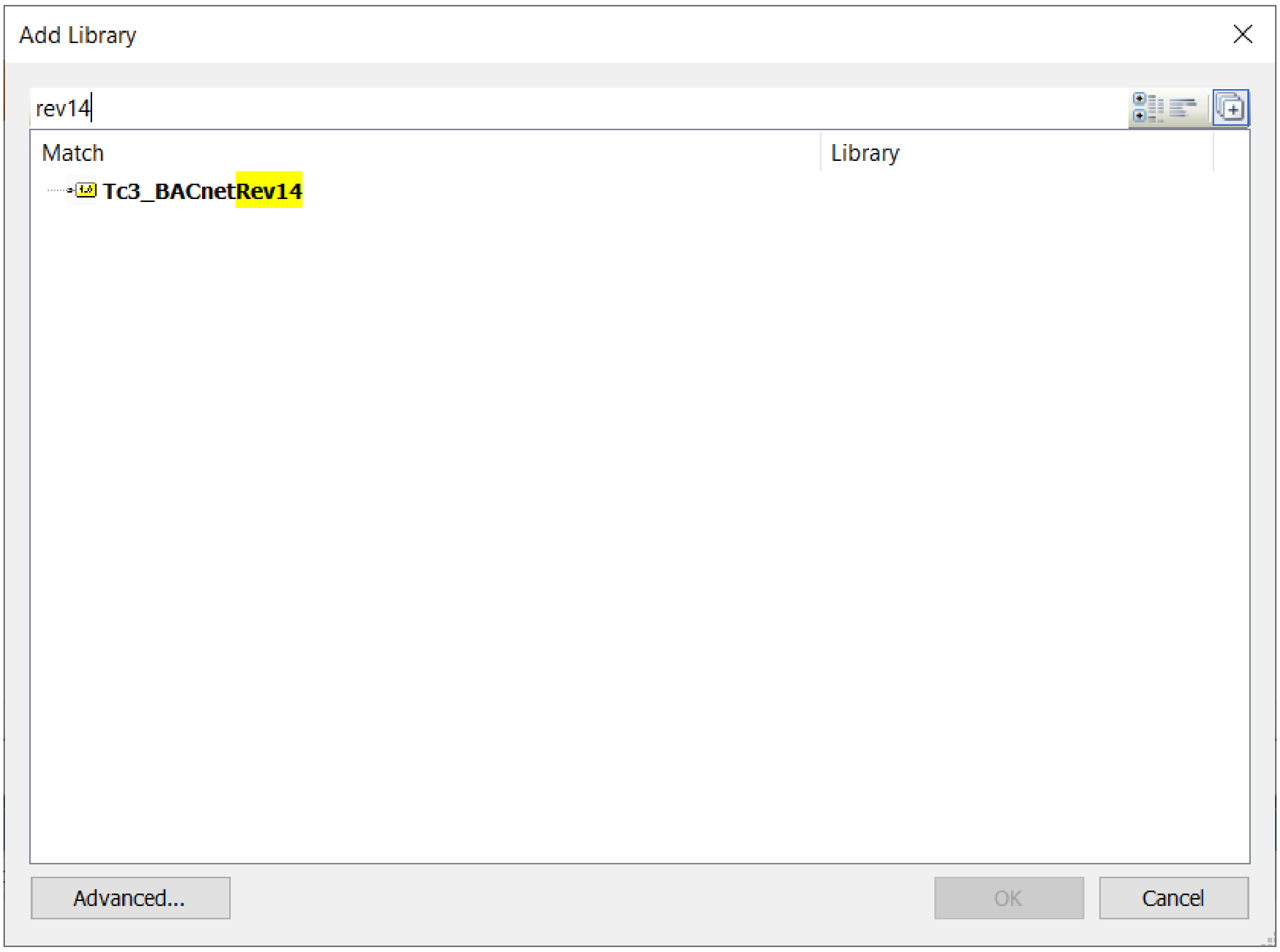
TwinCAT PLC Server v3.1 (Build 1917)

Copyright BECKHOFF © 1996-2019

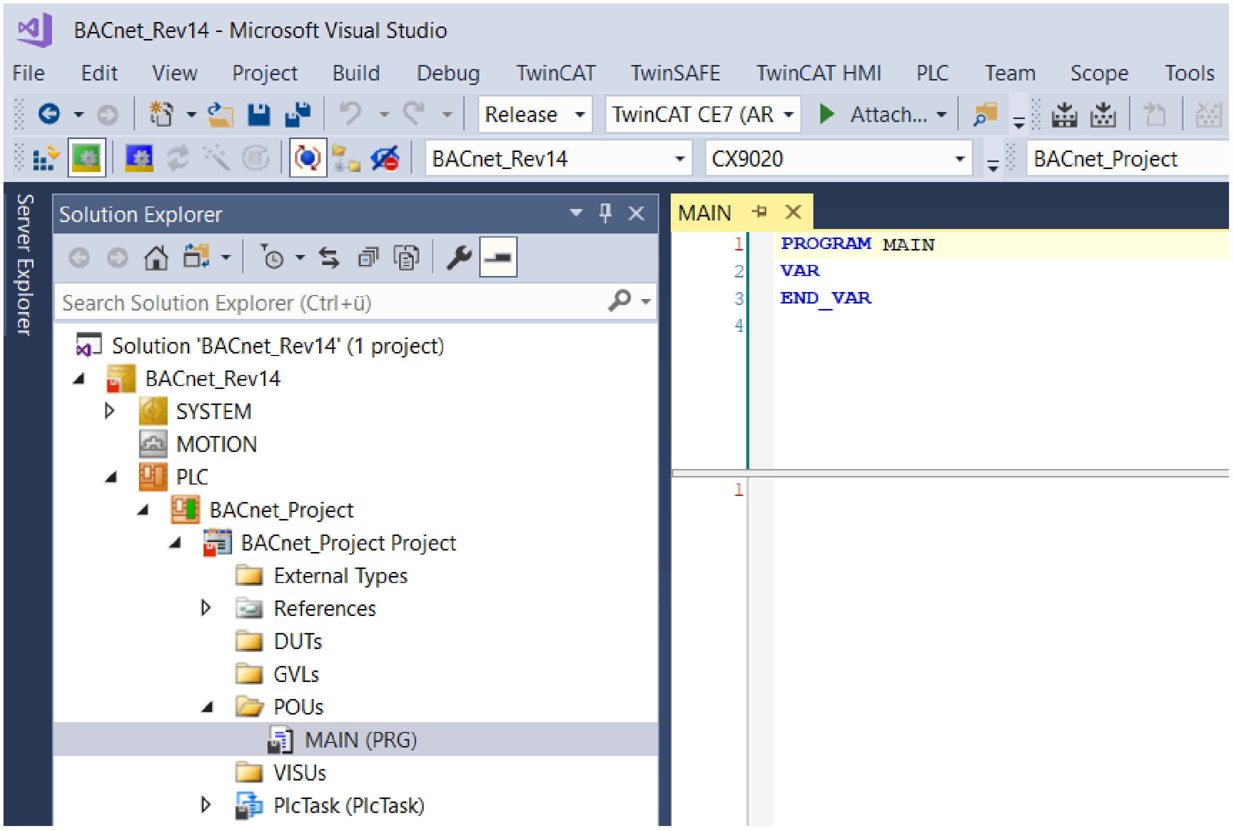
http://www.beckhoff.com

4. Select the library *Tc3_BACnetRev14*.

If you enter the text *rev14* in the full-text search field, only this library will be displayed. Select this item and click *OK*.



5. Double click the POU MAIN in the PLC project.



```
PROGRAM MAIN
VAR
END_VAR
```

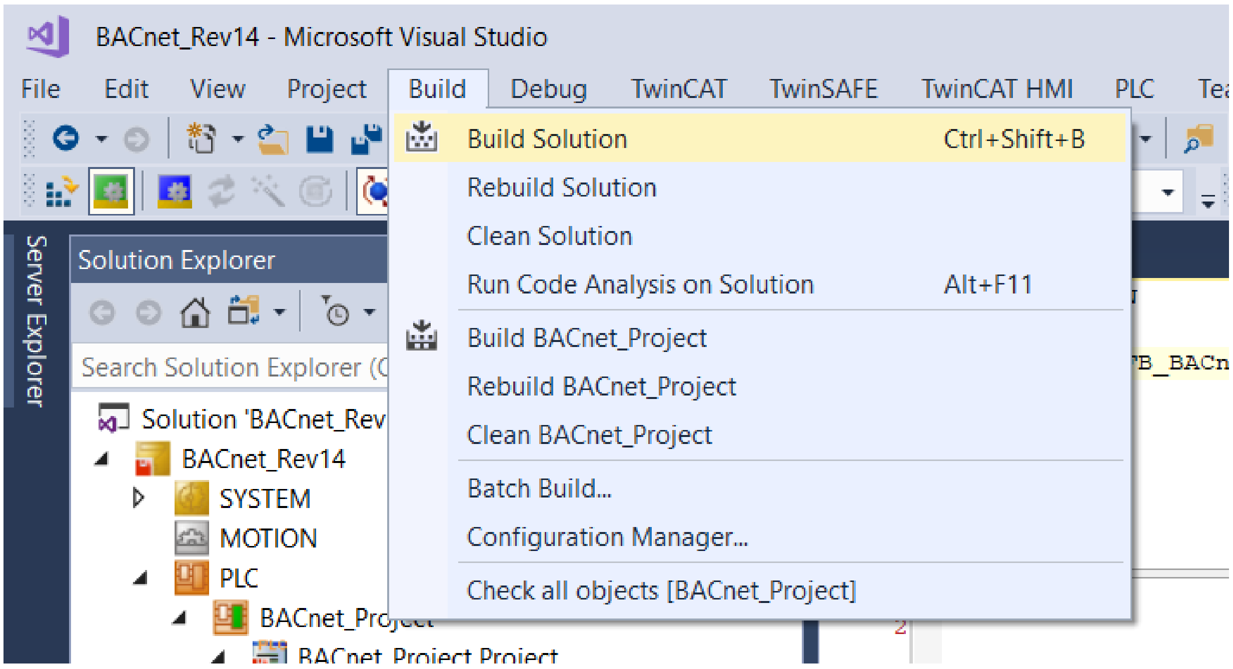
6. Add the following declaration in the variable window: *fbAv : FB_BACnet_AV,*

⇨ This command creates an instance of the function block FB_BACnet_AV with the name fbAv (which represents an Analog Value object).

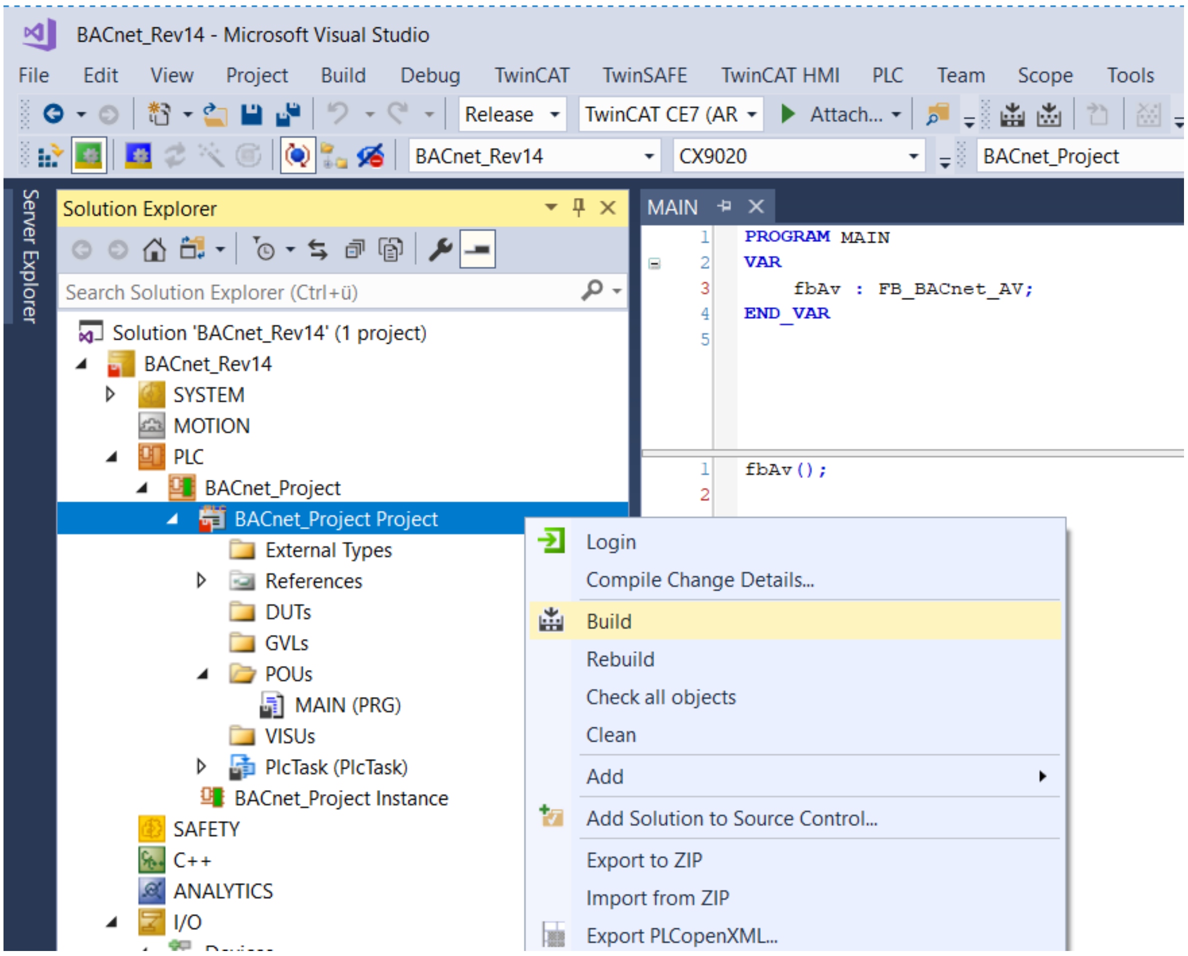
7. Add the following entry to the code window: *fbAv();*

   ⇨ This code is called at regular intervals according to the cycle time (default: 10 ms).

8. Build the project by selecting *Build / Build Solution*.



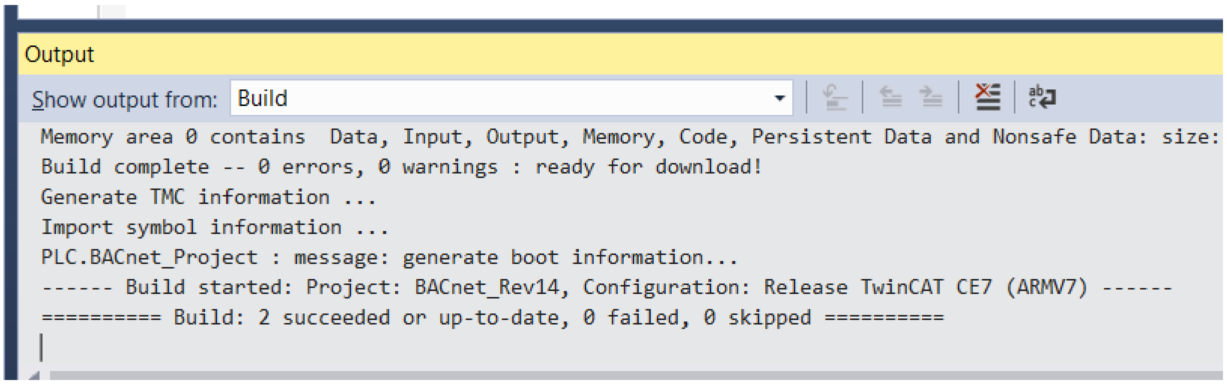
⇨ Alternatively, you can also call this function by right-clicking on the project name.



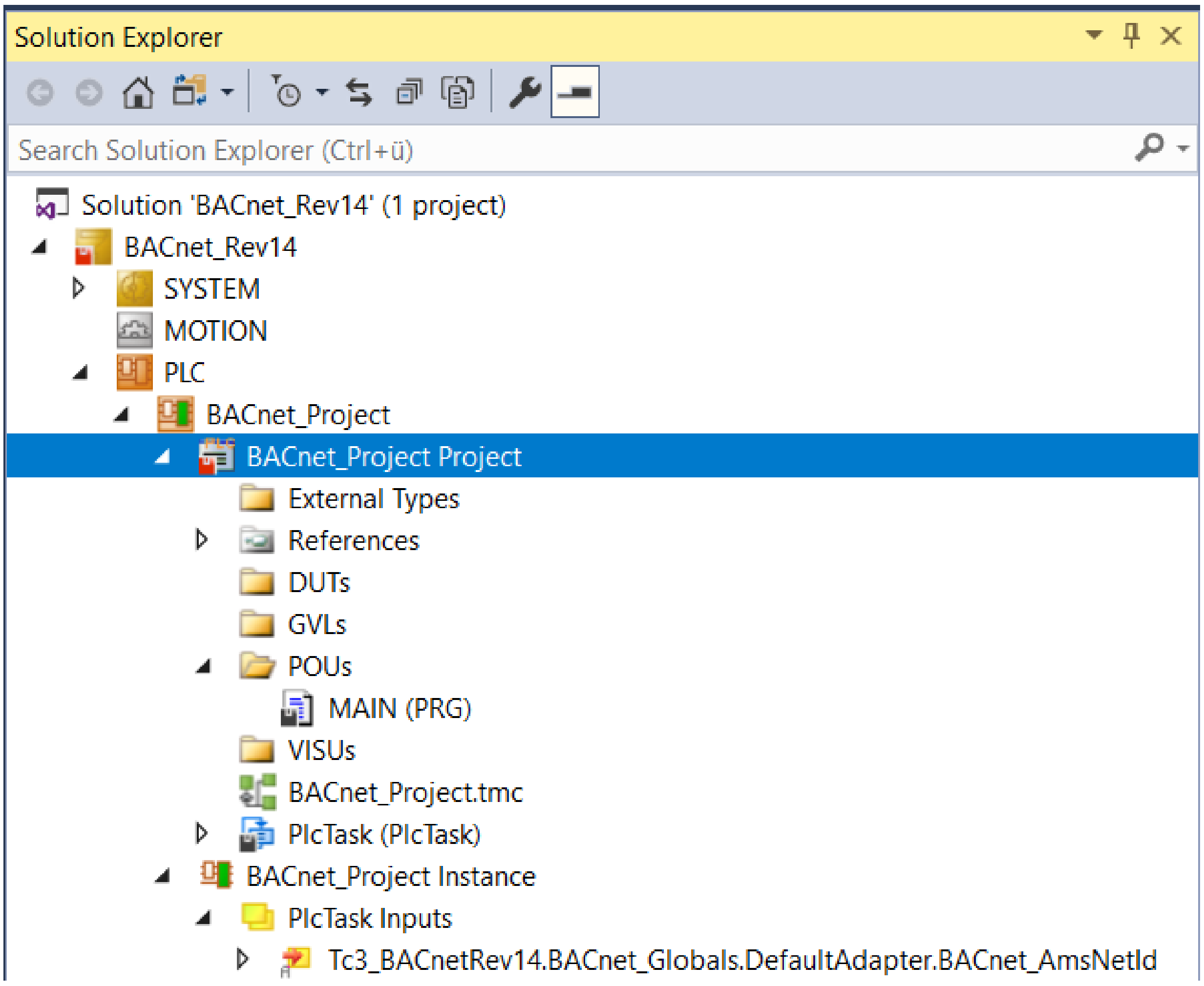
```
PROGRAM MAIN
VAR
    fbAv : FB_BACnet_AV;
END_VAR
```

```
fbAv();
```

9. Check if the build was successful and no error is displayed. If errors are displayed, check the project and correct the errors.
10. After successful compilation the variable *Tc3_BACnetRev14.BACnet_Globals.DefaultAdapter.BACnet_AmsNetId* is displayed in the process image.



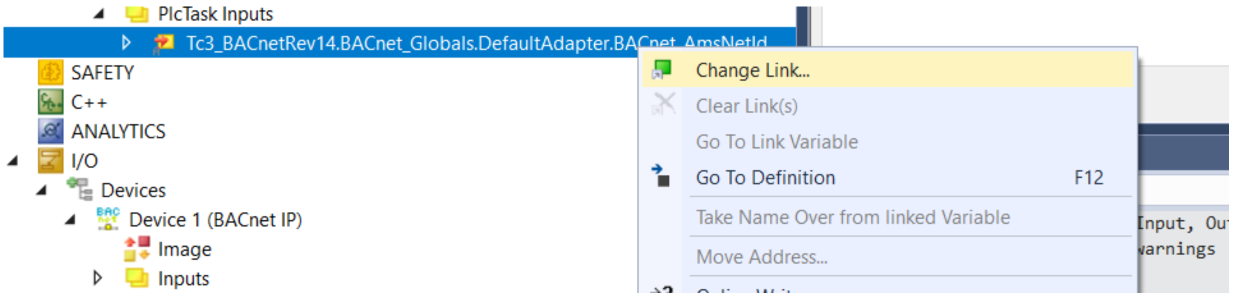
```
Memory area 0 contains  Data, Input, Output, Memory, Code, Persistent Data and Nonsafe Data: size:
Build complete -- 0 errors, 0 warnings : ready for download!
Generate TMC information ...
Import symbol information ...
PLC.BACnet_Project : message: generate boot information...
------ Build started: Project: BACnet_Rev14, Configuration: Release TwinCAT CE7 (ARMV7) ------
========== Build: 2 succeeded or up-to-date, 0 failed, 0 skipped ==========
```



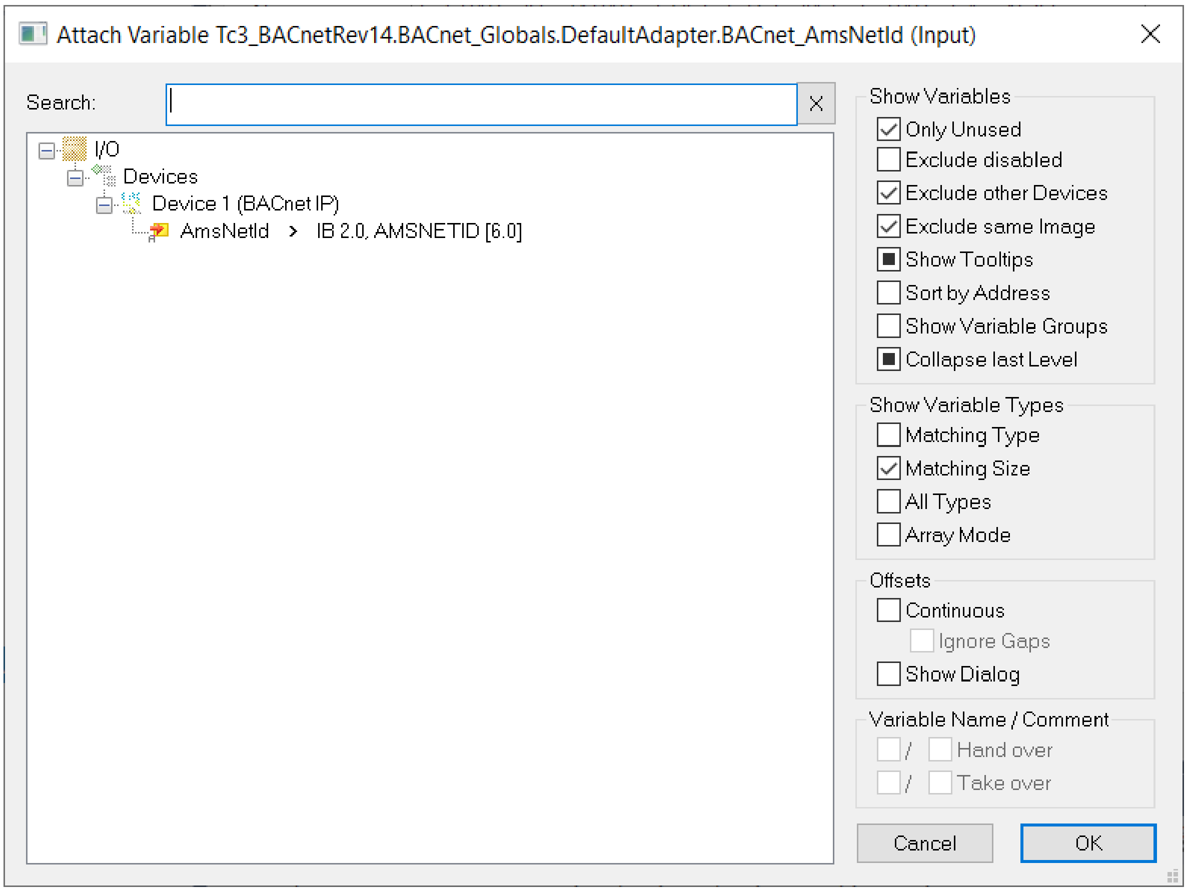
```
Solution 'BACnet_Rev14' (1 project)
  BACnet_Rev14
    SYSTEM
    MOTION
    PLC
      BACnet_Project
        BACnet_Project Project
          External Types
          References
          DUTs
          GVLs
          POUs
            MAIN (PRG)
          VISUs
          BACnet_Project.tmc
          PlcTask (PlcTask)
        BACnet_Project Instance
          PlcTask Inputs
            Tc3_BACnetRev14.BACnet_Globals.DefaultAdapter.BACnet_AmsNetId
```

⇨ This variable is used to establish the BACnet connection between the PLC program and the BACnet adapter.

11. Right click to this variable and select *Change Link*.



12. Select the variable of the same type in the BACnet adapter and create a link with OK.
    ⇨ The connection is indicated by a connection icon.


Attach Variable Tc3_BACnetRev14.BACnet_Globals.DefaultAdapter.BACnet_AmsNetId (Input)

Search:

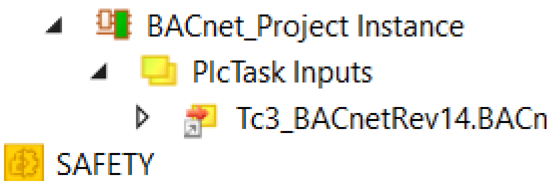
I/O
Devices
Device 1 (BACnet IP)
AmsNetId > IB 2.0, AMSNETID [6.0]

Show Variables
[✓] Only Unused
[ ] Exclude disabled
[✓] Exclude other Devices
[✓] Exclude same Image
[■] Show Tooltips
[ ] Sort by Address
[ ] Show Variable Groups
[■] Collapse last Level

Show Variable Types
[ ] Matching Type
[✓] Matching Size
[ ] All Types
[ ] Array Mode

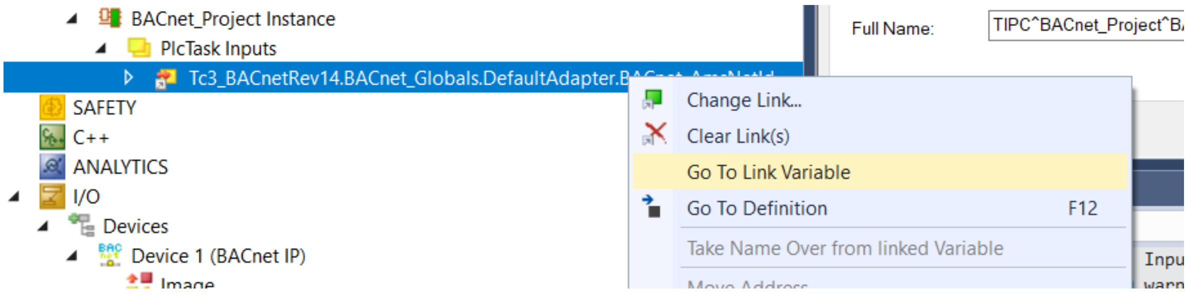
Offsets
[ ] Continuous
[ ] Ignore Gaps
[ ] Show Dialog

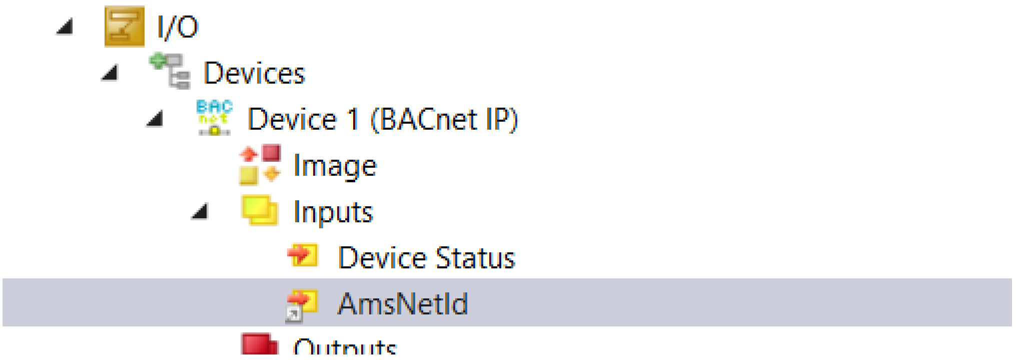
Variable Name / Comment
[ ] / [ ] Hand over
[ ] / [ ] Take over



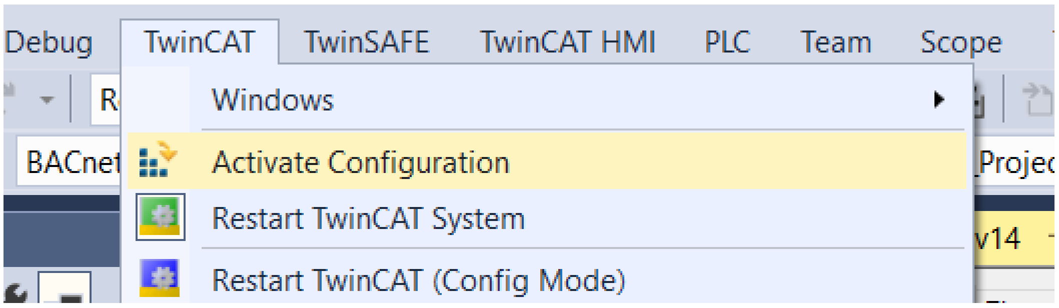
13. You can check the link at any time with the function *Go To Link* Variable.

⇨ Using this function, the cursor moves to the connected variable.

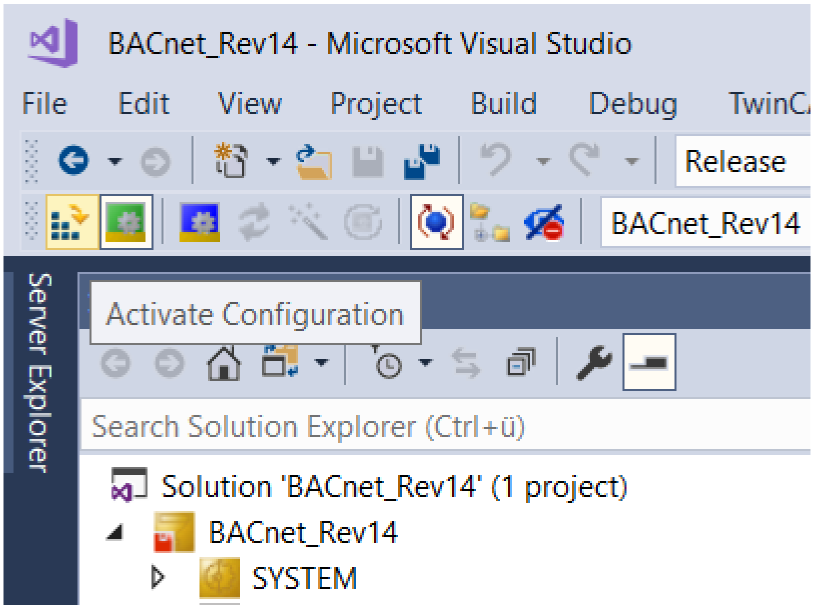




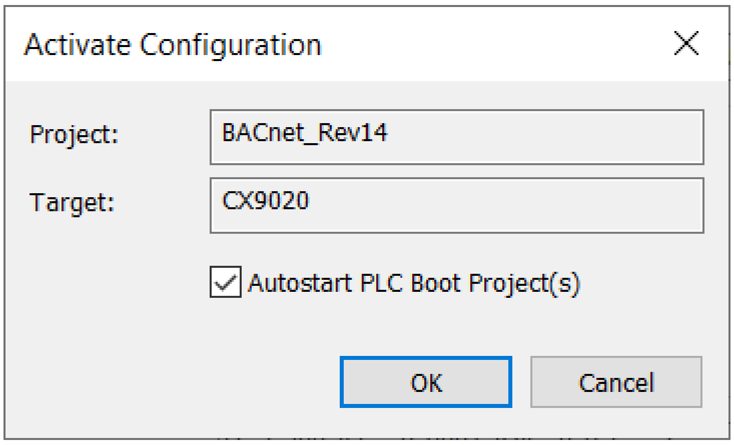
14. Activate the project using the function *Activate Configuration* in the TwinCAT menu.



⇨ Alternatively this function is also available in the TwinCAT toolbar.


BACnet_Rev14 - Microsoft Visual Studio
File
Edit
View
Project
Build
Debug
Release
BACnet_Rev14
Activate Configuration
Server Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'BACnet_Rev14' (1 project)
BACnet_Rev14
SYSTEM

15. Check that the correct target system is selected and click OK.

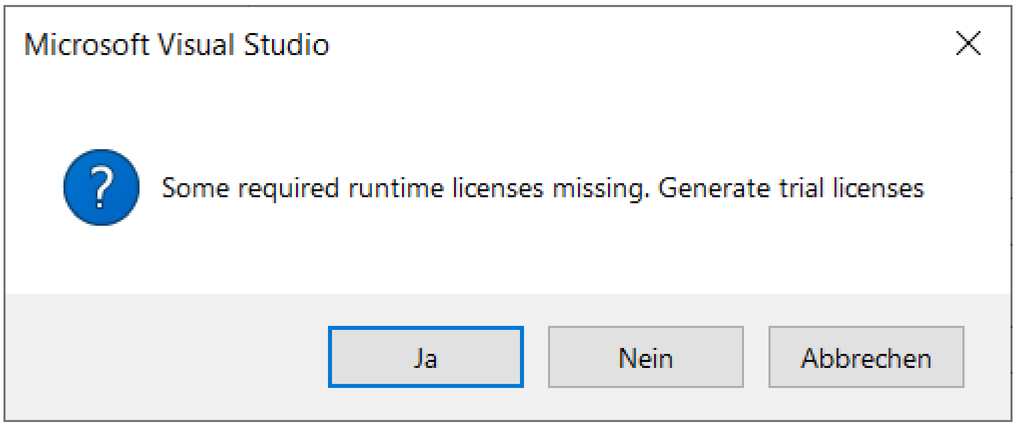

Activate Configuration

Project: BACnet_Rev14

Target: CX9020

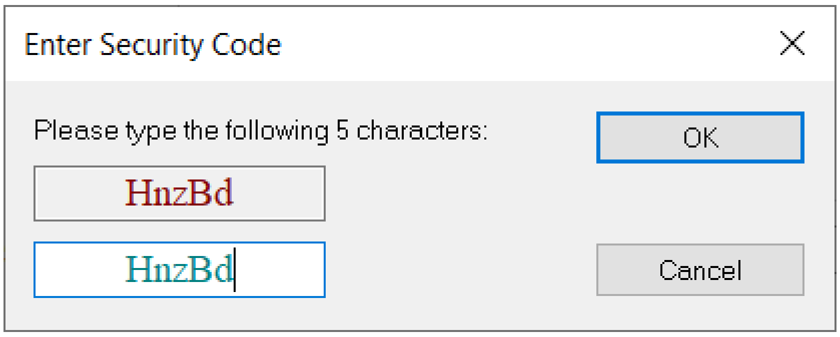
☑ Autostart PLC Boot Project(s)

⇨ If no licenses are available on the target system, so-called trial licenses can be generated. Trial licenses are valid for a period of 7 days. The Beckhoff Information System provides further information on TwinCAT licensing.


Microsoft Visual Studio

Some required runtime licenses missing. Generate trial licenses

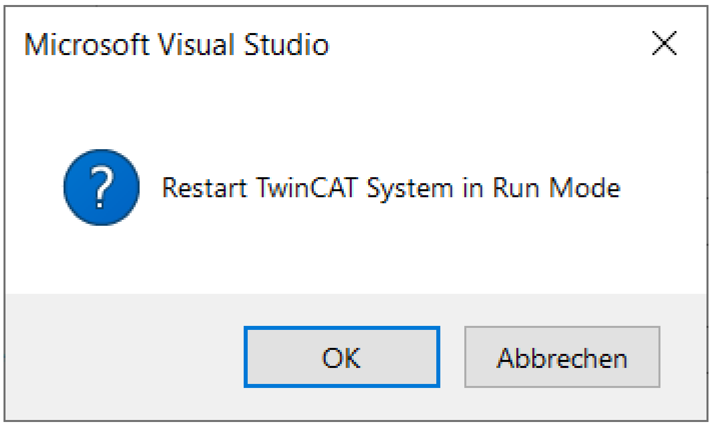
16. Enter the 5 letters from the upper field into the lower field.
The entry is case sensitive.


Enter Security Code

Please type the following 5 characters:

HnzBd

HnzBd


Microsoft Visual Studio

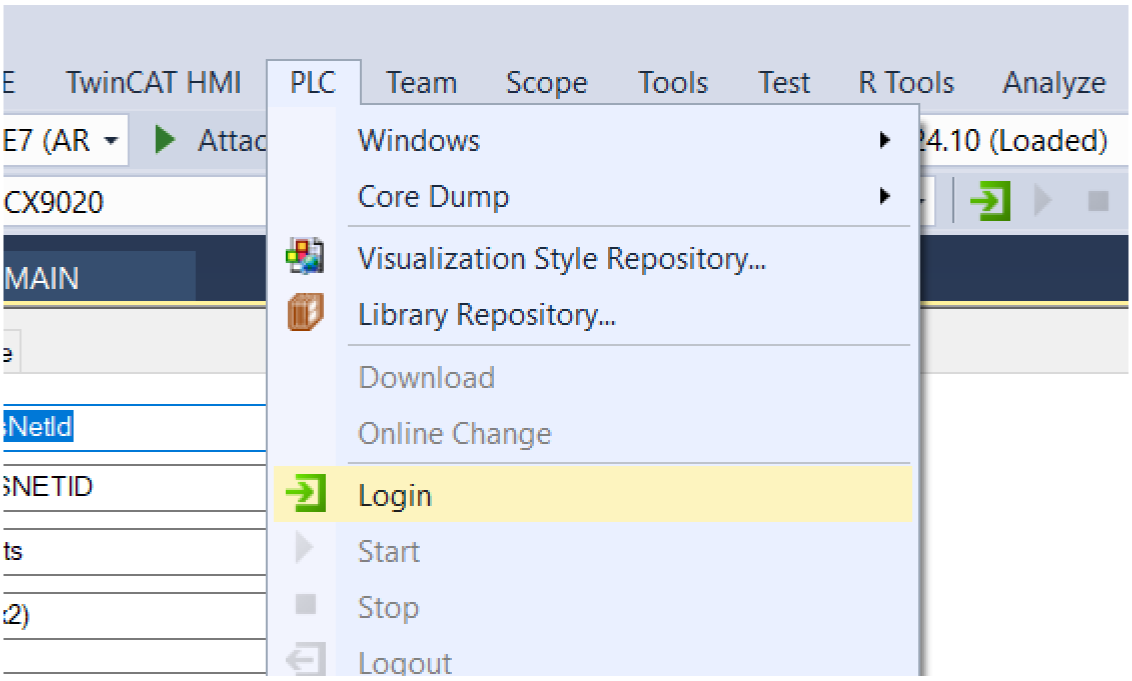
Restart TwinCAT System in Run Mode

17. To complete this process, TwinCAT requests a restart of the PLC. Click OK.

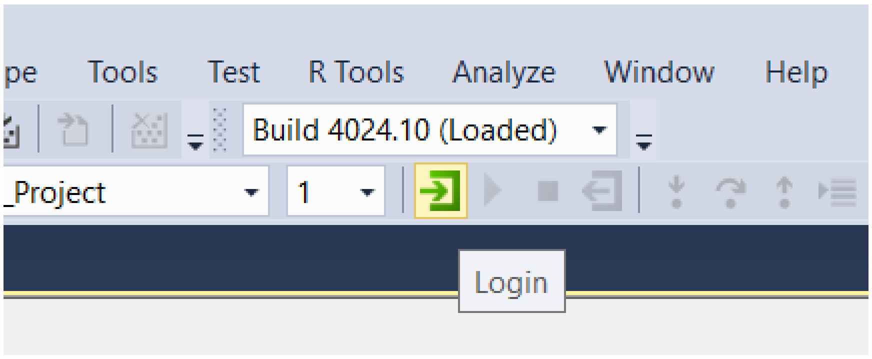
⇨ With these few and simple steps, a fully functional BACnet server was created. The server and the contained objects are now available in the BACnet network.

4.2.2 Testing the BACnet server

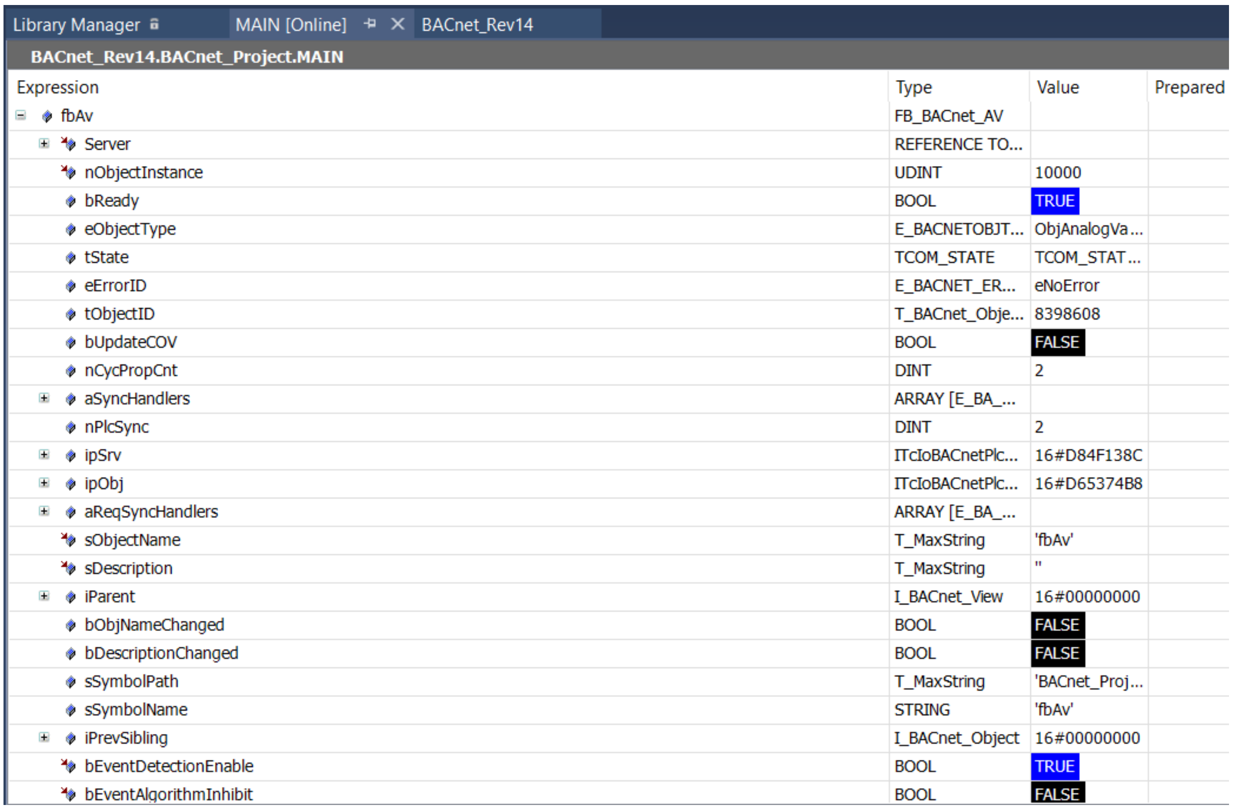
1. Login into the PLC. Select *Login* from the PLC menu.



⇨ Alternatively, this function is also available via the TwinCAT toolbar.



2. Navigate to the POU MAIN, expand the display of the variable *fbAv* with the plus sign and check the content of this function block representing the BACnet object.


Library Manager | MAIN [Online] | BACnet_Rev14

BACnet_Rev14.BACnet_Project.MAIN

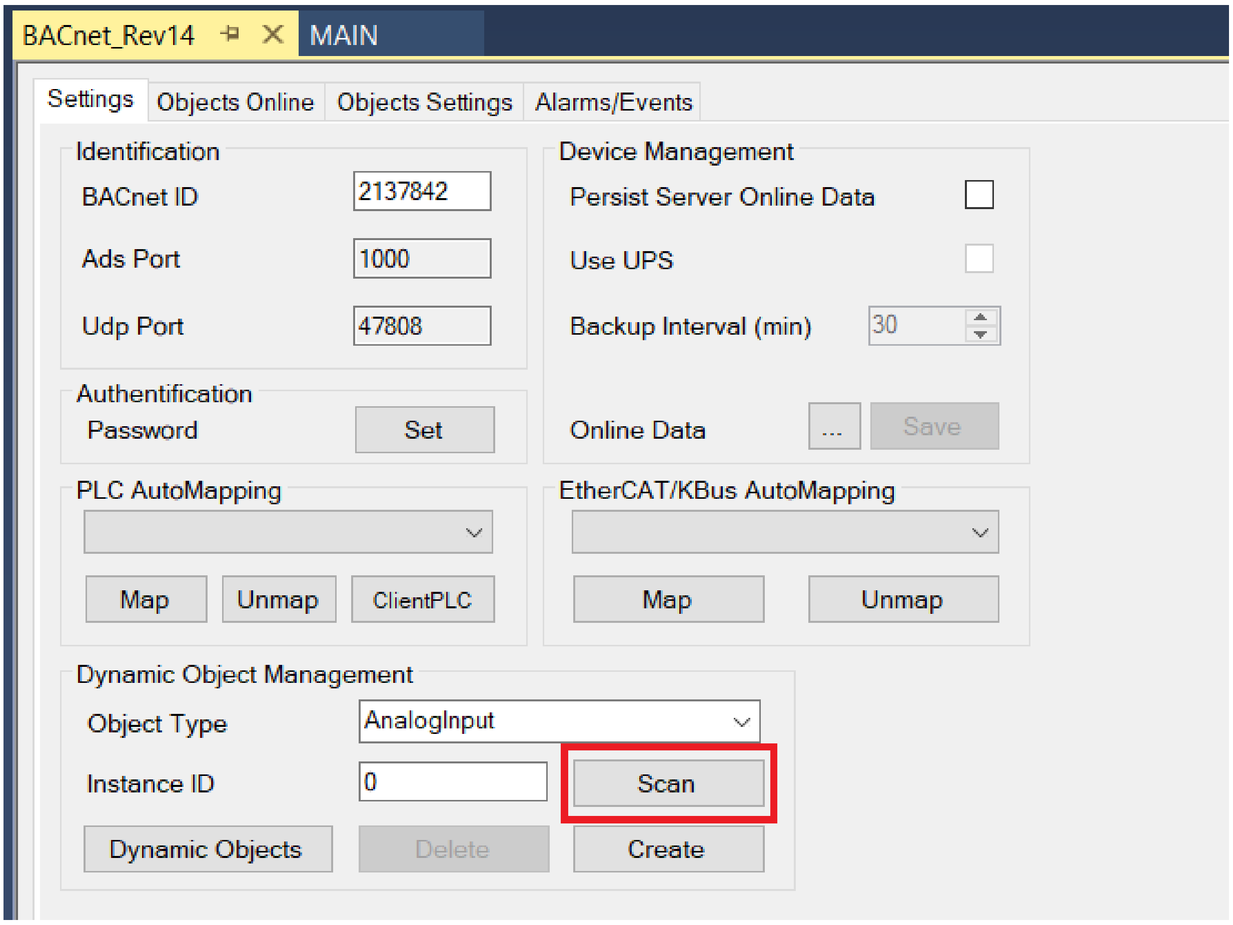
Expression	Type	Value	Prepared
fbAv	FB_BACnet_AV		
Server	REFERENCE TO...		
nObjectInstance	UDINT	10000	
bReady	BOOL	TRUE	
eObjectType	E_BACNETOBJT...	ObjAnalogVa...	
tState	TCOM_STATE	TCOM_STAT...	
eErrorID	E_BACNET_ER...	eNoError	
tObjectID	T_BACnet_Obje...	8398608	
bUpdateCOV	BOOL	FALSE	
nCycPropCnt	DINT	2	
aSyncHandlers	ARRAY [E_BA_...		
nPlcSync	DINT	2	
ipSrv	ITcIoBACnetPlc...	16#D84F138C	
ipObj	ITcIoBACnetPlc...	16#D65374B8	
aReqSyncHandlers	ARRAY [E_BA_...		
sObjectName	T_MaxString	'fbAv'	
sDescription	T_MaxString	''	
iParent	I_BACnet_View	16#00000000	
bObjNameChanged	BOOL	FALSE	
bDescriptionChanged	BOOL	FALSE	
sSymbolPath	T_MaxString	'BACnet_Proj...	
sSymbolName	STRING	'fbAv'	
iPrevSibling	I_BACnet_Object	16#00000000	
bEventDetectionEnable	BOOL	TRUE	
bEventAlgorithmInhibit	BOOL	FALSE	

⇨ The variable *bReady* indicates a successful start of this object. In this example, no specific configuration of the object properties has been made, so the property values are default values.

4.2.3 Testing BACnet using the System Manager

The BACnet objects created by the PLC are similar to the objects created dynamically by BACnet clients using the *Create Object* service. Follow these instructions to scan dynamic objects in the System Manager:

1. Double click the BACnet server and navigate to the *Settings* page. Use the function *Scan* in the section *Dynamic Object Management*.



BACnet_Rev14 | MAIN

Settings | Objects Online | Objects Settings | Alarms/Events

Identification
BACnet ID: 2137842
Ads Port: 1000
Udp Port: 47808

Authentification
Password: Set

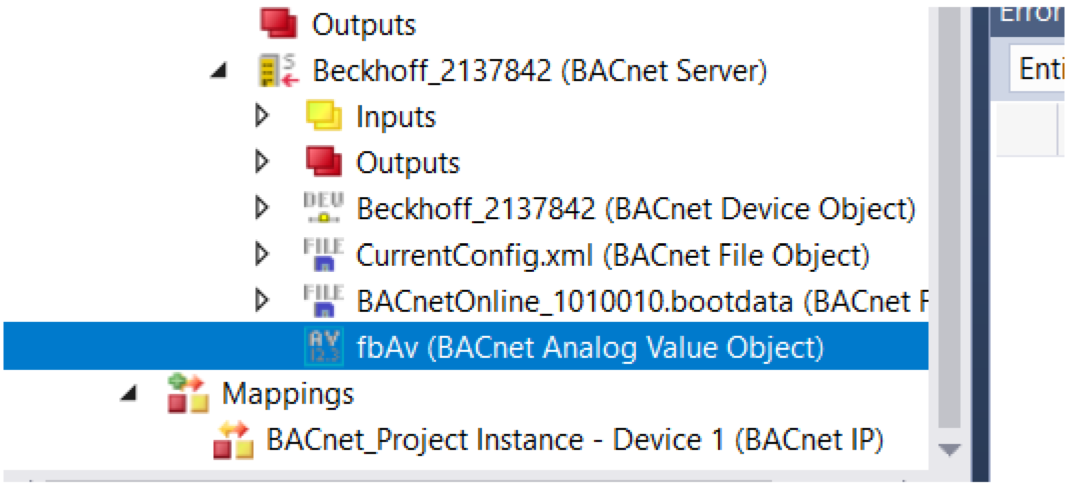
Device Management
Persist Server Online Data: ☐
Use UPS: ☐
Backup Interval (min): 30
Online Data: ... | Save

PLC AutoMapping
Map | Unmap | ClientPLC

EtherCAT/KBus AutoMapping
Map | Unmap

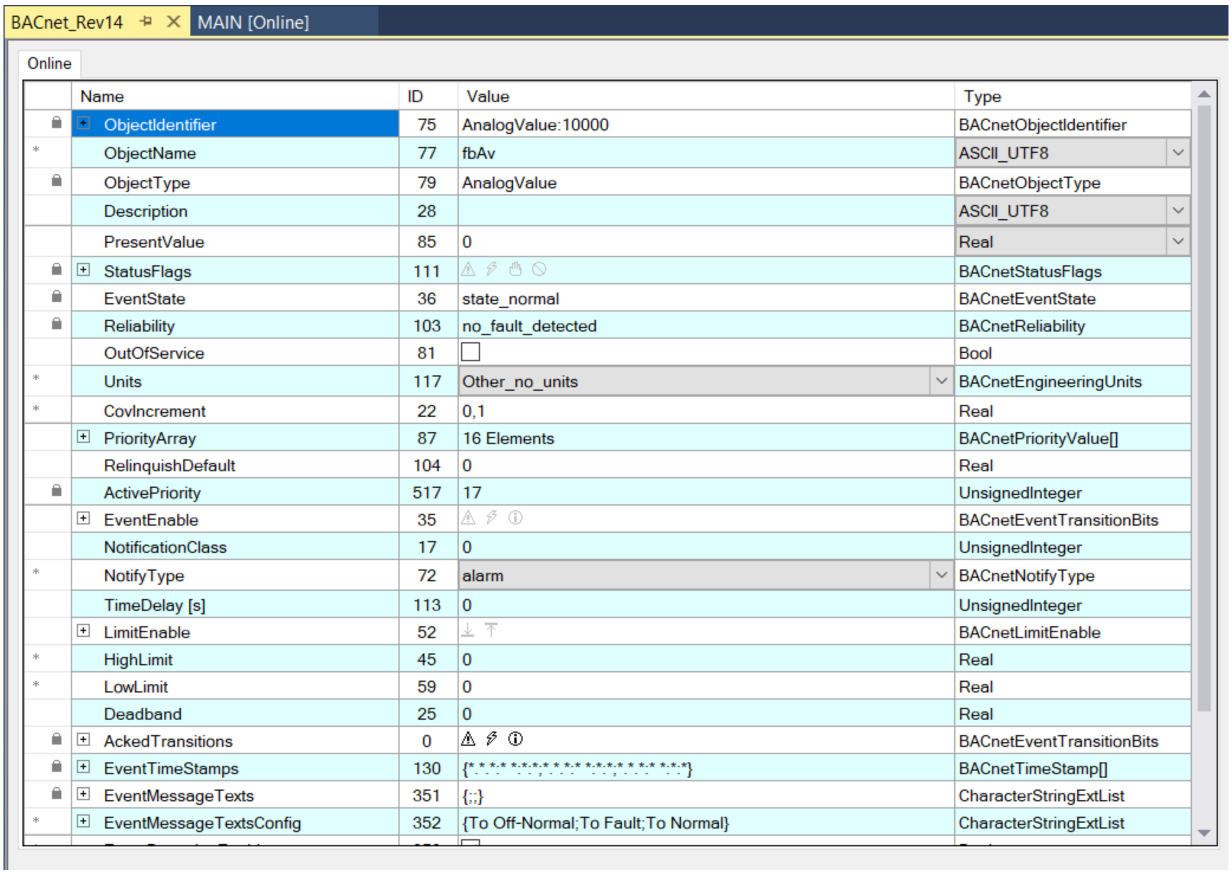
Dynamic Object Management
Object Type: AnalogInput
Instance ID: 0 | Scan
Dynamic Objects | Delete | Create

⇨ The tree below the BACnet Server now shows the object *fbAv* (the one created in the example above).



Outputs
Beckhoff_2137842 (BACnet Server)
Inputs
Outputs
Beckhoff_2137842 (BACnet Device Object)
CurrentConfig.xml (BACnet File Object)
BACnetOnline_1010010.bootdata (BACnet F
fbAv (BACnet Analog Value Object)
Mappings
BACnet_Project Instance - Device 1 (BACnet IP)

2. Double-click on this object.



	Name	ID	Value	Type
🔒	ObjectIdentifier	75	AnalogValue:10000	BACnetObjectIdentifier
*	ObjectName	77	fbAv	ASCII_UTF8
🔒	ObjectType	79	AnalogValue	BACnetObjectType
	Description	28		ASCII_UTF8
	PresentValue	85	0	Real
🔒	StatusFlags	111	⚠ ⚡ ✋ ⊘	BACnetStatusFlags
🔒	EventState	36	state_normal	BACnetEventState
🔒	Reliability	103	no_fault_detected	BACnetReliability
	OutOfService	81	☐	Bool
*	Units	117	Other_no_units	BACnetEngineeringUnits
*	CovIncrement	22	0,1	Real
	PriorityArray	87	16 Elements	BACnetPriorityValue[]
	RelinquishDefault	104	0	Real
🔒	ActivePriority	517	17	UnsignedInteger
	EventEnable	35	⚠ ⚡ ⓘ	BACnetEventTransitionBits
	NotificationClass	17	0	UnsignedInteger
*	NotifyType	72	alarm	BACnetNotifyType
	TimeDelay [s]	113	0	UnsignedInteger
	LimitEnable	52	⊥ ⊤	BACnetLimitEnable
*	HighLimit	45	0	Real
*	LowLimit	59	0	Real
	Deadband	25	0	Real
🔒	AckedTransitions	0	⚠ ⚡ ⓘ	BACnetEventTransitionBits
🔒	EventTimeStamps	130	{*.*.* *:*:*;*.*.* *:*:*;*.*.* *:*:*}	BACnetTimeStamp[]
🔒	EventMessageTexts	351	{;;}	CharacterStringExtList
*	EventMessageTextsConfig	352	{To Off-Normal;To Fault;To Normal}	CharacterStringExtList

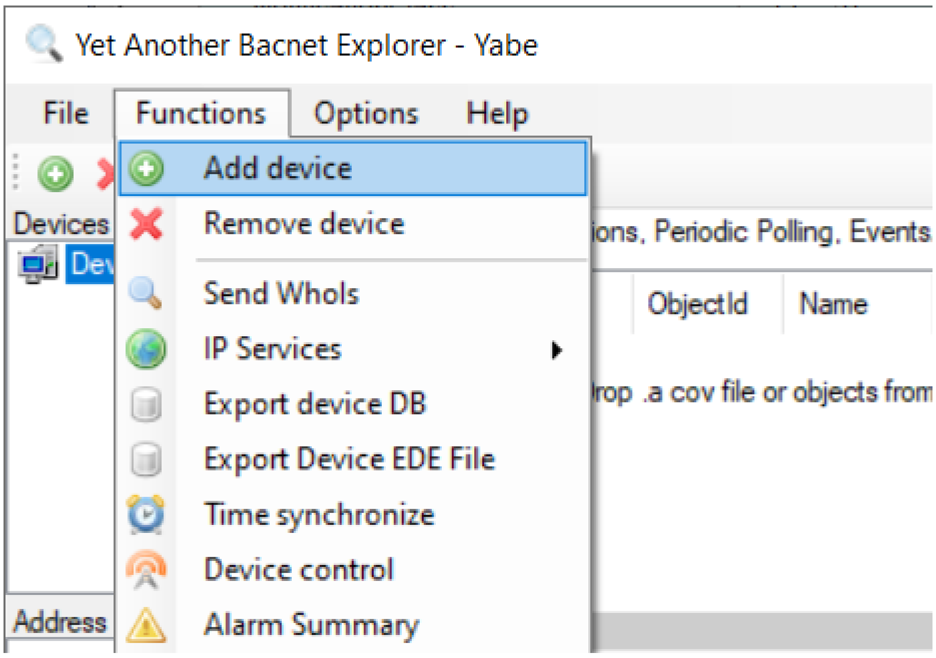
⇨ The online view of this object opens.

4.2.4 Testing BACnet using a BACnet Explorer

In this example, the tool YABE - **Y**et **A**nother **BA**Cnet **E**xplorer is used. The tool can be downloaded from here: https://sourceforge.net/projects/yetanotherbacnetexplorer/.

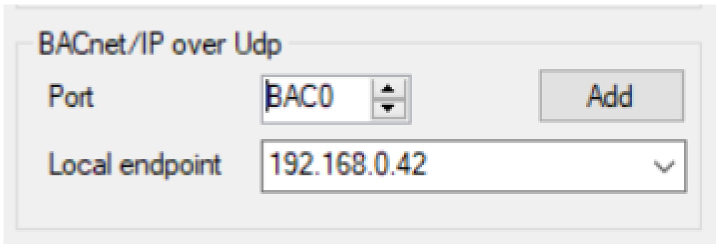
✓ Start YABE.

1. Select *Add device*. This function can be accessed via the *Functions* menu , the green button with the plus sign or by right-clicking on *Devices*.



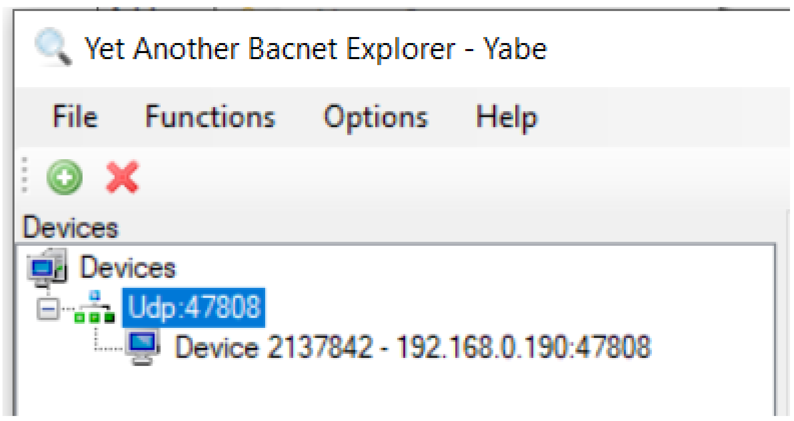
⇨ A dialog named *Scan* opens.

2. Select in the category BACnet/IP over UDP the standard BACnet port BAC0 (corresponds to 47808 in decimal notation) and select the network adapter of your engineering PC (*Local endpoint*).
3. Select the *Add* button (the button to the right of the BACnet port).



BACnet/IP over Udp
Port: BAC0
Local endpoint: 192.168.0.42

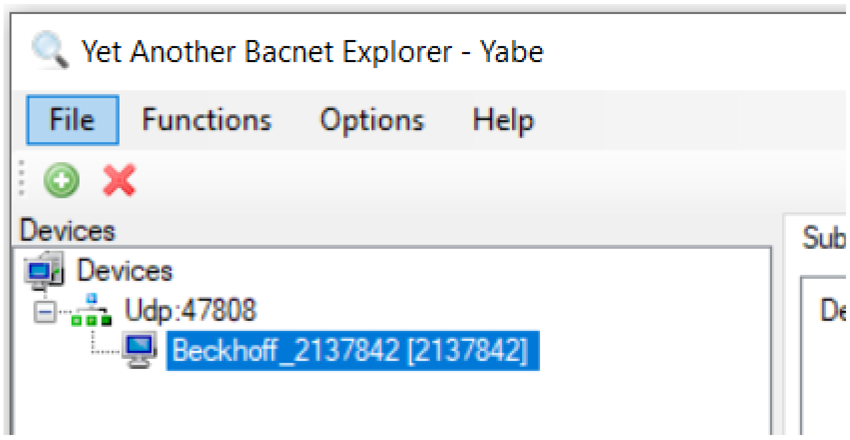
⇨ The device created in TwinCAT should now appear.



Yet Another Bacnet Explorer - Yabe

Devices
- Devices
  - Udp:47808
    - Device 2137842 - 192.168.0.190:47808

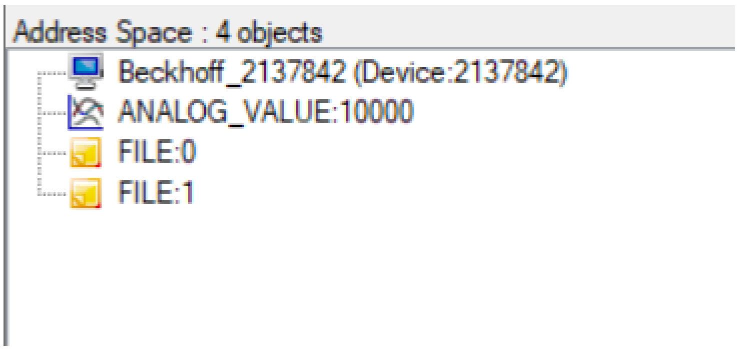
4. Select the device. This resolves the Device Object Name and starts the evaluation of the BACnet objects contained in the device.



Yet Another Bacnet Explorer - Yabe

Devices
- Devices
  - Udp:47808
    - Beckhoff_2137842 [2137842]

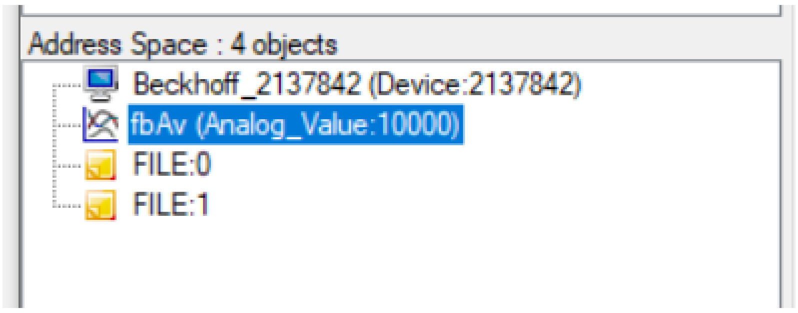
⇨ In the device window the tree items represent the objects contained in the device.



Address Space : 4 objects
- Beckhoff_2137842 (Device:2137842)
- ANALOG_VALUE:10000
- FILE:0
- FILE:1

5. Select *ANALOG_VALUE:10000*.

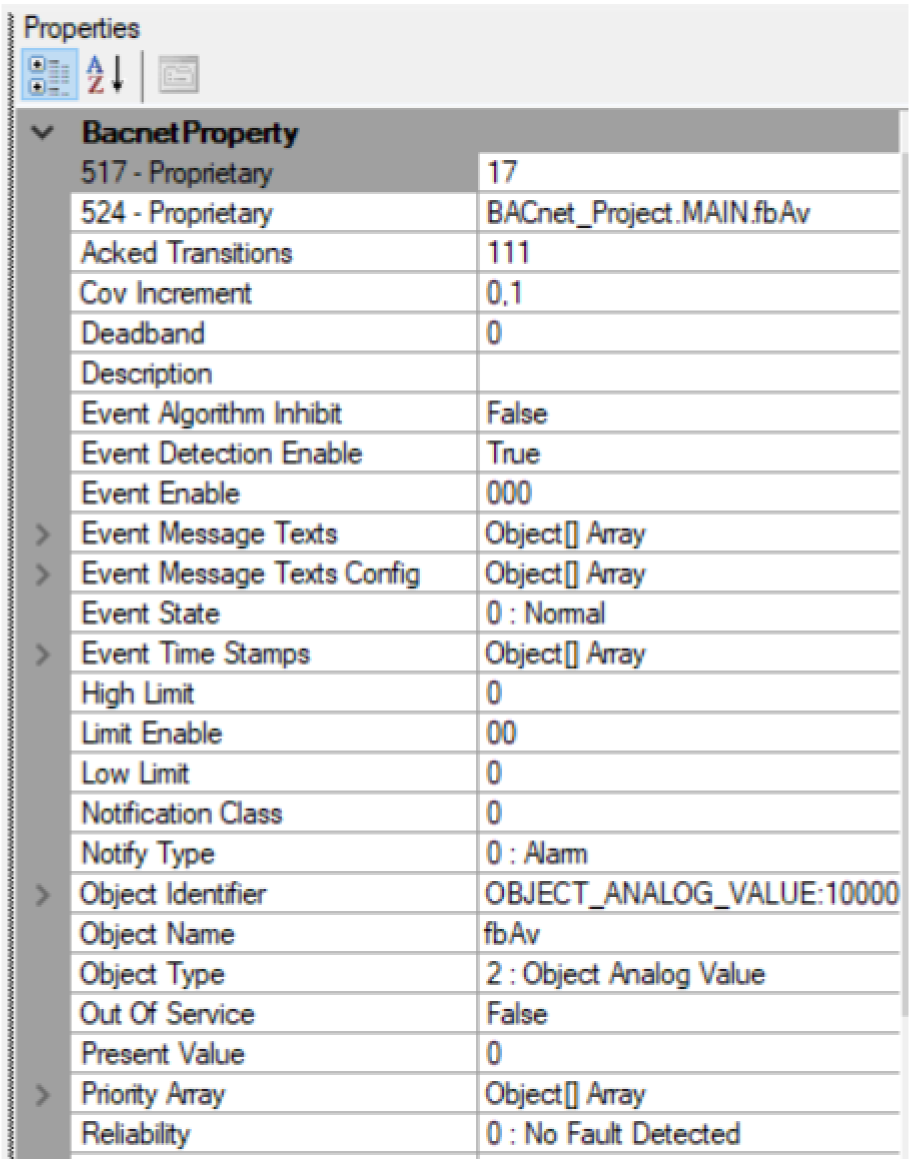
⇨ This starts the resolution of the object name (*fbAv*).


Address Space : 4 objects

- Beckhoff_2137842 (Device:2137842)
- fbAv (Analog_Value:10000)
- FILE:0
- FILE:1

Using CTRL-N after selecting a device resolves all object names of all objects in the device. This may take some time, especially if slower MS/TP devices are searched (MS/TP = Master Slave Token Passing = serial BACnet based on RS485).

⇨ Selecting an object in the list of objects opens the object properties display in the right *Properties* window.


Properties

Bacnet Property

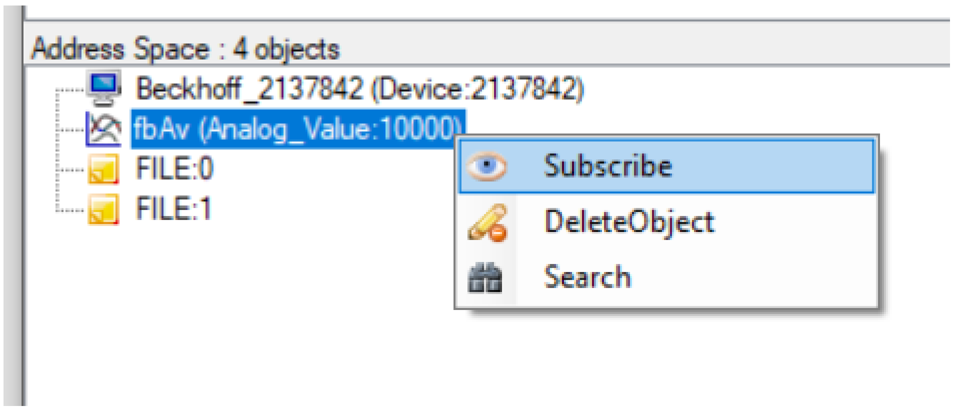
517 - Proprietary	17
524 - Proprietary	BACnet_Project.MAIN.fbAv
Acked Transitions	111
Cov Increment	0,1
Deadband	0
Description	
Event Algorithm Inhibit	False
Event Detection Enable	True
Event Enable	000
Event Message Texts	Object[] Array
Event Message Texts Config	Object[] Array
Event State	0 : Normal
Event Time Stamps	Object[] Array
High Limit	0
Limit Enable	00
Low Limit	0
Notification Class	0
Notify Type	0 : Alarm
Object Identifier	OBJECT_ANALOG_VALUE:10000
Object Name	fbAv
Object Type	2 : Object Analog Value
Out Of Service	False
Present Value	0
Priority Array	Object[] Array
Reliability	0 : No Fault Detected

This window does not update the property values automatically! To read the properties again, select another object and return to the observed object.

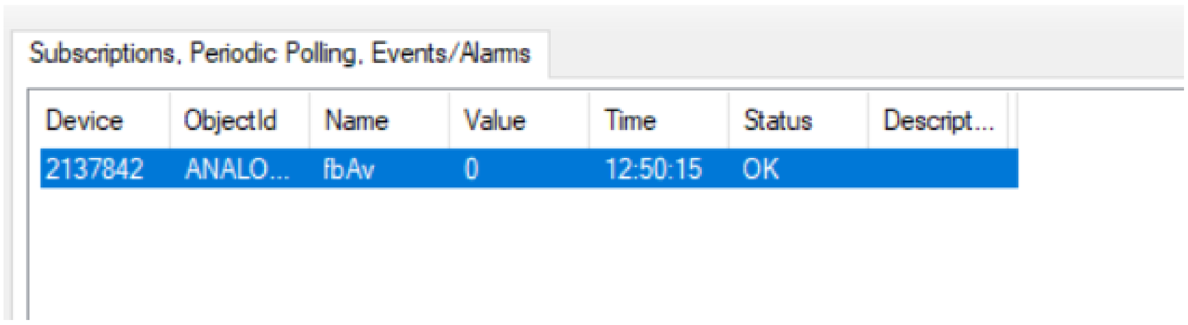
Subscribe for Change of Value (COV)

COV is a BACnet procedure to automatically report changes in Present Value and Status Flags to BACnet clients subscribed to the object.

1. To use COV in YABE, right-click on the object and select *Subscribe*. Alternatively, you can drag and drop the object into the *Subscriptions* window.

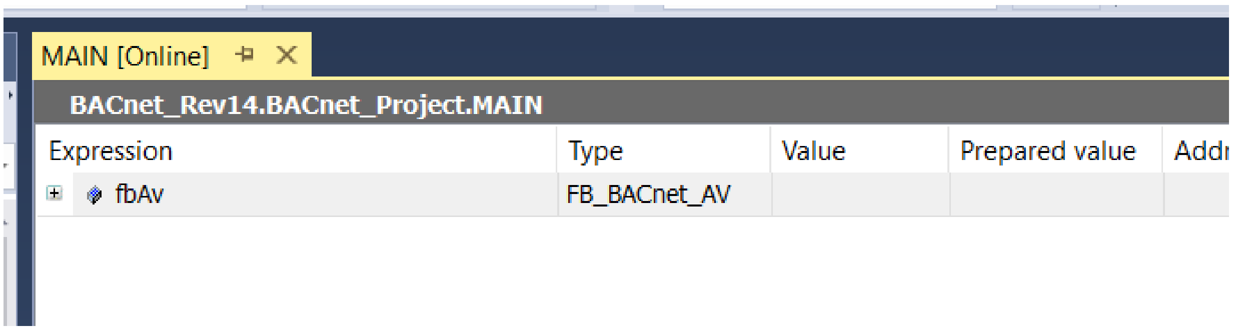

Address Space : 4 objects
Beckhoff_2137842 (Device:2137842)
fbAv (Analog_Value:10000)
FILE:0
FILE:1
Subscribe
DeleteObject
Search

⇨ The selected object is now displayed in the *Subscriptions* window and is automatically updated when the Present Value and/or the Status Flags of the object change. For analog object types, the COV Increment property is used to suppress minor value changes. The subscription lifetime can be adjusted in the Settings dialog (Subscriptions Lifetime).


Subscriptions, Periodic Polling, Events/Alarms

Device	ObjectId	Name	Value	Time	Status	Descript...
2137842	ANALO...	fbAv	0	12:50:15	OK	

2. Return to TwinCAT, log in to the PLC and open the POU MAIN.
3. Expand the display of FB *fbAv* by clicking on the plus symbol.

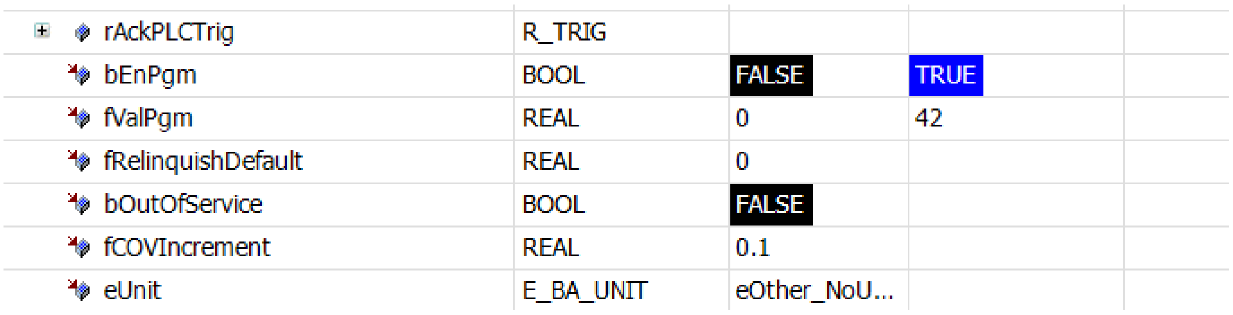

MAIN [Online]

BACnet_Rev14.BACnet_Project.MAIN

Expression	Type	Value	Prepared value	Addr
fbAv	FB_BACnet_AV			

4. Navigate to the variables *bEnPgm* and *fValPgm*.

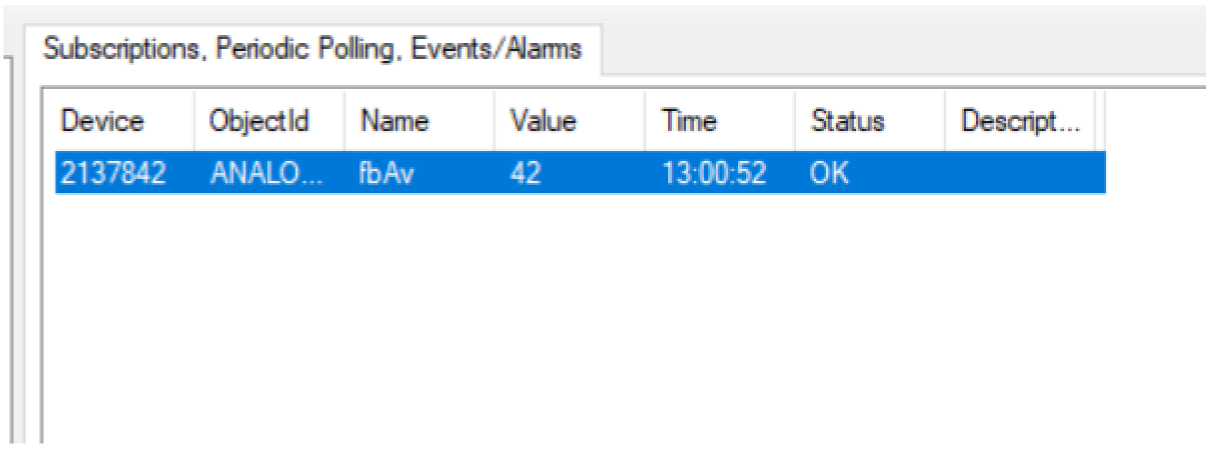
5. Insert the following values in the *Prepared value* column. If bEnPgm is set to TRUE, the value fValPgm is set to the Present Value with the specified priority for the program (default: 15).



rAckPLCTrig	R_TRIG		
bEnPgm	BOOL	FALSE	TRUE
fValPgm	REAL	0	42
fRelinquishDefault	REAL	0	
bOutOfService	BOOL	FALSE	
fCOVIncrement	REAL	0.1	
eUnit	E_BA_UNIT	eOther_NoU...	

6. Select *Write Values* from the PLC menu.
7. Switch to the online view in the TwinCAT System Manager or switch back to YABE.

⇨ The Present Value property should now display the value "42".



Device	ObjectId	Name	Value	Time	Status	Descript...
2137842	ANALO...	fbAv	42	13:00:52	OK	

5 PLC library: Tc3_BACnetRev14

This library provides the following items:

DUTs = **D**ata **U**nit **T**ypes

GVLs = **Gl**obal **V**ariable **L**ists

POUs = **P**rogram **O**rganizational **U**nits

Version = Global Version

5.1 DUTs

The datatypes are separated into three major items:

5.1.1 Enumerations

This section contains the required BACnet list. One exception is the list of enumerated technical units (E_BA_Unit). This can be found in the Tc3_BA2_Common library (as this is often shared between BACnet and the building automation libraries).

5.1.2 Interfaces

This section provides al list of interfaces, most of them used in the Base Object Types.

5.1.3 Types

This section provides a list of specific data-types, like ObjectIdentifier or PropertyList.

5.2 GVLs

The global variable lists are split into two parts:

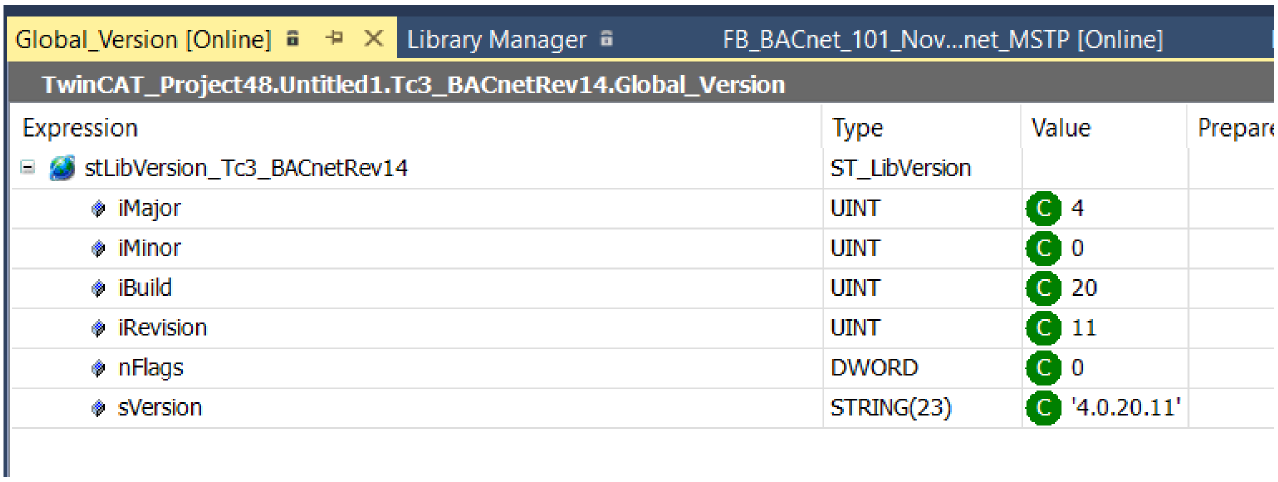
5.2.1 Version

The variable stLibVersion_Tc3_BACnetRev14 of type ST_LibVersion can be used to compare the version of the library with the version stored as project information.

The TwinCAT3 library version consists of the following elements:

iMajor = Major release number
iMinor = Minor release number
iBuild = Build number
iRevision = Revision number

In addition, this structure contains the version information in the form of a string separated by dots.

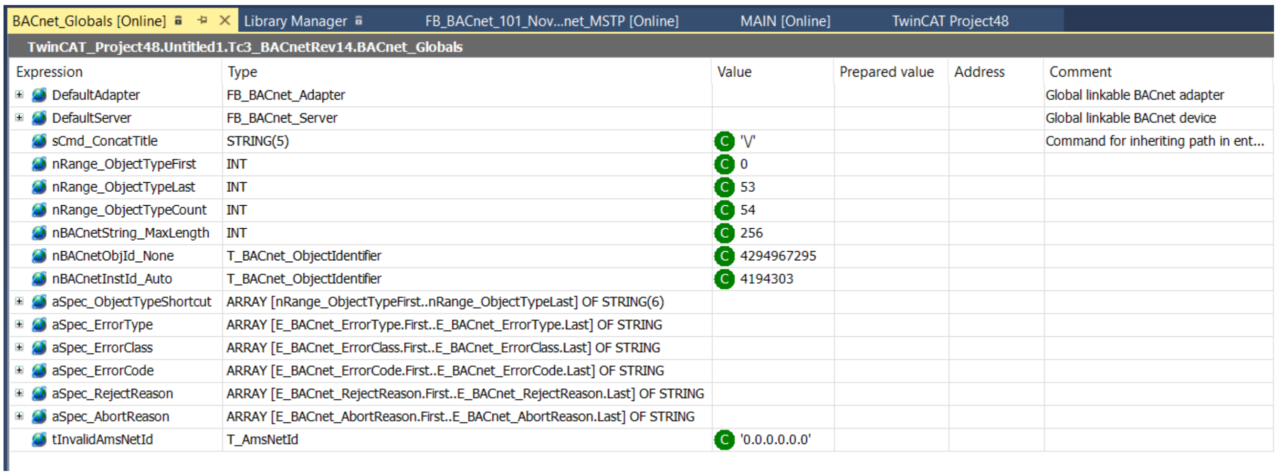

Global_Version [Online] | Library Manager | FB_BACnet_101_Nov...net_MSTP [Online]

TwinCAT_Project48.Untitled1.Tc3_BACnetRev14.Global_Version

Expression	Type	Value	Prepare
stLibVersion_Tc3_BACnetRev14	ST_LibVersion		
iMajor	UINT	4	
iMinor	UINT	0	
iBuild	UINT	20	
iRevision	UINT	11	
nFlags	DWORD	0	
sVersion	STRING(23)	'4.0.20.11'	

5.2.2 BACnet_Globals

This part of the library specifies global settings like the Default Adapter, BACnet-specific values like supported object types and Error-, Abort- and Reject-Codes.


BACnet_Globals [Online] | Library Manager | FB_BACnet_101_Nov...net_MSTP [Online] | MAIN [Online] | TwinCAT Project48

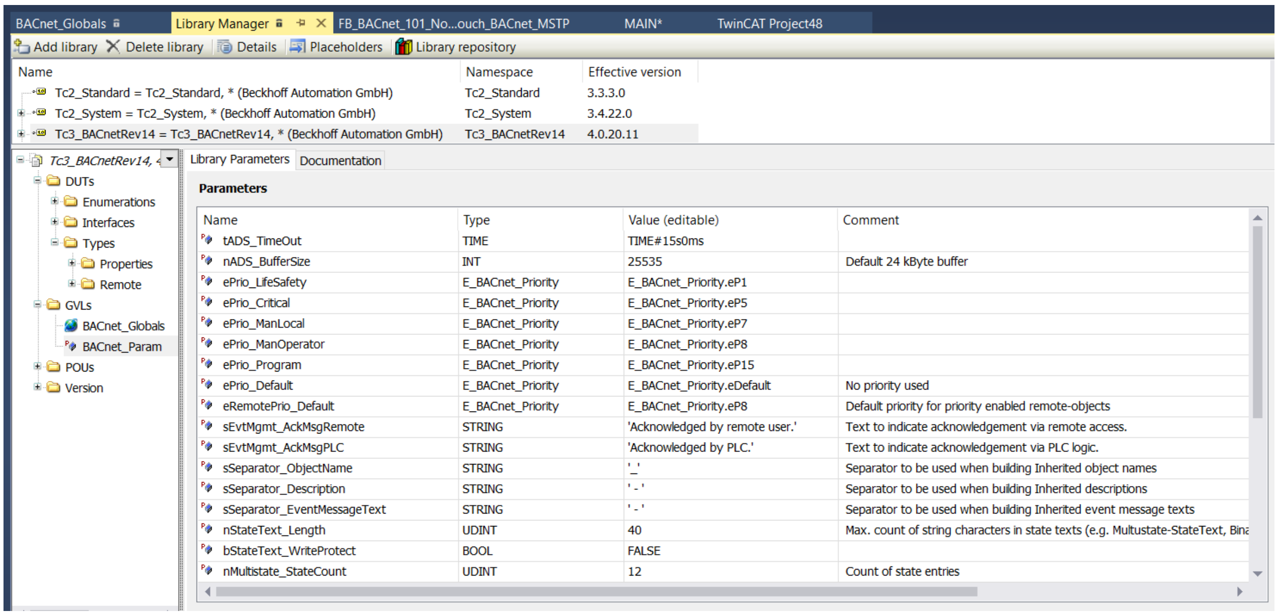
TwinCAT_Project48.Untitled1.Tc3_BACnetRev14.BACnet_Globals

Expression	Type	Value	Prepared value	Address	Comment
DefaultAdapter	FB_BACnet_Adapter				Global linkable BACnet adapter
DefaultServer	FB_BACnet_Server				Global linkable BACnet device
sCmd_ConcatTitle	STRING(5)	'V'			Command for inheriting path in ent...
nRange_ObjectTypeFirst	INT	0			
nRange_ObjectTypeLast	INT	53			
nRange_ObjectTypeCount	INT	54			
nBACnetString_MaxLength	INT	256			
nBACnetObjId_None	T_BACnet_ObjectIdentifier	4294967295			
nBACnetInstId_Auto	T_BACnet_ObjectIdentifier	4194303			
aSpec_ObjectTypeShortcut	ARRAY [nRange_ObjectTypeFirst..nRange_ObjectTypeLast] OF STRING(6)				
aSpec_ErrorType	ARRAY [E_BACnet_ErrorType.First..E_BACnet_ErrorType.Last] OF STRING				
aSpec_ErrorClass	ARRAY [E_BACnet_ErrorClass.First..E_BACnet_ErrorClass.Last] OF STRING				
aSpec_ErrorCode	ARRAY [E_BACnet_ErrorCode.First..E_BACnet_ErrorCode.Last] OF STRING				
aSpec_RejectReason	ARRAY [E_BACnet_RejectReason.First..E_BACnet_RejectReason.Last] OF STRING				
aSpec_AbortReason	ARRAY [E_BACnet_AbortReason.First..E_BACnet_AbortReason.Last] OF STRING				
tInvalidAmsNetId	T_AmsNetId	'0.0.0.0.0.0'			

5.2.3 BACnet_Param

Global settings and parameters for the library instance in the project can be defined and changed in this part of the library. Variables declared as `CONSTANT` are called parameters in TwinCAT and can be changed and adapted according to the requirements of the specific project.

Please note that the default parameters from the library repository remain unchanged. To restore the default values, simply remove the BACnet Rev14 library and add it back to the project.



Name	Namespace	Effective version
Tc2_Standard = Tc2_Standard, * (Beckhoff Automation GmbH)	Tc2_Standard	3.3.3.0
Tc2_System = Tc2_System, * (Beckhoff Automation GmbH)	Tc2_System	3.4.22.0
Tc3_BACnetRev14 = Tc3_BACnetRev14, * (Beckhoff Automation GmbH)	Tc3_BACnetRev14	4.0.20.11

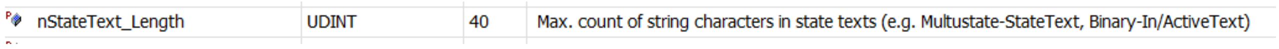
Parameters

Name	Type	Value (editable)	Comment
tADS_TimeOut	TIME	TIME#15s0ms	
nADS_BufferSize	INT	25535	Default 24 kByte buffer
ePrio_LifeSafety	E_BACnet_Priority	E_BACnet_Priority.eP1	
ePrio_Critical	E_BACnet_Priority	E_BACnet_Priority.eP5	
ePrio_ManLocal	E_BACnet_Priority	E_BACnet_Priority.eP7	
ePrio_ManOperator	E_BACnet_Priority	E_BACnet_Priority.eP8	
ePrio_Program	E_BACnet_Priority	E_BACnet_Priority.eP15	
ePrio_Default	E_BACnet_Priority	E_BACnet_Priority.eDefault	No priority used
eRemotePrio_Default	E_BACnet_Priority	E_BACnet_Priority.eP8	Default priority for priority enabled remote-objects
sEvtMgmt_AckMsgRemote	STRING	'Acknowledged by remote user.'	Text to indicate acknowledgement via remote access.
sEvtMgmt_AckMsgPLC	STRING	'Acknowledged by PLC.'	Text to indicate acknowledgement via PLC logic.
sSeparator_ObjectName	STRING	'_'	Separator to be used when building Inherited object names
sSeparator_Description	STRING	' - '	Separator to be used when building Inherited descriptions
sSeparator_EventMessageText	STRING	' - '	Separator to be used when building Inherited event message texts
nStateText_Length	UDINT	40	Max. count of string characters in state texts (e.g. Multustate-StateText, Bina
bStateText_WriteProtect	BOOL	FALSE	
nMultistate_StateCount	UDINT	12	Count of state entries

The default values provided by the library can be changed using the *Value (Editable)* column.

Example: Extending the text length of state text properties.

The following figure shows a default text length of 40 characters for text information provided in multistate and binary objects.



nStateText_Length	UDINT	40	Max. count of string characters in state texts (e.g. Multustate-StateText, Binary-In/ActiveText)

For larger text information, the value can be changed from 40 to 60 characters by editing the value.

Important: These parameters (like other variable initializations in TwinCAT) are only applied after activating the project or after a reset origin and restart of the PLC.

5.3 POUs

The function blocks in the POUs library section are intended for programming BACnet servers and BACnet clients.

The POUs section is divided into these sections:

Dynamic Objects: The Dynamic Object Manager is mainly used for applications that create or delete objects based on a configuration file or at runtime (e.g. a visualization).

Helper: This section contains helper functions such as conversion functions, functions for generating complex data types, etc.

Local: This section contains function blocks for local / server applications. This means that TwinCAT provides BACnet objects as a server, which represent values from the PLC.

Remote: This section contains function blocks for remote/client applications. This means that TwinCAT accesses other devices as a client and processes values from other devices in the PLC.

FB_BACnet_Adapter: This function block represents a BACnet adapter in the System Manager. Please note that the Default Adapter is already present in the GVL BACnet_Globals and is automatically called as soon as a BACnet function block is called in the PLC code.

5.3.1 Naming conventions

Local / server object function block names start with the prefix “FB_BACnet_” followed by the shortcut of the object type (e. g. ACC = Accumulator, AI = Analog Input, etc.).

Remote / client object function block names start with the prefix “FB_BACnetRM_” followed by the shortcut of the object type.

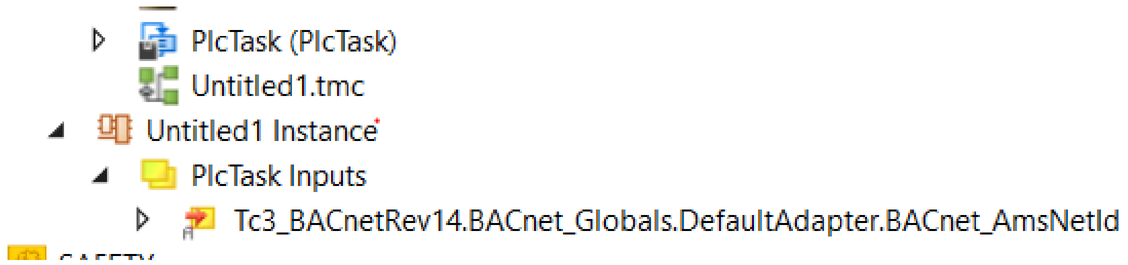
5.3.2 FB_BACnet_Adapter

The function block FB_BACnet_Adapter represents one BACnet device configured under I/O devices in the TwinCAT system manager. To avoid confusion with BACnet’s device object, a BACnet device will be called BACnet adapter in the context of this library. A BACnet adapter is connected to a network adapter if BACnet/ IP is used and in case of MS/TP one BACnet adapter is connected to an EL6861 terminal.

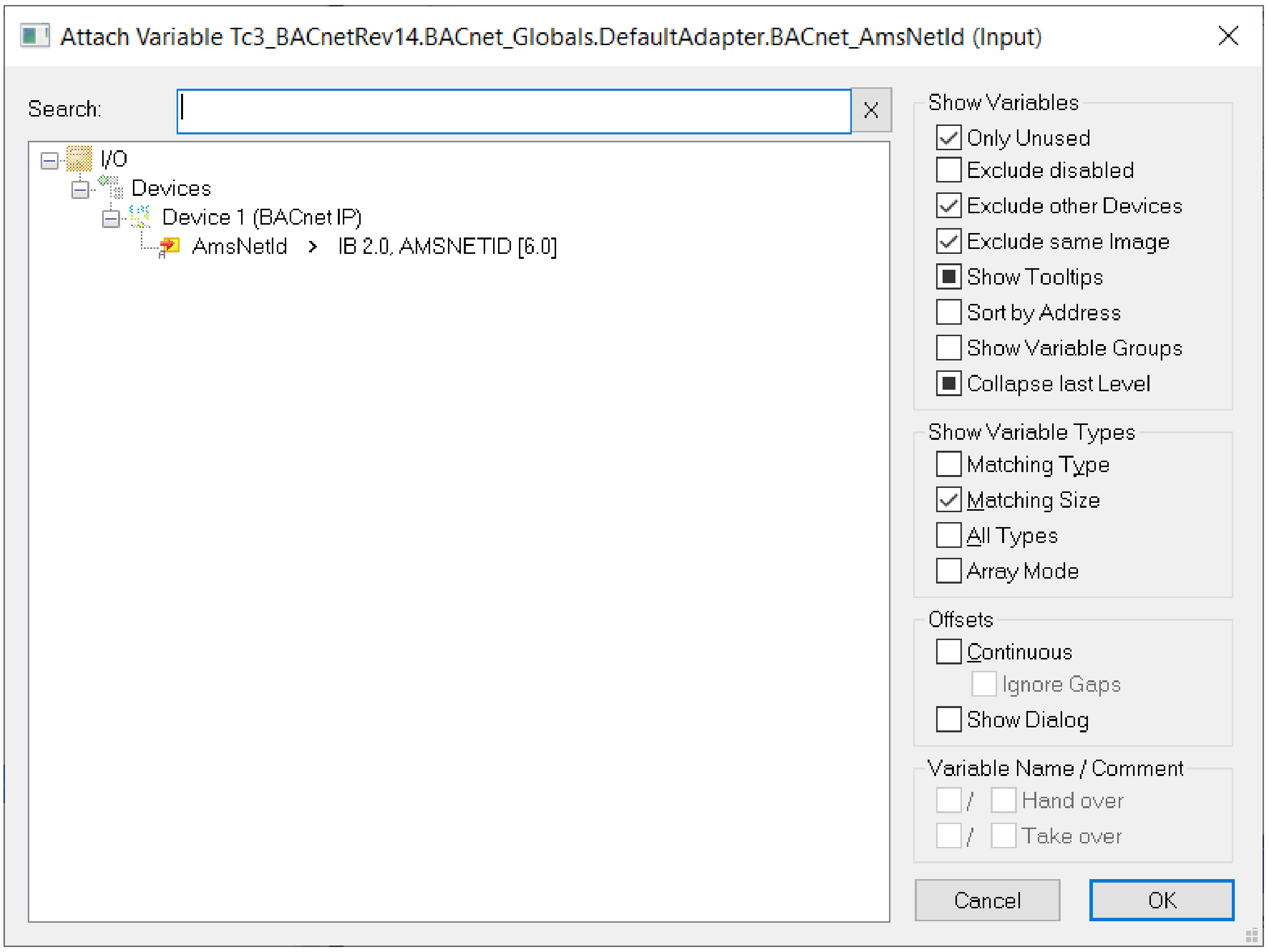
A connection between an instance of FB_BACnet_Adapter with its according I/O device can be established by connecting its AmsNetId variables.

5.3.2.1 Default Adapter

By adding the library *Tc3_BACnetRev14* a default adapter is automatically generated. This adapter is configured as global variable within the library.


PlcTask (PlcTask)
Untitled1.tmc
Untitled1 Instance
PlcTask Inputs
Tc3_BACnetRev14.BACnet_Globals.DefaultAdapter.BACnet_AmsNetId

This adapter needs to be connected to the device (e. g. BACnet/IP).


Attach Variable Tc3_BACnetRev14.BACnet_Globals.DefaultAdapter.BACnet_AmsNetId (Input)

Search:

I/O
Devices
Device 1 (BACnet IP)
AmsNetId > IB 2.0, AMSNETID [6.0]

Show Variables
- [✓] Only Unused
- [ ] Exclude disabled
- [✓] Exclude other Devices
- [✓] Exclude same Image
- [■] Show Tooltips
- [ ] Sort by Address
- [ ] Show Variable Groups
- [■] Collapse last Level

Show Variable Types
- [ ] Matching Type
- [✓] Matching Size
- [ ] All Types
- [ ] Array Mode

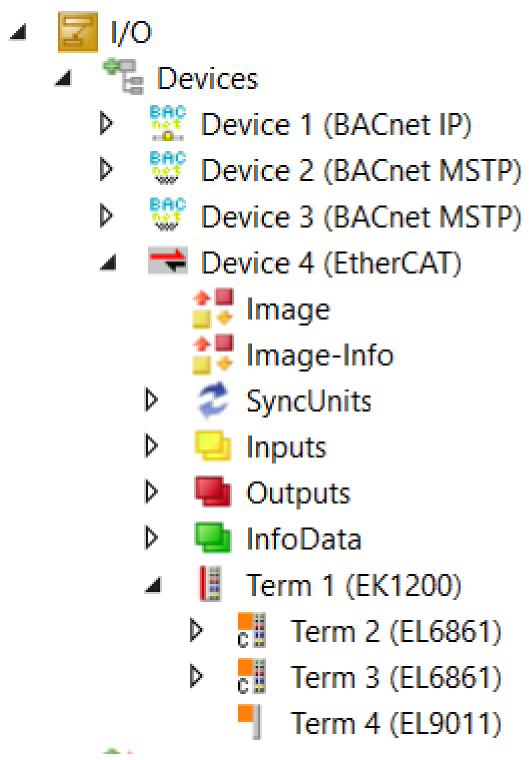
Offsets
- [ ] Continuous
- [ ] Ignore Gaps
- [ ] Show Dialog

Variable Name / Comment
- [ ] / [ ] Hand over
- [ ] / [ ] Take over

5.3.2.2 Using multiple BACnet adapters

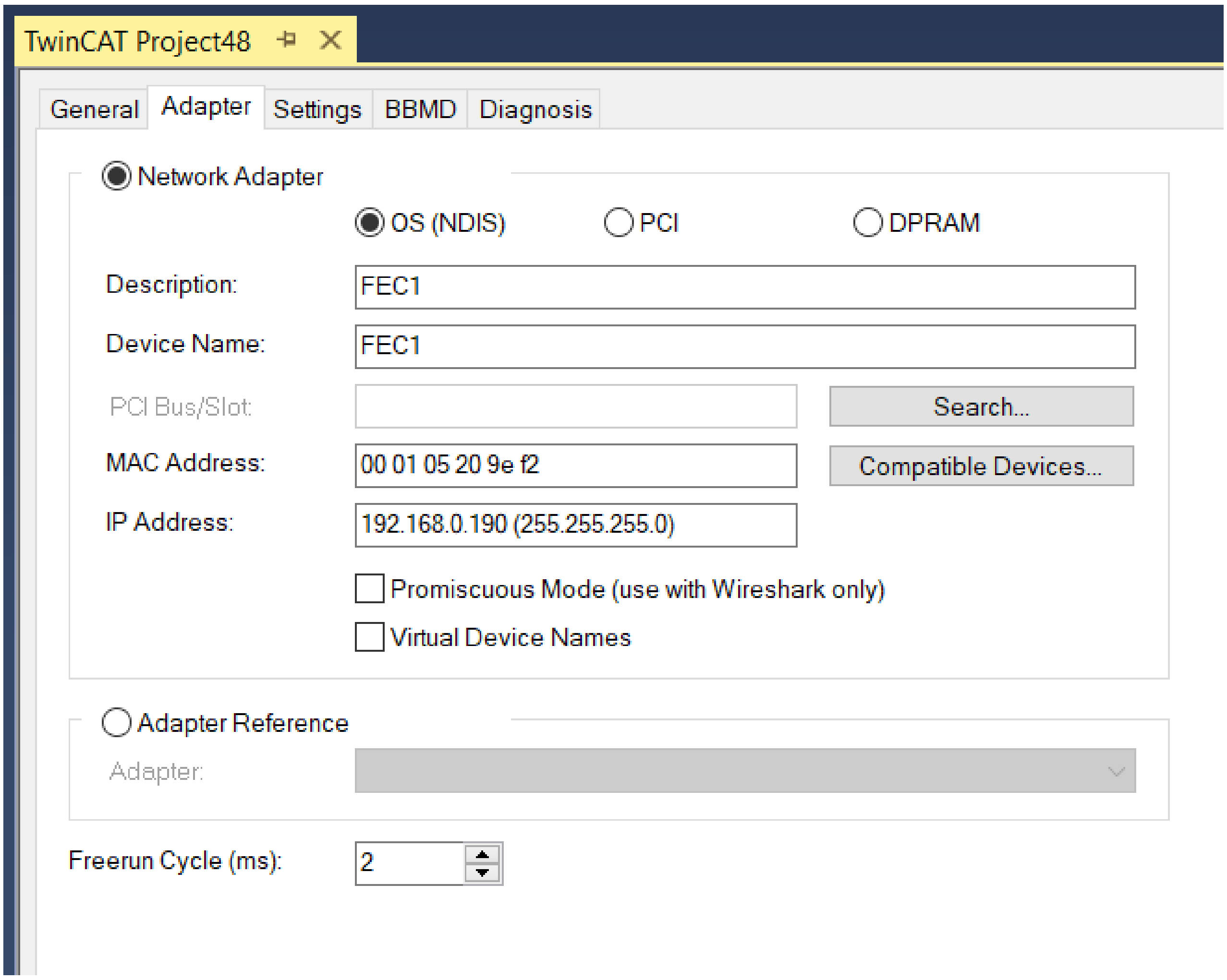
The following example illustrates how to use multiple BACnet adapters connecting to different devices.

The example below uses a total of three adapters, 1 x BACnet/IP and 2 x MS/TP using terminals EL6861. The next picture shows the overview in the TwinCAT system manager.



```
I/O
  Devices
    Device 1 (BACnet IP)
    Device 2 (BACnet MSTP)
    Device 3 (BACnet MSTP)
    Device 4 (EtherCAT)
      Image
      Image-Info
      SyncUnits
      Inputs
      Outputs
      InfoData
      Term 1 (EK1200)
        Term 2 (EL6861)
        Term 3 (EL6861)
        Term 4 (EL9011)
```

The device BACnet/IP is connected to the network adapter FEC1 of the CX9020.


TwinCAT Project48

General | Adapter | Settings | BBMD | Diagnosis

[● Network Adapter]

[● OS (NDIS)] PCI DPRAM

Description:	FEC1
Device Name:	FEC1
PCI Bus/Slot:	
MAC Address:	00 01 05 20 9e f2
IP Address:	192.168.0.190 (255.255.255.0)

☐ Promiscuous Mode (use with Wireshark only)

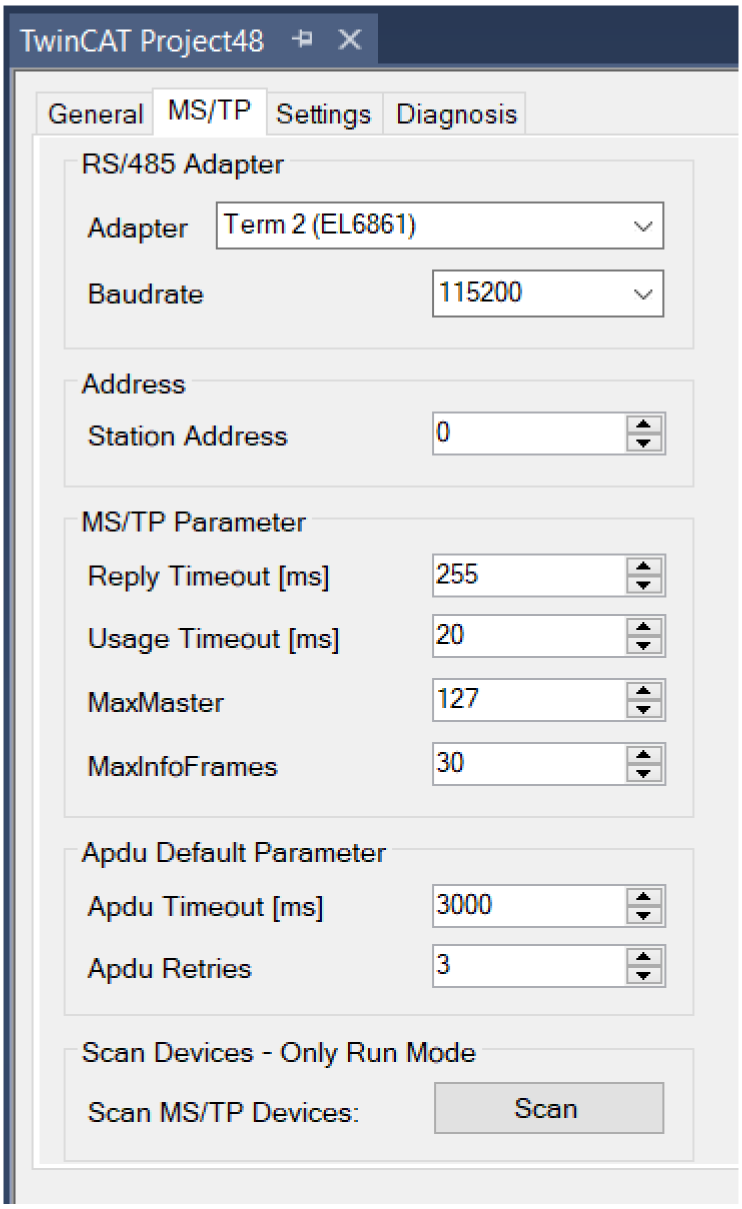
☐ Virtual Device Names

Adapter Reference

Adapter:

Freerun Cycle (ms): 2

The first MS/TP device *Device 2 (BACnet MSTP)* is connected to the terminal named *Term 2 (EL6861)*.



RS/485 Adapter

Adapter	Term 2 (EL6861)
Baudrate	115200

Address

Station Address	0

MS/TP Parameter

Reply Timeout [ms]	255
Usage Timeout [ms]	20
MaxMaster	127
MaxInfoFrames	30

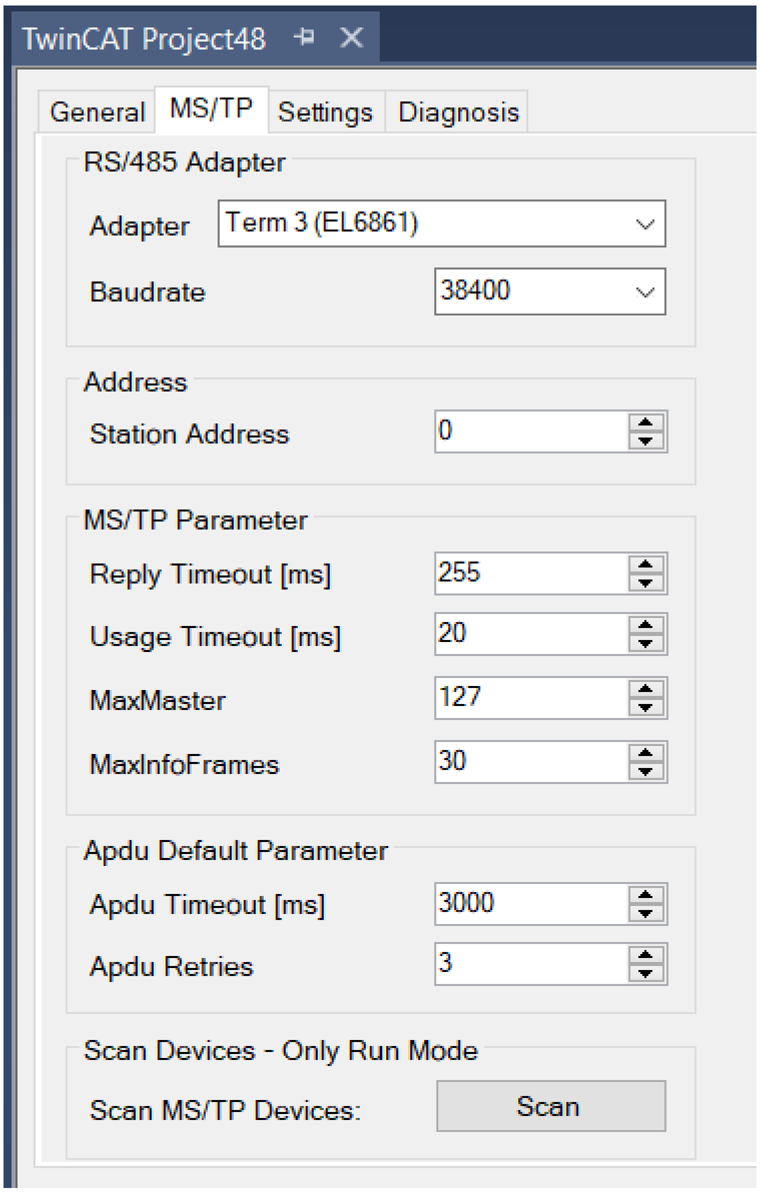
Apdu Default Parameter

Apdu Timeout [ms]	3000
Apdu Retries	3

Scan Devices - Only Run Mode

Scan MS/TP Devices:

The second MS/TP device *Device 3 (BACnet MSTP)* is connected to the terminal named *Term 3 (EL6861)*.



RS/485 Adapter

Adapter	Term 3 (EL6861)
Baudrate	38400

Address

Station Address	0

MS/TP Parameter

Reply Timeout [ms]	255
Usage Timeout [ms]	20
MaxMaster	127
MaxInfoFrames	30

Apdu Default Parameter

Apdu Timeout [ms]	3000
Apdu Retries	3

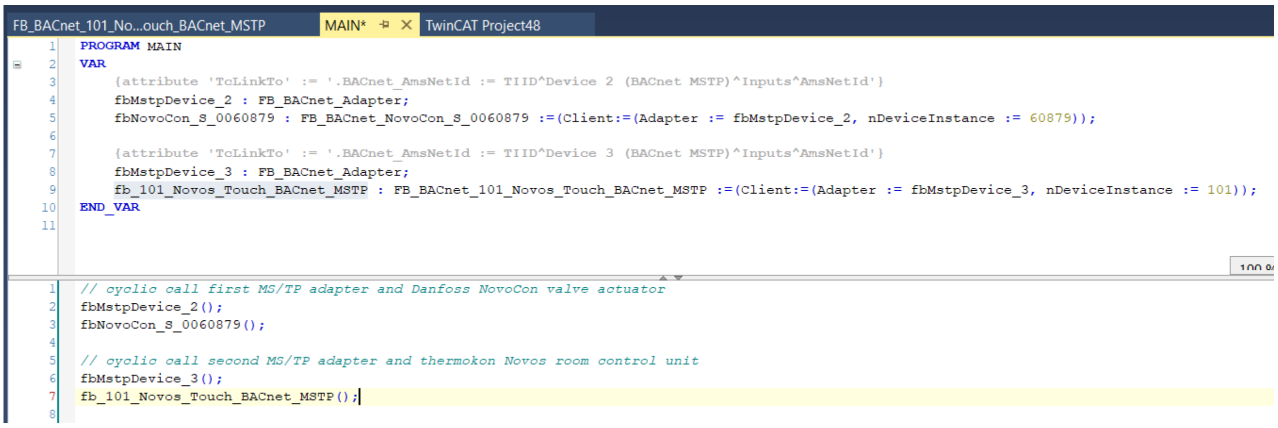
Scan Devices - Only Run Mode

Scan MS/TP Devices: Scan

All three devices are now connected to the PLC.

In this example devices connected to the network have already been scanned and are available as BACnet client references. Using the function *FB Code* (see separate chapter for details) code was generated to access the objects through function blocks from the BACnet library.

The next window shows the variables and code in the POU MAIN copied from the created client function blocks.



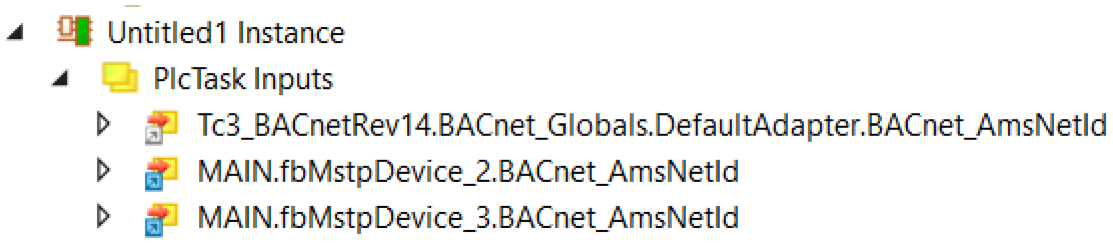
```
PROGRAM MAIN
VAR
    {attribute 'TcLinkTo' := '.BACnet_AmsNetId := TIID^Device 2 (BACnet MSTP)^Inputs^AmsNetId'}
    fbMstpDevice_2 : FB_BACnet_Adapter;
    fbNovoCon_S_0060879 : FB_BACnet_NovoCon_S_0060879 :=(Client:=(Adapter := fbMstpDevice_2, nDeviceInstance := 60879));

    {attribute 'TcLinkTo' := '.BACnet_AmsNetId := TIID^Device 3 (BACnet MSTP)^Inputs^AmsNetId'}
    fbMstpDevice_3 : FB_BACnet_Adapter;
    fb_101_Novos_Touch_BACnet_MSTP : FB_BACnet_101_Novos_Touch_BACnet_MSTP :=(Client:=(Adapter := fbMstpDevice_3, nDeviceInstance := 101));
END_VAR
```

```
// cyclic call first MS/TP adapter and Danfoss NovoCon valve actuator
fbMstpDevice_2();
fbNovoCon_S_0060879();

// cyclic call second MS/TP adapter and thermokon Novos room control unit
fbMstpDevice_3();
fb_101_Novos_Touch_BACnet_MSTP();
```

The code *{attribute ‚TcLinkTo‘* := … generates a dynamic connection to the referenced BACnet device. The dynamic connection symbol is marked in green in the system manager.

The default adapter is connected using a manual connection.

The three connections look like this:



```
Untitled1 Instance
    PlcTask Inputs
        Tc3_BACnetRev14.BACnet_Globals.DefaultAdapter.BACnet_AmsNetId
        MAIN.fbMstpDevice_2.BACnet_AmsNetId
        MAIN.fbMstpDevice_3.BACnet_AmsNetId
```

6 Programming a BACnet server

A BACnet server is a device that provides objects and services for other devices, e.g. an MBE (management operating device). Each BACnet server requires a device object. The ObjectList and StructuredObjectList properties of the Device object provide access to the server "database" (the objects contained in the device).

The function blocks that represent BACnet objects in a server are located in the folder POUs/Local/Objects.

In the following, the function blocks for BACnet objects are presented, followed by typical use cases, how and which function blocks are used.

6.1 BACnet object POUs

This chapter describes the function blocks used to program a BACnet server.

6.1.1 Function blocks without a suffix

ACC	Accumulator	This object type represents accumulated (pulse) values.
AI	Analog Input	This object type represents physical analog input information, e.g. a sensor value.
AO	Analog Output	This object type represents physical analog output information, e.g. via a 0-10V output.
AV	Analog Value	This object type represents a (virtual) analog value information, e.g. a setpoint.
BI	Binary Input	This object type represents a binary input information, e.g. the state of a lamp or a fuse.
BO	Binary Output	This object type represents a binary output information, e.g. via a switching output.
BV	Binary Value	This object type represents a (virtual) binary value information, e.g. an error state.
CAL	Calendar	This object type represents calendar (date-based) information.
Device	Device	This object type represents the physical device. It contains information such as the local clock, vendor, model name and more.
EE	EventEnrollment	This object type is used to apply event monitoring in addition to intrinsic reporting, e.g. to implement warning limits.
ELOG	Eventlog	This object type represents an event log buffer, e.g. to store alarms locally.
File	File	This object type represents files, e.g. the current configuration or persistent data.
Loop	Control Loop	This object type represents control loops, e.g. a PI or PID loop.
MI	Multistate Input	This object type represents a physical multistate input information, e.g. a local operating modes switch.
MO	Multistate Output	This object type represents a physical multistate output information, e.g. an operating modes switch controlled by the PLC.
MV	Multistate Value	This object type represents a (virtual) multistate value information, e.g. a program parameter.
NC	Notification Class	This object type represents an alarm class to notify recipients.
PC	Pulse Converter	This object type represents converted pulse information, e.g. energy consumption in kWh.
Prog	Program	This object type represents the PLC program.
SchedA	Schedule Analog	This object type represents a schedule with analog values.
SchedB	Schedule Binary	This object type represents a list of binary values.
SchedM	Schedule Multistate	This object type represents a schedule with multistate values.
TLM	Trendlog Multiple	This object type represents a trendlog object that supports multiple channels.
Tlog	Trendlog	This object type represents a trendlog object that supports a single channel.
View	Structured View	This object type represents a user-oriented object hierarchy.

6.1.2 Primitive Value object types

Date	Single Date Value	This object type represents a certain single date information (day, month, year-1900, weekday).
DateP	Date Pattern Value	This object type represents a date pattern. The pattern value 255 can be used as a wildcard.
DateTime	Date and Time Value	This object type represents a combination of a specific date and time.
DateTimeP	Date and Time Pattern Value	This object type represents a combination of date and time patterns.
INT	Signed Integer Value	This object type represents a signed integer value.
LAV	Large Analog Value (LREAL)	This object type represents a large analog value (8 BYTE LREAL).
String	Character String Value	This object type represents string information.
Time	Time Value	This object type represents a specific time (hour, minute, second, hundredths of a second).
TimeP	Time Pattern Value	This object type represents a time value that supports patterns.
UINT	Unsigned Integer Value	This object type represents a positive integer value (UNSIGNED).

Some function block names contain a suffix:

_IO: These FBs are intended for connection to hardware terminal channels. The required AT%I* and AT%Q* variables are specified in the FB implementation.
E.g. a BinaryOutput_IO provides these variables to connect to terminal channels:
bRawOvrrd AT %I* : BOOL; // Raw overridden (Optional)
bRawValFdbk AT %I* : BOOL; // Raw feedback Value (Optional)
bRawVal AT %Q* : BOOL; // Raw value

_ECAT: These FBs are intended for connection to EtherCAT hardware terminal channels. The difference to the FBs with the suffix _IO is the variable nRawState, which is used to determine the Underrange/Overrange state.

_Raw: If the PLC provides the value for the BACnet object, the raw value and the raw state can be provided by the PLC program.
Example: FB_BACnet_AI_Raw:
nRawState : USINT; // Raw state
// - Underrange: 0x01
// - Overrange: 0x02
// - Error: 0x04
nRawVal : INT; // Raw value

_Disp: These function blocks refer to value object types that represent read-only values such as the current room temperature. Present_Value is not writable.

_Event: These function blocks refer to a value object similar to the type _Disp (read-only). In addition, these function blocks support Event Reporting. Present_Value is not writable.

_Setp: These function blocks refer to setpoints. Setpoints are writable BACnet objects without command prioritization ("last writer wins"). Present_Value is writable, but not commandable.

_Buf: These function blocks implement a log buffer in the PLC. This can be used for local visualizations of trend log or event log information.

_5P: These function blocks refer to commandable output or value objects that provide a set of 5 priorities for command prioritization. The priority level of each of the 5 priorities can be set in the global BACnet_Param settings.

Default settings:

Priority category	Default value
Life-Safety	1
Critical Equipment Control	5
Minimum on/off times	6 (defined by the BACnet standard, cannot be changed)
Manual Local Operator	7 (local visualization)
Manual operator	8 (BMS)
Program (PLC)	15

_IO5P: Same as _5P, but used for connection to hardware terminals.

_Raw5P: Same as _5P, but the process value is provided by the PLC.

_Ref: (Loop object type) These function blocks refer to loop objects that support external analog objects for the setpoint, manipulated variable and feedback value. See chapter Control loops.

6.2 Typical BACnet scenarios

This chapter introduces typical scenarios which are commonly used for BACnet projects.

6.2.1 Command prioritization

Command prioritization provides a mechanism to determine which process or user role has precedence over lower prioritized processes or user-roles. BACnet specifies 16 levels of priorities, where 1 is specified as the highest (Manual Life-Safety) Priority where 16 specifies the lowest and default priority.

The Present_Value is always taken from the value in the highest prioritized slot. A client may remove a priority at a specific slot by writing a value of NULL at the given priority.

If no priority is active (all 16 slots having a value NULL), the value for the Present_Value property is taken from the Relinquish_Default property.

The priority is specified using a WriteProperty or WritePropertyMultiple service. If the priority is missing, 16 will be used as the default priority.

Priority	Meaning
1	Manual Life-Safety
2	Automatic Life-Safety
3	Available
4	Available
5	Critical Equipment Control, e. g. defrost
6	Minimum On-/Off
7	Available
8	Manual Operator, e. g. BMS
9	Available
10	Available
11	Available
12	Available
13	Available
14	Available
15	Available
16	Available (Default, if no priority is specified)

Function blocks with the suffix “_5P” implement a total of 5 of the 16 possible priorities (which is normally enough for most projects).

6.2.2 Event reporting

BACnet provides two different procedures to implement event reporting:

Intrinsic Reporting and Algorithmic Change Reporting.

The determination whether an event is a non-critical information, e. g. an operational message or an abnormal state, e. g. an alarm is specified by the Notify_Type property.

6.2.2.1 Intrinsic Reporting

Intrinsic reporting observes the Present_Value property, the event parameters are specified within the same object (e. g. High_Limit, TimeDelay, Alarm_Value, etc.).

Event generating objects have a connection to a NotificationClass object within the same device (referenced by the same instance number in the event generating object and the NotificationClass object).

Events are being notified to recipients which are present in the Recipient_List property of the NotificationClass and which match the active days / active time in the subscription.

6.2.2.2 Algorithmic Change Reporting

Algorithmic Change Reporting uses the Event Enrollment object to specify the event algorithm and the event parameters. The Event Enrollment object has a logical connection to a Notification Class object like every other event-generating object.

A typical scenario in projects implements an Event Enrollment providing warning limits notified to a Warning Notification Class. Intrinsic Reporting may then be used to detect alarm conditions (High- and Low-Limit) being notified to another Notification Class for alarms.

6.2.3 Scheduling

BACnet time scheduling provides strategies to implement date- and/or time-based functionality.

For this, BACnet uses two object types, Calendar and Schedule.

The Calendar object type provides information if TODAY matches one or more entries in the Date_List property. This is used e. g. to determine, if TODAY is a public holiday or not (assuming the date-list contains this information). An entry in the Date_List property may be a single Date, a DateRange or a WeekNDay (which is a combination of days, weeks and months).

The Schedule object type is mainly based upon two properties.

The Weekly_Schedule property represents a Schedule program (time/value list) for each of the days from Monday to Sunday.

The Exception_Schedule property specifies a list of exceptions which have precedence over the WeeklySchedule. The exceptions may be specified as a single Date, DateRange or WeekNDay or may be based upon a Calendar reference within the same device. Unfortunately, it is not possible from the BACnet standard to specify a global Calendar and refer to this object from other devices. If needed, the content of a global calendar may need to be copied to local calendar objects in every BACnet device.

Schedule objects contain a Schedule_Default property which specifies the fallback value at midnight, except the value is repeated at 00.00.00:00 the next day.

The schedule object has no specific data-type. The actual data-type is determined by the Weekly- and ExceptionSchedule, Schedule_Default and the list of ObjectPropertyReferences.

ObjectPropertyReferences may contain properties in objects within the same or in foreign devices.

The priority of the Schedule process is specified using the Priority_For_Writing property in the range from 1-16.

6.2.4 Trend-Logging

Trendlog objects are used to log values from a single source, Trendlog Multiple objects provide multiple channels (but are rarely used).

Logging can be based on cyclic polling, COV (Tlog only) or triggered data acquisition (Trigger).

In addition to the values of the referenced object property, other events are also recorded (e.g. activation/deactivation of recording, emptying the buffer, etc.).

The size of the log buffer is determined by the BUFFER_SIZE property.

Record_Count provides information about the current utilization of the log buffer. Writing a null value to Record_Count clears the buffer.

Trendlog objects can be combined with event creation. In this case, a Notification_Threshold can be specified to notify clients to retrieve the values before logging ends or older values are overwritten.

6.2.5 Event-Logging

Event logging provides strategies to locally store events or alarms e. g. in case the connection to the BMS is interrupted.

In a BACnet server implementation specifying event log objects require to use the same instance number as the Notification Class objects. In this case the Event log object is automatically assigned and retrieves all events/alarms which are distributed through this notification class object.

6.2.6 Control Loops

BACnet specifies a Loop object type to provide the parameters for control loops (e. g. P, I, D, Bias, Maximum_Output, etc.). These values are provided as properties of the Loop object type.

In case the setpoint, manipulated variable value and the controlled variable value should be visible for BACnet clients, analog objects may be specified in addition to the loop object. TwinCAT provides two implementations for control loops:

FB_BACnet_Loop: Loop object w/o external references, the setpoint and manipulated and control values are taken from internal variables.

FB_BACnet_Loop_Ref: Loop object with external references. A typical implementation may include an Analog Value object for the setpoint, Analog Output for the manipulated variable value and an Analog Input for the controlled variable value.

Example:

```
VAR
// control loop using internal variables
    fbLoopInternal : FB_BACnet_Loop := (
                       bEn          := TRUE,
                       sDescription := 'Loop using internal control parameters',
                       eOutputUnit  := E_BA_Unit.eOther_Percent,
                       eAction      := E_BA_Action.eReverse,
                       fProportionalConstant := 5.0,
                       fIntegralConstant := 180,
                       fSetpoint    := 20
);
                     fCtrlVal : REAL := 18;


// control loop using external BACnet objects
    fbLoopRef_Setpt : FB_BACnet_AV_Setp := (fValue := 20);
    fbLoopRef_CtrlVar : FB_BACnet_AI := (fVal := 18);
    fbLoopRef_Y : FB_BACnet_AO := ();
    fbLoopRef : FB_BACnet_Loop_Ref := (
        bEn := TRUE,
        sDescription := 'Loop using reference objects',
        stControlledVariableReference :=
            F_BACnet_Reference(fbLoopRef_CtrlVar, PropPresentValue),
        stSetpointReference :=
            F_BACnet_Reference(fbLoopRef_Setpt, PropPresentValue),
        stManipulatedVariableReference :=
            F_BACnet_Reference(fbLoopRef_Y, PropPresentValue),
    eOutputUnit := E_BA_Unit.eOther_Percent,
    eAction := E_BA_Action.eReverse,
    fProportionalConstant := 5.0,
    fIntegralConstant := 180
);
END_VAR
------------------------------------------------------------------
// internal control loop
fbLoopInternal.fCtrlVar := fCtrlVal;
fbLoopInternal();

// control loop using external object references
fbLoopRef_Setpt();
fbLoopRef_CtrlVar();
fbLoopRef_Y();
fbLoopRef();
```

6.2.7 TimeSynchronization

BACnet TimeSynchronization may be based upon the local time. In this case both the time master and the time recipients must be location within the same time zone. If the devices are located in different time zones, UTC-TimeSynchronization may be used instead. For this, the property UTC_Offset in the device object must be set to the offset in minutes to UTC. In addition, daylight saving information must be provided by the device and must be calculated in addition to the UTC time.

Implementing a BACnet server does normally not require any specific action. In case a TimeSynchronization message is received, the internal controller clock is updated.

If the controller is acting as a time master, the function block FB_BACnet_Adapter provides two methods to synchronize clocks in other BACnet devices.

TimeSync

TimeSyncEx

TimeSynchronization is an unconfirmed service. A response from the time recipients to the time master is not expected. All devices receiving the time synchronization message shall adjust their local clock.

Example for a time master functionality:

```
VAR
    stDateTimeSync : ST_BA_DateTime;
    bSuccess : BOOL;
    bSync : BOOL;
END_VAR
------------------------------------------------------------------
If bSync then
    bSync := FALSE;
    bSuccess := BACnet_Globals.DefaultAdapter.TimeSync(pDateTime := ADR(stDateTimeSync),
bSendBroadcast := TRUE);
END_IF
```

6.2.8 Retrieving diagnosis information

To retrieve BACnet diagnosis information the FB_BACnet_Adapter provides this method: GetDiagnosis.

GetDiagnosis provides access to the built-in diagnosis in the TwinCAT System Manager, which can be viewed under the "Diagnosis" tab of a BACnet device. In addition, each FB_BACnet_Client provides additional diagnosis information (m_stDiag) for each client and thus offers two further options for monitoring client connections:

Roundtrip measurement: It is possible to determine the time of a complete cyclic request.

Diagnosis of the different request types, this is displayed per client connection in the diagnosis.

Sample to retrieve diagnosis information:

```
VAR
    fbDevice : FB_BACnet_Device;
    stDiagnosis : ST_BACnet_Diagnosis;
    bSuccess : BOOL;
    bGetDiagnosis : BOOL;
END_VAR
------------------------------------------------------------------
fbDevice();
IF bGetDiagnosis THEN
    bGetDiagnosis := FALSE;
    bSuccess := BACnet_Globals.DefaultAdapter.GetDiagnosis( ADR( stDiagnosis ) );
END_IF
```

6.2.9 Scanning other BACnet devices

In some cases it is necessary to scan the BACnet network (e. g. from a visualization) to identify external BACnet devices. The FB_BACnet_Adapter provides two methods to start a scan process:

StartScan

StartScanEx

Calling one of these functions generates a BACnet Who-Is request to the network. All devices receiving this request and matching the request parameters shall respond using the I-Am service. The result of this scan process can be retrieved using the FB_BACnet_Adapter method *GetScanResult* after waiting an appropriate time.

GetScanResult returns the number of external devices found on the network, regardless of the buffer size to store the results. In case of an error this function returns -1.

Example scanning for other BACnet devices:

```
VAR
    fbDevice : FB_BACnet_Device;
    a_stScanResult : ARRAY[0..MAX_SCANRESULTS] OF ST_BACnetRM_ScanResult;
    bSuccessScan : BOOL;
    bScan : BOOL;
    fbWaitTimer : TON;
    nScanResult : DINT;
END_VAR
VAR CONSTANT
    MAX_SCANRESULTS : UDINT := 200;
    tWaitTime : TIME := T#5S;
END_VAR
------------------------------------------------------------------
fbDevice();
fbWaitTimer( IN := NOT fbWaitTimer.Q, PT := tWaitTime );

IF bScan THEN
    bScan := FALSE;
    nScanResult := -1;
    bSuccessScan := BACnet_Globals.DefaultAdapter.StartScan();
END_IF

IF bSuccessScan AND fbWaitTimer.Q THEN
    bSuccessScan := FALSE;
    nScanResult := BACnet_Globals.DefaultAdapter.GetScanResult( ADR( a_stScanResult ),
MAX_SCANRESULTS );
END_IF
```

6.2.10 Creating a structured view (DPAD)

A DPAD (**D**ata **P**oint **A**ddressing **D**escription) is used to navigate to BACnet objects using the facility view, not the technical view. A facility manager knows in which facility, building, floor, room, etc. a BACnet object is located.

It is good practice to place a shortcut for the DPAD into the object name property of the BACnet objects and a (most likely larger) information into the description property. In addition the object type Structured View (FB_BACnet_View) may be used to provide the user-view to the facility hierarchy.

The following code illustrates how to create a navigation using the operator '\/' (Backslash and Slash). This operator can be placed to the following properties (type Character String):

Object Name

Description

Event Message Text

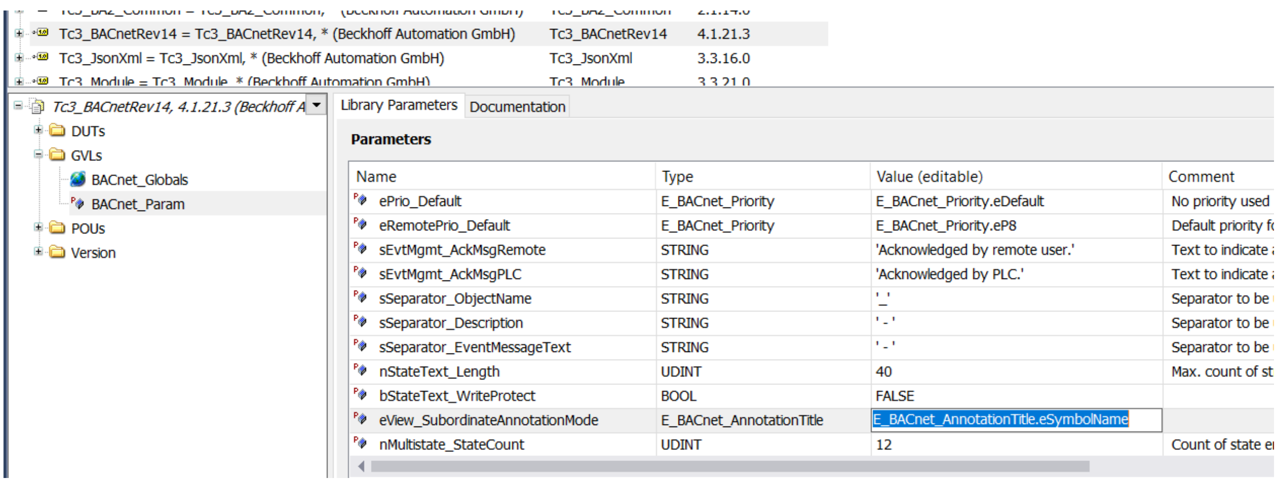
At runtime the separator is replaced by the characters specified in the Global / BACnet_Param section.

Code example for creating a DPAD structure

```
PROGRAM MAIN
VAR
    fbDPADFirstLevel : FB_BACnet_View := (
        eNodeType := E_BACnet_NodeType.eArea,
        sObjectName := '\/A',
        sDescription := '\/Facilities');

    fbDPADSecondLevel : FB_BACnet_View := (
        iParent := fbDPADFirstLevel,
        eNodeType := E_BACnet_NodeType.eOrganizational,
        sObjectName := '\/B',
        sDescription := '\/Building');

    fbDPADThirdLevel : FB_BACnet_View := (
        iParent := fbDPADSecondLevel,
        eNodeType := E_BACnet_NodeType.eNetwork,
        sObjectName := '\/C',
        sDescription := '\/Floor');

    fbAi : FB_BACnet_AI := (
        iParent := fbDPADThirdLevel,
        sObjectName := '\/ObjectName',
        sDescription := '\/Description',
        sDeviceType := 'TemperatureSensor',
        eUnit := E_BA_Unit.eTemperature_DegreesCelsius,
        fMinPresValue := -50.0,
        fMaxPresValue := 150.0,
        fHighLimit := 100,
        fLowLimit := -25,
        bHighLimitEnable := TRUE,
        bLowLimitEnable := TRUE,
        nNotificationClass := 10,
        aEventEnable := [ TRUE, TRUE, FALSE ],
        aEventMessageTextsConfig := [ '\/Alarm', '\/Fault', '\/Normal' ]);
END_VAR
-----------------------------------------------------------------
fbDPADFirstLevel();
fbDPADSecondLevel();
fbDPADThirdLevel();
fbAi();
```

This code generates three Structured View objects and one Analog Input object and connects the DPAD through the iParent elements referring to the calling function block.

The selection how the tree items are displayed in the system manager can be specified in the Global Params of the library instance:



Tc3_BACnetRev14 = Tc3_BACnetRev14, * (Beckhoff Automation GmbH)	Tc3_BACnetRev14	4.1.21.3
Tc3_JsonXml = Tc3_JsonXml, * (Beckhoff Automation GmbH)	Tc3_JsonXml	3.3.16.0
Tc3_Module = Tc3_Module, * (Beckhoff Automation GmbH)	Tc3_Module	3.3.21.0

Tc3_BACnetRev14, 4.1.21.3 (Beckhoff A
- DUTs
- GVLs
  - BACnet_Globals
  - BACnet_Param
- POUs
- Version

Library Parameters | Documentation

Parameters

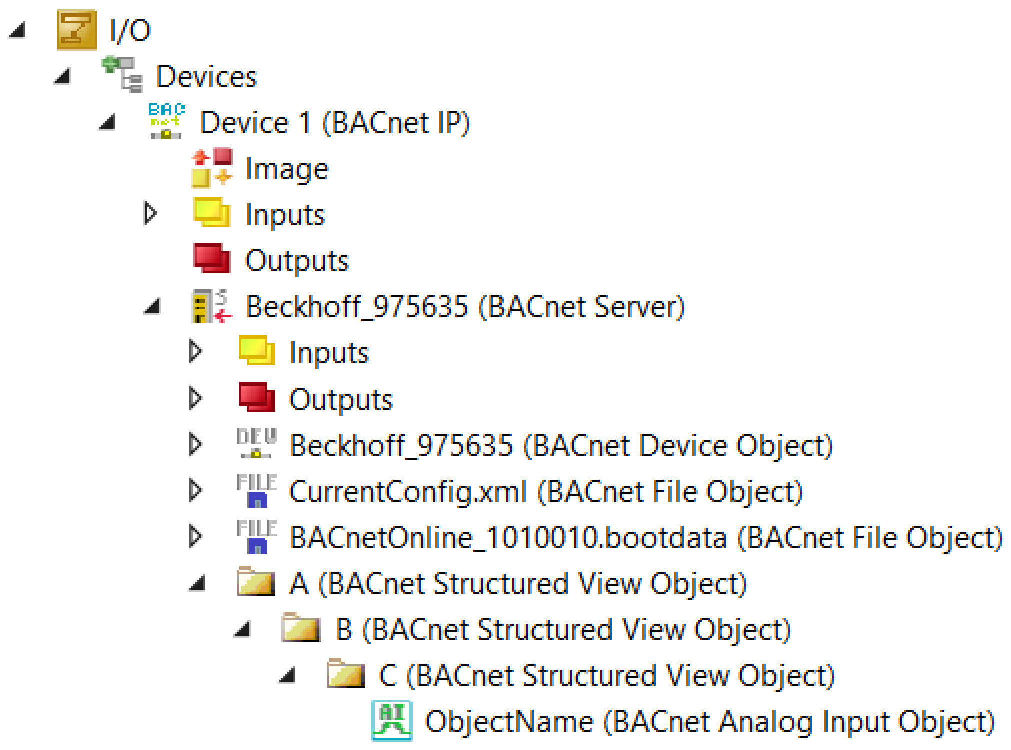
Name	Type	Value (editable)	Comment
ePrio_Default	E_BACnet_Priority	E_BACnet_Priority.eDefault	No priority used
eRemotePrio_Default	E_BACnet_Priority	E_BACnet_Priority.eP8	Default priority f
sEvtMgmt_AckMsgRemote	STRING	'Acknowledged by remote user.'	Text to indicate
sEvtMgmt_AckMsgPLC	STRING	'Acknowledged by PLC.'	Text to indicate
sSeparator_ObjectName	STRING	'_'	Separator to be
sSeparator_Description	STRING	' - '	Separator to be
sSeparator_EventMessageText	STRING	' - '	Separator to be
nStateText_Length	UDINT	40	Max. count of st
bStateText_WriteProtect	BOOL	FALSE	
eView_SubordinateAnnotationMode	E_BACnet_AnnotationTitle	E_BACnet_AnnotationTitle.eSymbolName	
nMultistate_StateCount	UDINT	12	Count of state e

eSymbolName: The symbol name from the PLC is used as tree item name.

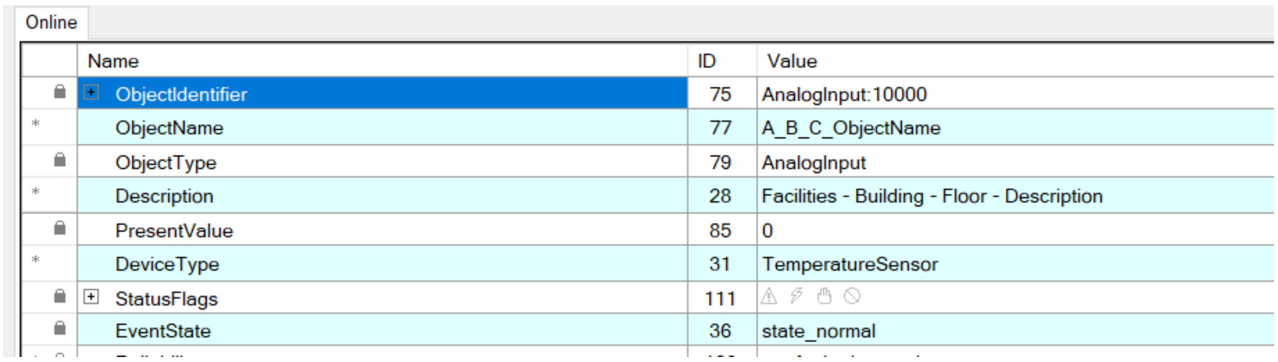
eObjectName: The BACnet object name is used as tree item name.

eDescription: The BACnet description is used as tree item name.

The system manager tree looks like this (in the example below, the object name was used):

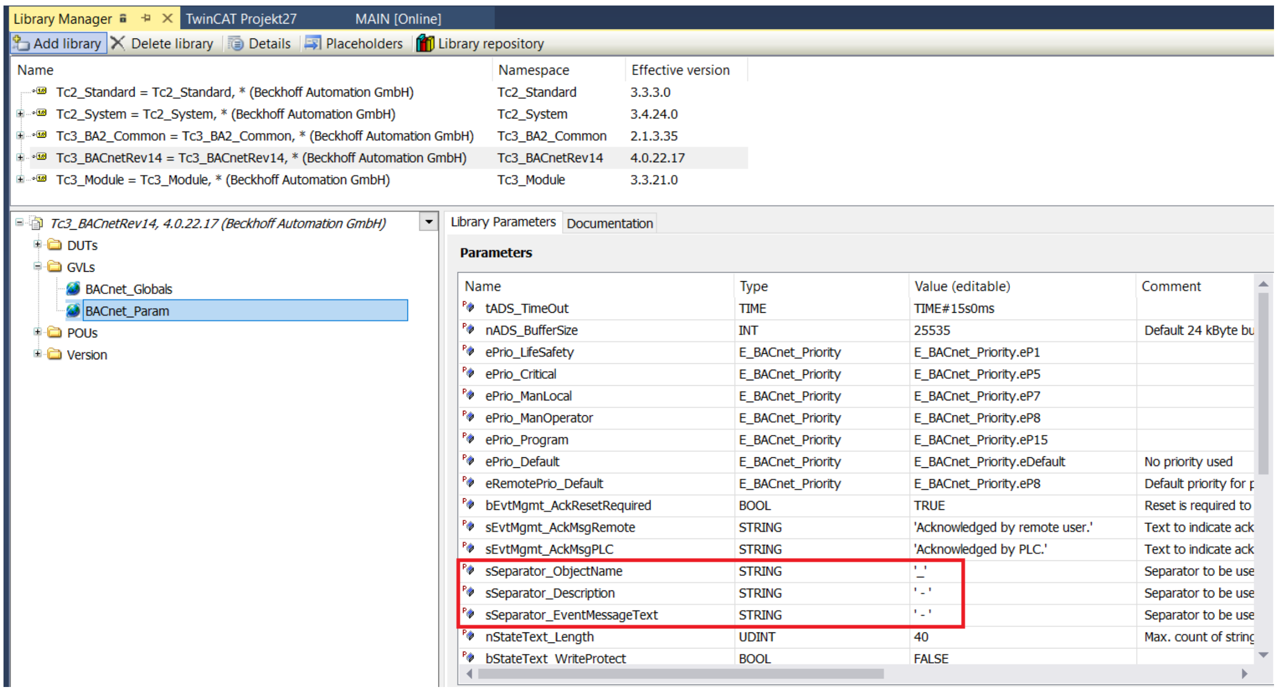


```
I/O
  Devices
    Device 1 (BACnet IP)
      Image
      Inputs
      Outputs
      Beckhoff_975635 (BACnet Server)
        Inputs
        Outputs
        Beckhoff_975635 (BACnet Device Object)
        CurrentConfig.xml (BACnet File Object)
        BACnetOnline_1010010.bootdata (BACnet File Object)
        A (BACnet Structured View Object)
          B (BACnet Structured View Object)
            C (BACnet Structured View Object)
              ObjectName (BACnet Analog Input Object)
```



	Name	ID	Value
🔒	⊞ ObjectIdentifier	75	AnalogInput:10000
*	ObjectName	77	A_B_C_ObjectName
🔒	ObjectType	79	AnalogInput
*	Description	28	Facilities - Building - Floor - Description
🔒	PresentValue	85	0
*	DeviceType	31	TemperatureSensor
🔒	⊞ StatusFlags	111	⚠ ⚡ ✋ ⊘
🔒	EventState	36	state_normal

The separators used for the three properties possibly containing a DPAD (Object Name, Description and Event Message Texts) can be individually specified in the Global Params setting of the library.



Name	Namespace	Effective version
Tc2_Standard = Tc2_Standard, * (Beckhoff Automation GmbH)	Tc2_Standard	3.3.3.0
Tc2_System = Tc2_System, * (Beckhoff Automation GmbH)	Tc2_System	3.4.24.0
Tc3_BA2_Common = Tc3_BA2_Common, * (Beckhoff Automation GmbH)	Tc3_BA2_Common	2.1.3.35
Tc3_BACnetRev14 = Tc3_BACnetRev14, * (Beckhoff Automation GmbH)	Tc3_BACnetRev14	4.0.22.17
Tc3_Module = Tc3_Module, * (Beckhoff Automation GmbH)	Tc3_Module	3.3.21.0

Tc3_BACnetRev14, 4.0.22.17 (Beckhoff Automation GmbH)
- DUTs
- GVLs
  - BACnet_Globals
  - BACnet_Param
- POUs
- Version

Parameters

Name	Type	Value (editable)	Comment
tADS_TimeOut	TIME	TIME#15s0ms	
nADS_BufferSize	INT	25535	Default 24 kByte bu
ePrio_LifeSafety	E_BACnet_Priority	E_BACnet_Priority.eP1	
ePrio_Critical	E_BACnet_Priority	E_BACnet_Priority.eP5	
ePrio_ManLocal	E_BACnet_Priority	E_BACnet_Priority.eP7	
ePrio_ManOperator	E_BACnet_Priority	E_BACnet_Priority.eP8	
ePrio_Program	E_BACnet_Priority	E_BACnet_Priority.eP15	
ePrio_Default	E_BACnet_Priority	E_BACnet_Priority.eDefault	No priority used
eRemotePrio_Default	E_BACnet_Priority	E_BACnet_Priority.eP8	Default priority for p
bEvtMgmt_AckResetRequired	BOOL	TRUE	Reset is required to
sEvtMgmt_AckMsgRemote	STRING	'Acknowledged by remote user.'	Text to indicate ack
sEvtMgmt_AckMsgPLC	STRING	'Acknowledged by PLC.'	Text to indicate ack
sSeparator_ObjectName	STRING	'_'	Separator to be use
sSeparator_Description	STRING	' - '	Separator to be use
sSeparator_EventMessageText	STRING	' - '	Separator to be use
nStateText_Length	UDINT	40	Max. count of string
bStateText_WriteProtect	BOOL	FALSE	

6.2.11 Linking hardware using attributes

Using the attribute “TcLinkTo”, hardware terminals can be connected the raw value and raw state variables. To determine the TIID path, navigate to the terminal and copy the path.

Example:

```
{attribute 'TcLinkTo' :=
    '.nRawVal  := TIID^Device 2 (EtherCAT)^Term 1 (EK1200)^Term 4 (EL3214)^RTD Inputs Channel 
1^Value;
     .nRawState := TIID^Device 2 (EtherCAT)^Term 1 (EK1200)^Term 4 (EL3214)^RTD Inputs Channel 
1^Status' }
fbAi_IO : FB_BACnet_AI_IO := (
    fMinPresValue := -150.0,
    fMaxPresValue := 150.0
);
```

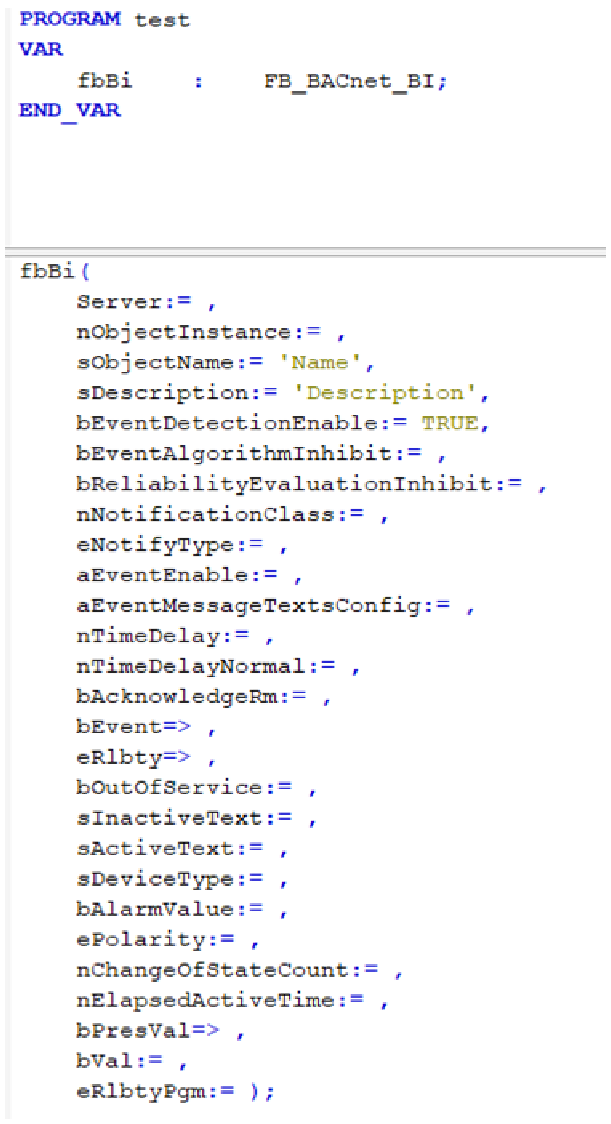
The links established using this procedure are shown as green arrow symbols in the system manager.

6.3 Important notes using the library

This chapter contains notes and recommendations how to use the library when programming a BACnet server.

6.3.1 Declaring properties at start-up

In normal PLC programming you will probably find calls like this:



```
PROGRAM test
VAR
    fbBi    :    FB_BACnet_BI;
END_VAR
```

```
fbBi(
    Server:= ,
    nObjectInstance:= ,
    sObjectName:= 'Name',
    sDescription:= 'Description',
    bEventDetectionEnable:= TRUE,
    bEventAlgorithmInhibit:= ,
    bReliabilityEvaluationInhibit:= ,
    nNotificationClass:= ,
    eNotifyType:= ,
    aEventEnable:= ,
    aEventMessageTextsConfig:= ,
    nTimeDelay:= ,
    nTimeDelayNormal:= ,
    bAcknowledgeRm:= ,
    bEvent=> ,
    eRlbty=> ,
    bOutOfService:= ,
    sInactiveText:= ,
    sActiveText:= ,
    sDeviceType:= ,
    bAlarmValue:= ,
    ePolarity:= ,
    nChangeOfStateCount:= ,
    nElapsedActiveTime:= ,
    bPresVal=> ,
    bVal:= ,
    eRlbtyPgm:= );
```

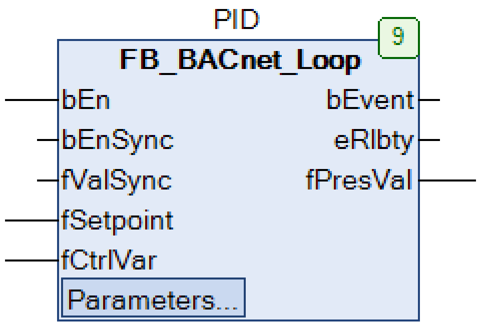
If the values are constantly written by the PLC, writing from the BACnet would have no effect, the values would be overwritten immediately. Therefore, especially the properties that do not change often or at all should be declared as initial parameters in the variable area. The following code shows an example of initialization of properties at the time of the first PLC start.

```
PROGRAM test
VAR
       fbBI    :    FB_BACnet_BI    := (
                                 sObjectName := 'Name',
                                 sDescription := 'Description',
                                 bEventDetectionEnable := TRUE
                                                              );
       bDescriptionChanged          :     BOOL;
END_VAR


--------------

fbBI();
IF bDescriptionChanged THEN
       bDescriptionChanged := FALSE;
       fbBI.sDescription := 'New Description';
END_IF
```

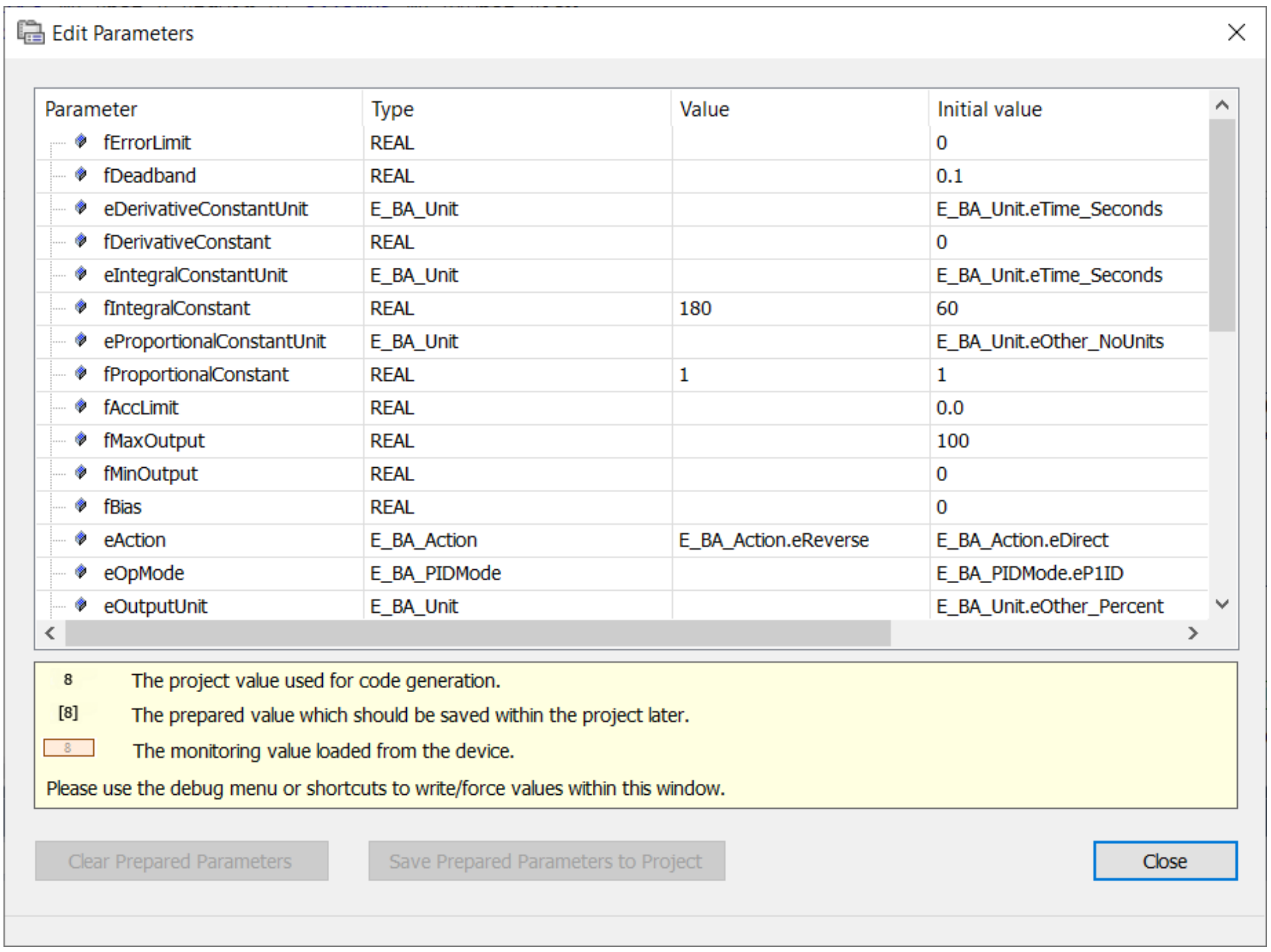
In the above example, the property values (only those that require configuration) are taken from the declaration at the time of the first PLC start. If a configuration property needs to be changed, we recommend a write-on-change procedure, as shown here, rather than cyclic writing. Furthermore, the implementation of a cyclic writing would lead to the fact that no more values are accepted that were written by BACnet.

6.4 Parameter dialogs

Using CFC (Continuous Function Chart), accessing the configuration parameters is provided by parameter dialogs. Open the parameter dialog by using the button in the left corner.



Values shown in this dialog may be changed in the column *Value* and can be written to the PLC by using the *write values* function in the TwinCAT PLC menu. The values are also stored persistently using the PLC persistence.


Edit Parameters
Clear Prepared Parameters
Save Prepared Parameters to Project
Close

Parameter	Type	Value	Initial value
fErrorLimit	REAL		0
fDeadband	REAL		0.1
eDerivativeConstantUnit	E_BA_Unit		E_BA_Unit.eTime_Seconds
fDerivativeConstant	REAL		0
eIntegralConstantUnit	E_BA_Unit		E_BA_Unit.eTime_Seconds
fIntegralConstant	REAL	180	60
eProportionalConstantUnit	E_BA_Unit		E_BA_Unit.eOther_NoUnits
fProportionalConstant	REAL	1	1
fAccLimit	REAL		0.0
fMaxOutput	REAL		100
fMinOutput	REAL		0
fBias	REAL		0
eAction	E_BA_Action	E_BA_Action.eReverse	E_BA_Action.eDirect
eOpMode	E_BA_PIDMode		E_BA_PIDMode.eP1ID
eOutputUnit	E_BA_Unit		E_BA_Unit.eOther_Percent

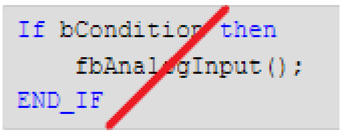
8	The project value used for code generation.
[8]	The prepared value which should be saved within the project later.
8	The monitoring value loaded from the device.

Please use the debug menu or shortcuts to write/force values within this window.

6.4.1 Always call BACnet function blocks cyclically

It is very important that all BACnet function blocks are called once per cycle and only once!

Do **not** call the function blocks in this way:



```
If bCondition then
    fbAnalogInput();
END_IF
```

This would create a BACnet object that is no longer updated if bCondition is FALSE. The object looks faulty to the client trying to access it.

However, it is possible to declare a BACnet object as a variable and not call it at all. In this case the BACnet object is not generated and therefore does not exist in the BACnet server.

6.4.2 Call BACnet function blocks using the same cycle time

Calling different BACnet FBs using different cycle times leads to runtime errors caused by delayed library calls. The BACnet supplement and the PLC library require a synchronization at start-up. Calling BACnet FBs delayed in any way is not possible. If objects need to be created or deleted at runtime, refer to the section Dynamic Object Manager in this manual.

6.5 Calculating the router memory

Memory for the BACnet object representation in the PLC is taken from the router memory. The default setting in TwinCAT is 32MB and may need to be increased in case of a larger BACnet object database. The log-buffer of Trendlog or Eventlog objects is taken from the router memory as well.

As an average the properties of BACnet objects require 20 KB each.

To reserve some memory space for other purposes, the library raises an error condition when more 60% of the available router memory is requested by the BACnet library. In this case, further BACnet objects are not created anymore so the project database may be incomplete.

On Windows Embedded compact (Windows CE) systems, the maximum router memory is limited to 200MB. On Big-Windows (Windows 10) or BSD the router memory depends on the non-paged pool. E. g. if the total non-paged pool is 1 GB, the router memory may be increased to 600-700MB. Please note, increasing the router memory may require a reboot of the PLC.

6.5.1 Example calculation of required router memory

The example below shows a calculation of the required router memory.

Example requirement:

280 digital and analog objects = 280 x 20 KB = 5,600 KB

280 Trendlog = 280 x 20 KB = 5,600 KB (properties only)

Trendlog buffer size (7 days / 10 min. interval) = 1440 / 10 * 7 = 1008 records per Trendlog

Memory requirement for Trendlog buffer: 280 * 1008 * 56 = 15,805,440 (56 bytes per data record in log memory)

Total Router Memory required

5,600,000 bytes normal objects

5,600,000 bytes Trendlog objects

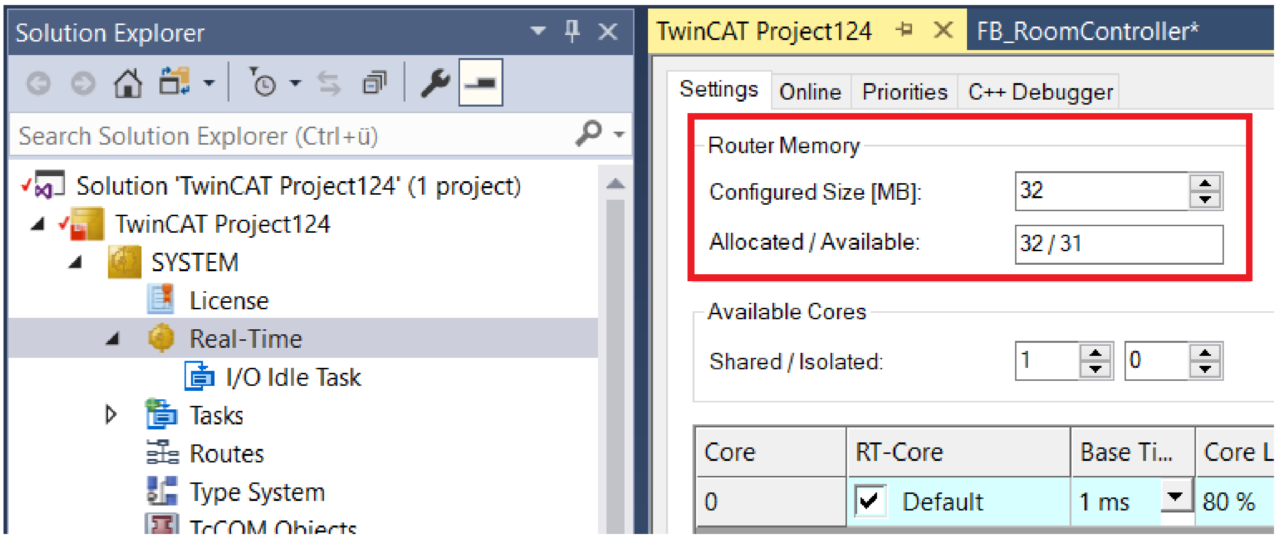
15,805,440 bytes Trendlog buffer

Total Router memory for BACnet

27,005,440 bytes

Please note that the total BACnet router memory is only 60 % of the total required router memory, so the default value of 32 MB would not be sufficient for this example and must be increased.

The amount of router memory can be specified in the Real-Time settings of TwinCAT. Please note, that the router memory will be assigned after restarting the controller.



Configured Size [MB]: 32

Allocated / Available: 32 / 31

Shared / Isolated: 1 0

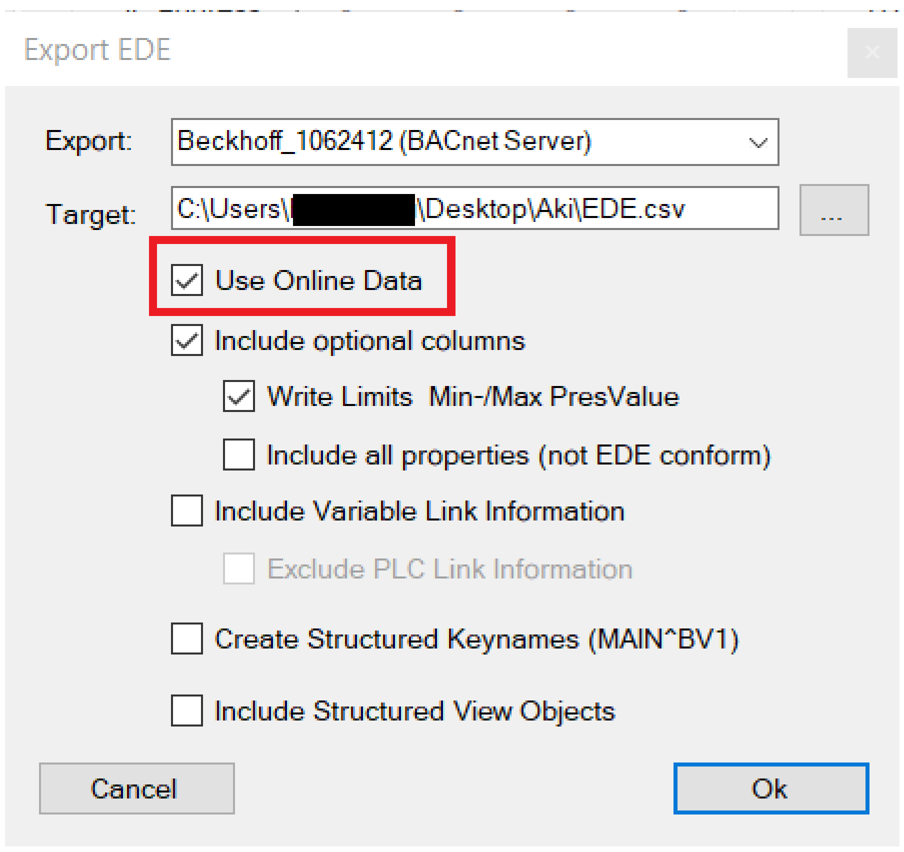
Core	RT-Core	Base Ti...	Core L
0	✓ Default	1 ms	80 %

6.6 Specific Functionality

This chapter describes specific functionality using the BACnet supplement or the library.

6.6.1 Generating an EDE file

When exporting an EDE file, it is important to check the “Use Online Data” checkbox as shown below. Due to the dynamic creation by the BACnet Rev 14 library, the objects need to be visible at the time of the EDE file creation.


Export EDE

Export: Beckhoff_1062412 (BACnet Server)

Target: C:\Users\[redacted]\Desktop\Aki\EDE.csv

[✓] Use Online Data

[✓] Include optional columns

[✓] Write Limits Min-/Max PresValue

[ ] Include all properties (not EDE conform)

[ ] Include Variable Link Information

[ ] Exclude PLC Link Information

[ ] Create Structured Keynames (MAIN^BV1)

[ ] Include Structured View Objects

6.6.2 Cycle time exceed counter

At the time of the PLC start exceeds of the cycle time may happen and are normal. The synchronization between the BACnet supplement and the PLC may take longer than the cycle time. Exceeds should not happen after the initialization phase though.

6.6.3 Enable / Disable Properties

To disable properties that are not needed for a particular object instance, the "*aDisabled*" array of the "*stSettings*" structure can be used.

The structure "*stSettings*" is also available in the Global Params. The settings in the Global Params affect all objects.

Please note that related properties (with the same number in the footnotes of the BACnet standard) must all be disabled or enabled!

6.6.4 Writeprotect Properties

To apply a write protection to specific properties, the array “*aWriteProtected*” of the structure “*stSettings*” can be used.

The structure “*stSettings*” is available in the Global Params, too. The settings in the Global Params affect all objects.

In case of a WriteProperty or WritePropertyMultiple request to those properties an error message “Write_Access_Denied” is returned to the BACnet client.

6.6.4.1 Example Disable/Writeprotect

The following example shows how to disable and writeprotect properties using stSettings.

```
fbBi : FB_BACnet_BI := (
        sObjectName := 'Example Binary Input Object',
        sDescription := 'Objectname and Description properties are read-only',
        stSettings := (
            aDisabled := [
                E_BACnetPropIdentifier.PropChangeOfStateCount,
                E_BACnetPropIdentifier.PropChangeOfStateTime,
                E_BACnetPropIdentifier.PropTimeOfStateCountReset
                    ],
            aWriteProtected := [
                E_BACnetPropIdentifier.PropObjectName,
                E_BACnetPropIdentifier.PropDescription
                    ]
            )
);
```

6.6.5 Enabling write access from the PLC

Objects representing an input information like ACC, AI, BI, etc. provide an input variable (nVal, fVal or bVal) whose value is transferred to the Present_Value property without further processing.

Objects representing an output information like AO, AV, etc. provide an input variable representing the PLC value (nVal, fVal, bVal). In addition, to activate the value at the specified program priority the variable bEnPgm needs to be set to TRUE. In case the value of bEnPgm is set to FALSE, the value will be set to NULL at the specified program priority.

In case of _5P function blocks for each of the priorities (with the exception of manual operator which is likely provided from the BMS) a separate flag is provided.

The currently active Priority can be determined from the variable eActPrio (1-16, 17=Relinquish_Default).

6.6.6 Adding a recipient to a recipient list

Entries in the recipient list property of Notification Class objects are complex data-types supporting two options. Either the device instance number may be used to identify the recipient or, as an alternative, the BACnet MAC-address (physical address) may be specified.

The example below shows both options for the variable aRecipientList:

```
aRecipientList := [
(
    stValidDays :=
        (bMonday:=TRUE, bTuesday:=TRUE, bWednesday:=TRUE, bThursday:=TRUE, bFriday:=TRUE),
    stFromTime := F_BA_ToSTTime(T#0H),
    stToTime := F_BA_ToSTTime(T#23H59M59S),
    stRecipient :=
            F_BACnet_DeviceRecipient(nDeviceInstance:=42),
        nProcessId := 10000,
        bIssueConfirmed := FALSE,
        stEventTransitions :=
            (bToOffNormal:=TRUE, bToFault:=TRUE, bToNormal:=TRUE)
),
(
    stValidDays := (bSunday:=TRUE, bSaturday:=TRUE),
    stFromTime := F_BA_ToSTTime(T#7H),
    stToTime := F_BA_ToSTTime(T#15H30M),
    stRecipient :=
            F_BACnet_EthernetRecipient(nIPAddress1:=192,168,10,200, nPort:=47808, nNetworkNr:=444),
        nProcessId := 30100,
        bIssueConfirmed := TRUE,
        stEventTransitions := (bToOffNormal:=TRUE)
)
]
```

6.6.7 Using UTF-8 characters

UTF-8 is the default character set used in the BACnet supplement. The UTF-8 character set covers all Unicode characters and requires up to 4 BYTE per character depending on the language.

Regular strings without special characters do not need any specific handling. Latin-1 characters like äöüÄÖÜß©ÔØ, etc. do not require a specific handling either.

In case of Cyrillic, African or Asian characters, a conversion is required. In this case the attribute ‘TcEncoding’ needs to be set to ‘UTF-8’. Like every attribute this applies to the variable below the attribute and needs to be repeated for each variable containing UTF-8 characters.

Example using UTF-8 characters:

{attribute 'TcEncoding' := 'UTF-8'}
sMyUTF8Text : STRING := wsLiteral_TO_UTF8("äöüßéèêµ€° Ἀθῆναι İstanbul Київ");

Please note to use double quotation marks in case of WSTRING variables!

6.7 FB_BACnet_Server

This function block represents the BACnet server in the PLC. In addition to information about memory usage and memory state, there are various options, such as acknowledging alarms, writing persistent data, resetting the server's error state, or iterating over the object database.

This function block is called automatically and normally does not have to be called within the PLC program.
If you want to access the properties of the server, use the already existing instance from the BACnet Globals.

6.8 FB_BACnet_Device

This function block may be used to change settings of the device object at runtime.

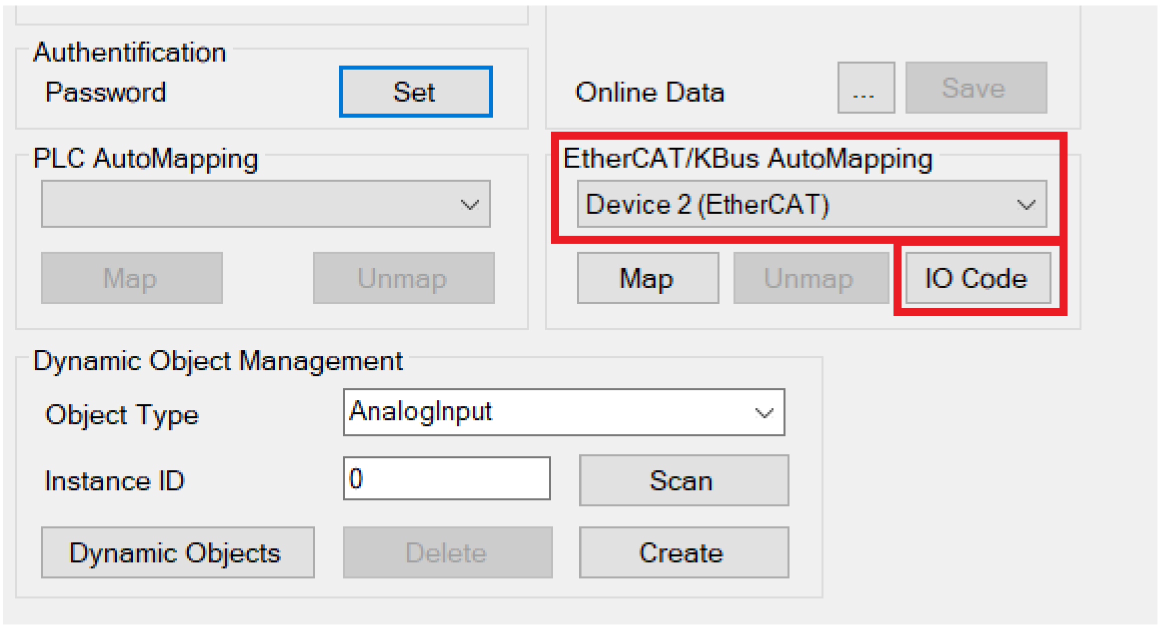
Example:

```
VAR
    fbDevice : FB_BACnet_Device;
    bChangeDeviceObj : BOOL;
END_VAR
------------------------------------------------------------------
fbDevice();
IF bChangeDeviceObj THEN
    bChangeDeviceObj := FALSE;
    fbDevice.sObjectName := 'TEST_ObjName';
    fbDevice.sDescription := 'TEST_Description';
    fbDevice.sLocation := 'TEST_Location';
END_IF
```

6.9 IO Code

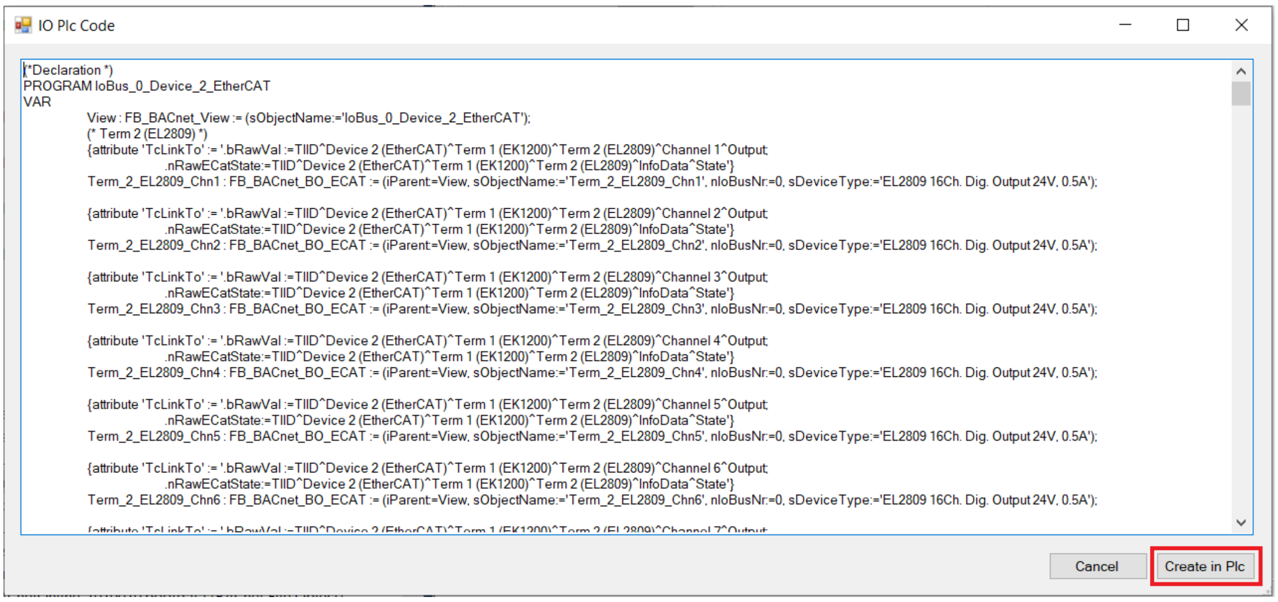
This function may be used to automatically generate the necessary program code for referencing hardware terminals to the PLC and BACnet objects. The generated code is created as a program in the PLC.

1. To start the IO Code generation, navigate to the BACnet server and select the Settings dialog.



2. Select the hardware and use the button IO Code.
   ⇨ Digital and Analog input and output terminals and channels are detected and the program code is generated.

- Digital input channels are mapped to BACnet Binary-Input objects.
- Digital output channels are mapped to BACnet Binary-Output objects.
- Analog input channels are mapped to BACnet Analog-Input objects.
- Analog output channels are mapped to BACnet Analog-Output objects.



```
(*Declaration *)
PROGRAM IoBus_0_Device_2_EtherCAT
VAR
        View : FB_BACnet_View := (sObjectName:='IoBus_0_Device_2_EtherCAT');
        (* Term 2 (EL2809) *)
        {attribute 'TcLinkTo' := '.bRawVal :=TIID^Device 2 (EtherCAT)^Term 1 (EK1200)^Term 2 (EL2809)^Channel 1^Output;
                    .nRawECatState:=TIID^Device 2 (EtherCAT)^Term 1 (EK1200)^Term 2 (EL2809)^InfoData^State'}
        Term_2_EL2809_Chn1 : FB_BACnet_BO_ECAT := (iParent:=View, sObjectName:='Term_2_EL2809_Chn1', nIoBusNr:=0, sDeviceType:='EL2809 16Ch. Dig. Output 24V, 0.5A');

        {attribute 'TcLinkTo' := '.bRawVal :=TIID^Device 2 (EtherCAT)^Term 1 (EK1200)^Term 2 (EL2809)^Channel 2^Output;
                    .nRawECatState:=TIID^Device 2 (EtherCAT)^Term 1 (EK1200)^Term 2 (EL2809)^InfoData^State'}
        Term_2_EL2809_Chn2 : FB_BACnet_BO_ECAT := (iParent:=View, sObjectName:='Term_2_EL2809_Chn2', nIoBusNr:=0, sDeviceType:='EL2809 16Ch. Dig. Output 24V, 0.5A');

        {attribute 'TcLinkTo' := '.bRawVal :=TIID^Device 2 (EtherCAT)^Term 1 (EK1200)^Term 2 (EL2809)^Channel 3^Output;
                    .nRawECatState:=TIID^Device 2 (EtherCAT)^Term 1 (EK1200)^Term 2 (EL2809)^InfoData^State'}
        Term_2_EL2809_Chn3 : FB_BACnet_BO_ECAT := (iParent:=View, sObjectName:='Term_2_EL2809_Chn3', nIoBusNr:=0, sDeviceType:='EL2809 16Ch. Dig. Output 24V, 0.5A');

        {attribute 'TcLinkTo' := '.bRawVal :=TIID^Device 2 (EtherCAT)^Term 1 (EK1200)^Term 2 (EL2809)^Channel 4^Output;
                    .nRawECatState:=TIID^Device 2 (EtherCAT)^Term 1 (EK1200)^Term 2 (EL2809)^InfoData^State'}
        Term_2_EL2809_Chn4 : FB_BACnet_BO_ECAT := (iParent:=View, sObjectName:='Term_2_EL2809_Chn4', nIoBusNr:=0, sDeviceType:='EL2809 16Ch. Dig. Output 24V, 0.5A');

        {attribute 'TcLinkTo' := '.bRawVal :=TIID^Device 2 (EtherCAT)^Term 1 (EK1200)^Term 2 (EL2809)^Channel 5^Output;
                    .nRawECatState:=TIID^Device 2 (EtherCAT)^Term 1 (EK1200)^Term 2 (EL2809)^InfoData^State'}
        Term_2_EL2809_Chn5 : FB_BACnet_BO_ECAT := (iParent:=View, sObjectName:='Term_2_EL2809_Chn5', nIoBusNr:=0, sDeviceType:='EL2809 16Ch. Dig. Output 24V, 0.5A');

        {attribute 'TcLinkTo' := '.bRawVal :=TIID^Device 2 (EtherCAT)^Term 1 (EK1200)^Term 2 (EL2809)^Channel 6^Output;
                    .nRawECatState:=TIID^Device 2 (EtherCAT)^Term 1 (EK1200)^Term 2 (EL2809)^InfoData^State'}
        Term_2_EL2809_Chn6 : FB_BACnet_BO_ECAT := (iParent:=View, sObjectName:='Term_2_EL2809_Chn6', nIoBusNr:=0, sDeviceType:='EL2809 16Ch. Dig. Output 24V, 0.5A');
```

3. Using the button Create in Plc generates the code as a program in the folder POUs/BACnet_IoBus.
4. Call this program cyclically, e. g. from the POU MAIN.

```
IoBus_0_Device_2_EtherCAT();
```

6.10 Recommended workflow / BACnet persistence

In building automation projects, changes of parameter settings very often happen while programming and configuring the application.

It is recommended to adjust parameters like P, I, D or Hi-Limit / Lo-Limit, even Description texts, etc. without using the BACnet persistence as long as possible. As final step before handing over the building automation application, the BACnet persistence should be activated.

Normally the values are synchronized at start-up from the BACnet supplement to the PLC. If BACnet persistence is activated, values previously written from BACnet will overwrite values in the PLC.

The FB_BACnet_Server provides a variable eInitMode to define the direction of this synchronization:

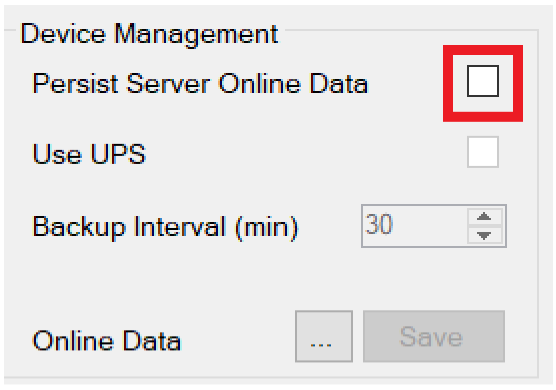
- eInitReset: This value is set after a Reset to Origin was performed. BACnet objects in the stack will be removed and eInitMode will be set to eInitToPlc.
- eInitToPlc (default): Values are synchronized from BACnet to the PLC.
- eInitForceFromPlc: Values are synchronized from the PLC to BACnet.

6.11 Persistence

The property values changed at runtime are stored in the file *BACnetOnline_1010010.bootdata* in the TwinCAT boot folder. Additionally there is a backup file named *BACnetOnline_1010010.bootdata-old*.

Switch on and configure BACnet persistence

In the settings of the BACnet server, persistence can be switched on and off in the dialog box **Device Management**.


Device Management

Persist Server Online Data	☐
Use UPS	☐
Backup Interval (min)	30
Online Data	... Save

Without using an uninterruptible power supply (UPS), persistence is performed at a configurable interval. The default value of 30 minutes should not be undercut, otherwise the flash medium could be subject to greater wear.

If a UPS is used, the property *Use UPS* can be activated. In this case, the PLC program must ensure that the function blocks of the UPS are used to detect a power supply failure. In this case, the function for persisting the data must be called on the part of the PLC.

Calling persistence from the PLC program

Two different methods are available to trigger persistence.

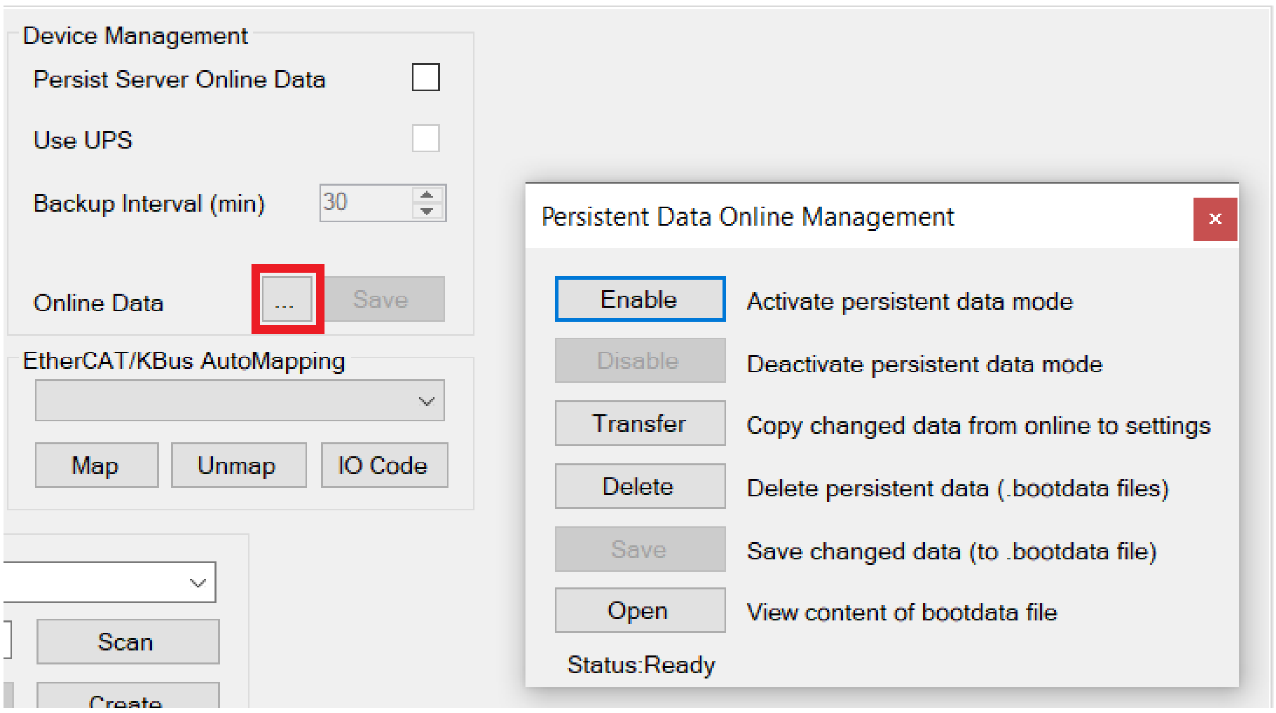
1. Setting the variable *bWritePersistent*::
   ```
   BACnet_Globals.DefaultServer.bWritePersistent := TRUE;
   ```
   The variable must also be reset from the PLC program.
2. Calling the SavePersistentStackData method:
   ```
   bSuccess := BACnet_Globals.DefaultServer.SavePersistentStackData();
   ```
   The return value can be used to detect whether the call was successful.

View and edit persistence data

✓ You have opened the Device Management dialog box.

1. Select the **Online Data > …** option.

⇨ The dialog for editing the persistence data appears, the contents of which are explained below.



Device Management
Persist Server Online Data
Use UPS
Backup Interval (min)
30
Online Data
...
Save
EtherCAT/KBus AutoMapping
Map
Unmap
IO Code
Scan
Create
Persistent Data Online Management
Enable
Activate persistent data mode
Disable
Deactivate persistent data mode
Transfer
Copy changed data from online to settings
Delete
Delete persistent data (.bootdata files)
Save
Save changed data (to .bootdata file)
Open
View content of bootdata file
Status:Ready

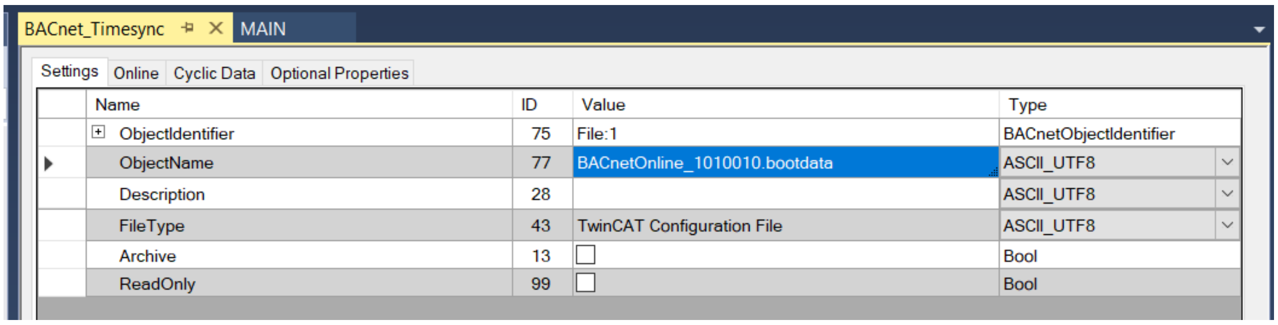
Option	Description
Enable	Switch on persistence.
Disable	Switch off persistence.
Transfer	Allows copying the persistent data to the default settings of the objects (Settings).
Delete	Deletes the files that contain the persistence data.
Save	Saves changed data in the persistence file.
Open	Can load and display a persistence file.

Accessing the persistence file with the FILE object

The persistence file is represented via BACnet within the server by the File object with instance number 1.

1. Double-click the *Value* field of the ObjectName to view the currently existing contents of the persistence file.

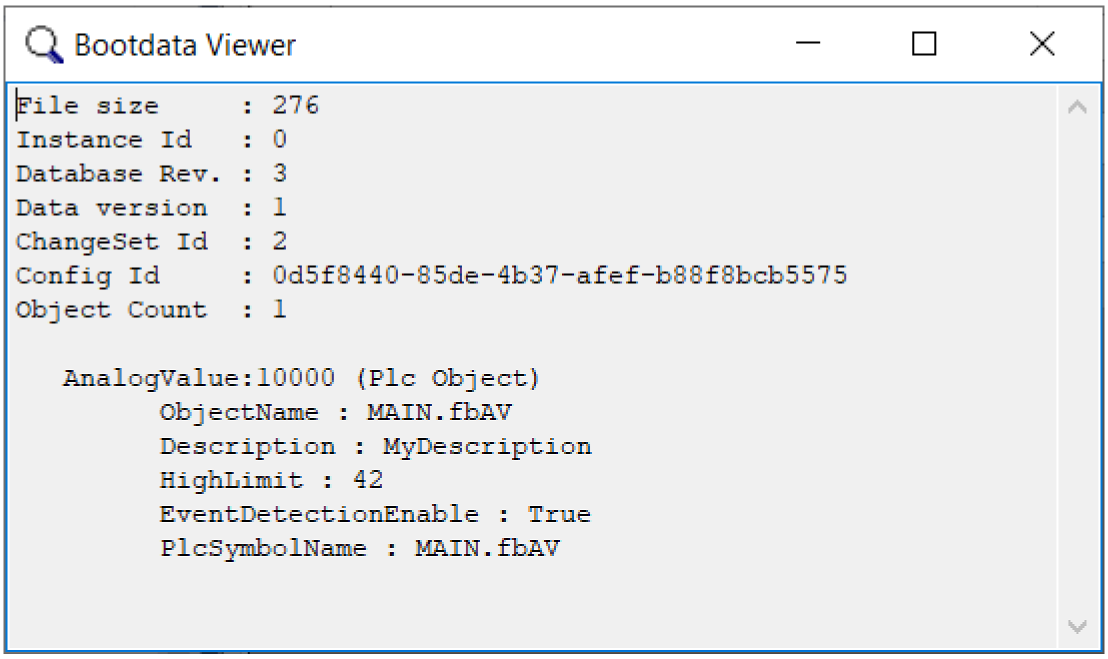
⇨ If the persistence file contains data, the Bootdata Viewer is called and the changed properties are displayed.



Name	ID	Value	Type
ObjectIdentifier	75	File:1	BACnetObjectIdentifier
ObjectName	77	BACnetOnline_1010010.bootdata	ASCII_UTF8
Description	28		ASCII_UTF8
FileType	43	TwinCAT Configuration File	ASCII_UTF8
Archive	13	☐	Bool
ReadOnly	99	☐	Bool

Bootdata Viewer

In the window **Bootdata Viewer** the changed properties from the persistence file are displayed and can also be marked and copied from there for further processing.



```
File size      : 276
Instance Id    : 0
Database Rev.  : 3
Data version   : 1
ChangeSet Id   : 2
Config Id      : 0d5f8440-85de-4b37-afef-b88f8bcb5575
Object Count   : 1

   AnalogValue:10000 (Plc Object)
         ObjectName : MAIN.fbAV
         Description : MyDescription
         HighLimit : 42
         EventDetectionEnable : True
         PlcSymbolName : MAIN.fbAV
```

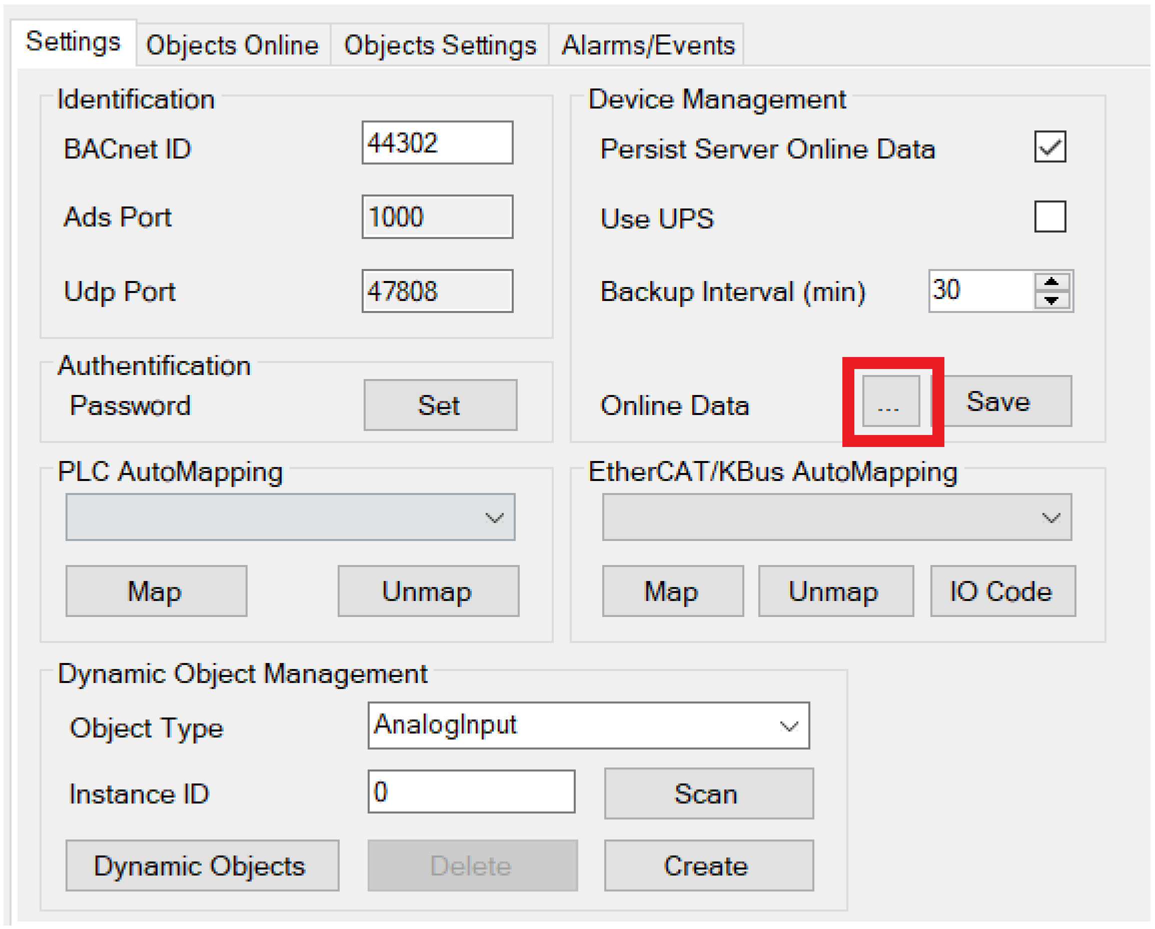
In this example, in the Analog Value object with instance number 10000, the properties `ObjectName`, `Description`, `HighLimit`, `EventDetectionEnable` and `PlcSymbolName` have been changed.

Delete persistence data

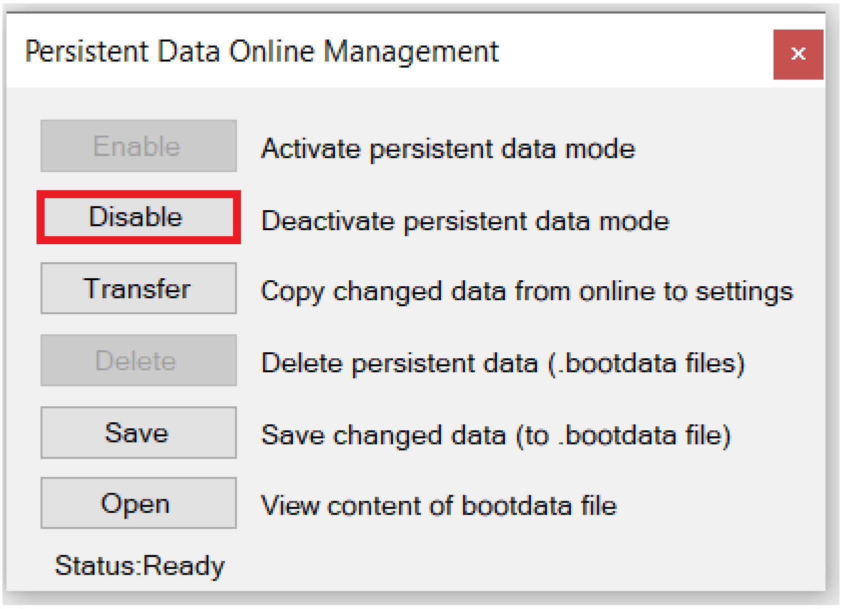
Call Dialog Online Data

1. In the BACnet server, open the **Settings** tab.

2. Click the menu item **Online Data > …**



⇨ The dialog box **Persistent Data Online Management** opens.

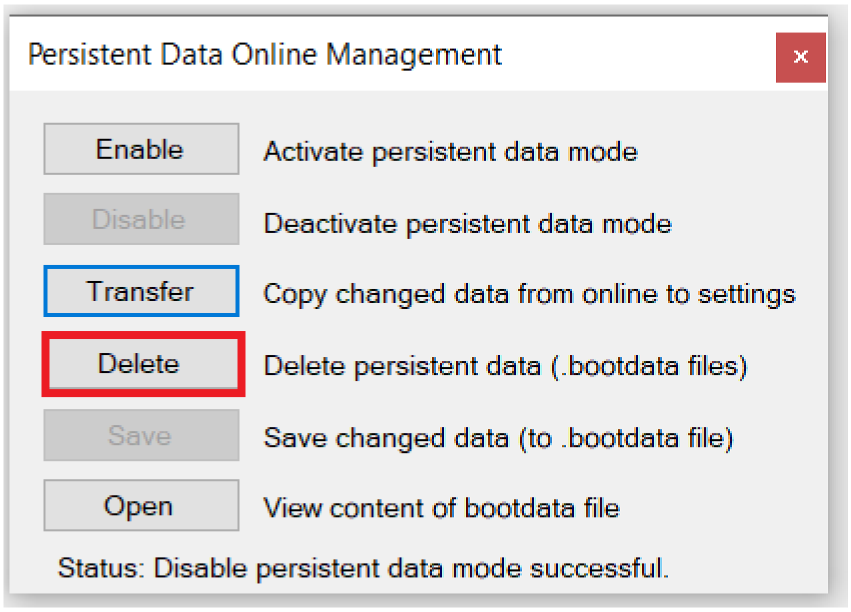

Persistent Data Online Management

Enable	Activate persistent data mode
Disable	Deactivate persistent data mode
Transfer	Copy changed data from online to settings
Delete	Delete persistent data (.bootdata files)
Save	Save changed data (to .bootdata file)
Open	View content of bootdata file

Status:Ready

3. Click the **Disable** button.

4. Then click the **Delete** button in the same menu.



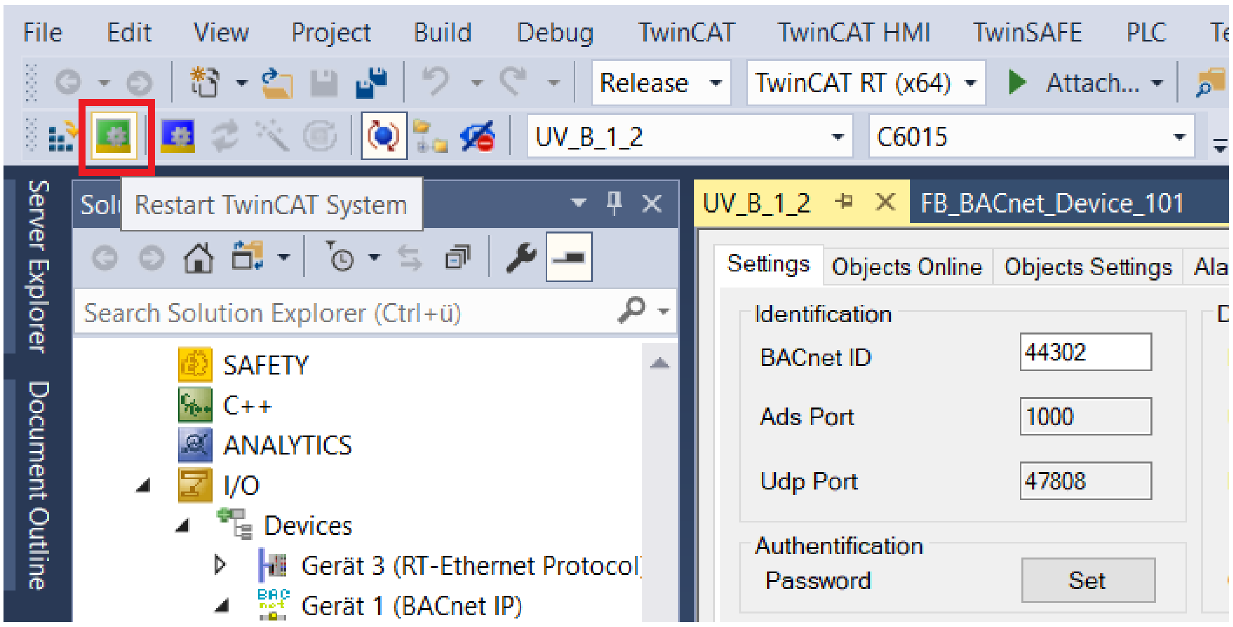
Persistent Data Online Management

Enable	Activate persistent data mode
Disable	Deactivate persistent data mode
Transfer	Copy changed data from online to settings
Delete	Delete persistent data (.bootdata files)
Save	Save changed data (to .bootdata file)
Open	View content of bootdata file

Status: Disable persistent data mode successful.

5. Note the status display.
   This provides information on whether the function could be executed successfully.

6. Finally, restart via the TwinCAT main menu by clicking the Runtime Mode button.



6.12 Time synchronization

In this description, the term *time* stands for the combination of date and time (Date and Time).

BACnet time synchronization

In BACnet, time synchronization is performed via two services:

TimeSynchronization synchronizes the time based on the local time, i.e. the timer and the time receiver must be in the same time zone.

When using *UTCTimeSynchronization* (UTC=Universal Time Coordinated) the time is transmitted as GMT (Greenwich Mean Time = local time London). The timer converts the time to GMT before sending. Using the local settings UTC_Offset and DaylightSavingsStatus (in the device object of the server), the time receiver then converts back to the local time.

UTC_Offset specifies the value in minutes relative to GMT. A positive value is to be used westward from GMT, a negative value eastward from GMT.
Western Europe has an offset of -60 minutes from GMT. This is set as the default value in the TwinCAT BACnet server and may have to be adjusted when used in other time zones with the use of UTC.

- The time synchronization has a direct effect on the recording of BACnet logging objects (Trendlog, Trendlog-Multiple and Eventlog). If the time changes, this is accepted as an entry in the log memory. Short time synchronization intervals can therefore result in many (unnecessary) entries in the log memory of the logging objects.
- There should be only one timer in a BACnet network. This must be ensured by the operator or specialist planner.
  Even if it is basically possible according to the BACnet standard to send the time synchronization to only one group of participants (multicast) or to only one single participant (unicast), in practice the time is often sent as a broadcast to *all* participants of the network.

TwinCAT as BACnet time receiver

If a TwinCAT controller receives a BACnet time synchronization, the internal time of the controller is set with this time.

TwinCAT as BACnet timer

If a TwinCAT controller is used as a BACnet timer, it must determine the exact time from an external location, e.g. by using the NTP service on an external NTP server (NTP = Network Time Protocol). For example, time.windows.com is available as a publicly available time server.

The method `TimeSyncEx im FB_BACnet_Adapter` is available for taking over the time and, if necessary, broadcasting in the network.

The following values are passed to the call as parameters:

Name	Type	Description
pDateTime	ST_BA_DateTime	Pointer to a structure of type ST_BA_DateTime (see library Tc3_BA2_Common). In this structure the date and time to be set is passed or returned in case of eSyncMode = 3.

eSyncMode	E_BACnet_TimeSyncMode	Enumeration
		The value determines the type of synchronization:
		`eSyncAndSendLocalTime := 0`
		Sets pDateTime in the BACnet stack (as Local Time) and sends the time as broadcast (TimeSync request)
		Use case: The controller receives the time via NTP and sends the local BACnet time to other controllers as broadcast.
		`eSyncOnly := 1`
		Sets pDateTime in the BACnet stack (as Local Time)
		Use case: All controllers receive the time via NTP and synchronize their own BACnet clock.
		`eSyncAndSendUtcTime := 2`
		Sets pDateTime in the BACnet stack (as UTC Time) and sends the time as broadcast (UTC Time Sync request)
		Use case: The controller receives the time via NTP and sends the BACnet time as UTC to other controllers as broadcast.
		`eSyncGetTime := 3`
		Returns the current time in the BACnet stack in pDateTime.
		Use case: Is rather rarely used. A possible use case would be, for example, a comparison of the BACnet clock with the trend recording.

6.13 Recommended cycle time of the PLC task

The default value for the PLC cycle time (PLC task) is 10 ms. In many cases this is not necessary for processing the BACnet data.

It is recommended to use a cycle time of 40-50 ms for the PLC task.

Even higher values (i.e. a slower cycle time) are not recommended, otherwise the communication via BACnet, e.g. with a MBE (Management Bedieneinrichtung, management operating equipment), will be slowed down.

7 Programming a BACnet client

A BACnet client is a reference to an external peer BACnet device. As a client, the TwinCAT BACnet supplement provides access to objects and services provided by an external device.

For each external device one reference needs to be established.

The function blocks to access BACnet objects as a client are located in the section POUs/Remote/Objects.

Difference between server and client FBs

Client FBs are designed to provide cyclic data-exchange to an external device, mostly the present value and status flags are transferred. Other properties like description, limits, state-texts, minimum-/maximum present value, etc. are not part of the standard FB implementation. Even though, the function blocks FB_BACnet_ReadProperty and FB_BACnet_WriteProperty may be used to access properties other than the standard ones provided in the FB implementation. These function blocks provide acyclic access to BACnet property data.

7.1 Writemode

Writing to the third-party device is implemented as `WriteOnChange` by default. This means that values are only written if the value has changed. A periodic writing can be set in the client FBs.

`Typ E_BACnet_Writemode` specifies write access to client properties:

`eAuto` = Automatic, the mode is determined from the properties of the client device if `TRUE`

`eCyclic` = Cyclic writing, cycle time is taken from tWriteCycleTime

`eOnChange` = Writing is only triggered if the value has changed

7.2 Readmode

Using the remote function blocks as a client requires specifying the access mode to the peer objects. The variable `eReadMode` of type `E_BACnet_CommMode` specifies the access method.

`eAuto` = Automatic

`eCov` = Unconfirmed COV

`eCovC` = Confirmed COV

`eCovU` = Unsubscribed COV

`eRp` = Read Property

`eRpm` = RPM=Read Property Multiple

7.2.1 Automatic mode

The following description explains the implementation of the Automatic readmode. During startup in automatic mode important properties such as ProtocolServicesSupported, ApduSize are read from the peer device. Using these property data the read mode will be calculated using the following rules. It will also be calculated how many properties in one Rpm request are optimal to avoid segmentation.

7.2.1.1 Case 1: Devices with small APDU size

MS/TP or BACnet over LonTalk devices support only small APDU (Application Protocol Data Unit) sizes. MS/TP supports a maximum of 480 bytes, BACnet over LonTalk only up to 206 Bytes.

The maximum number of parallel requests is set to 1. This means only 1 request is sent and the response is awaited before continuing with the next request.

If the APDU size is less or equal to 480 bytes (e. g. MS/TP devices), then the ReadPropertyMultiple threshold will be set to 5. Change-of-Value will not be used in this case.

The number of properties per RPM request is limited 20 properties.

If the APDU size is less or equal to 206 bytes (BACnet over LonTalk devices), the number of properties per RPM request is limited 10 properties.

7.2.1.2 Case 2: BACnet/IP devices

BACnet/IP devices support an APDU size of up to 1476 bytes.

If the APDU size is greater than 480 bytes (e. g. 1476 for BACnet/IP), then the number of parallel requests is set to 50, this means there may be up to 50 requests awaiting the responses.

The ReadPropertyMultiple threshold will be set to 300.This means, if more than 300 properties are requested cyclically RPM will be used in available instead of COV.

7.2.1.3 ReadPropertyMultiple threshold and selected read service

If the total number of properties to be read in one cycle exceeds the ReadPropertyMultiple threshold and ReadPropertyMultiple is supported by the peer device, the readmode is set to ReadPropertyMultiple.

If the total number of properties to be read in one cycle is less or equal than the ReadPropertyMultiple threshold and ChangeOfValue is supported by the peer device, the readmode is set to ChangeOfValue.

If the total number of properties to be read in one cycle is less or equal than the ReadPropertyMultiple threshold and ChangeOfValue is not supported by the peer device, the readmode is set to ReadPropertyMultiple.

If the total number of properties to be read in one cycle is less or equal than the ReadPropertyMultiple threshold and ReadPropertyMultiple is not supported by the peer device, the readmode is set to ReadProperty.

7.2.2 Applying the Readmode to the entire peer device

The access mode can be specified for the entire client connection (i.e. this setting is applied to all objects of the peer device).

```
Client : FB_BACnet_Client :=
(eReadMode:=E_BACnet_CommMode.eCovU,bSuppCov:=TRUE,bSuppCovP:=TRUE,bSuppRpm:=TRUE,tReadCycleTime:=T#
2S550ms,tWriteCycleTime:=T#2S550ms);
```

7.2.3 Applying the Readmode to single objects

Devices from various vendors claim support for Change-of-Value, but not for all objects in a device. In this is the case trying to apply COV to all objects fails. Knowing which objects support COV (taken from the device manual or from other sources like the PICS = Protocol Implementation Conformance Statement document) allows to setup a COV access to those objects supporting this feature.

Example: Object AV:2 is requested using Readmode COVC (Confirmed COV).

```
Actual_Flow_Rate_feedback : FB_BACnetRM_AV :=
(Client:=Client,nObjectInstance:=2,eReadMode:=E_BACnet_CommMode.eCovC );
```

No fallback procedure is provided in case the peer device runs out of memory and is unable to accept further COV subscriptions. In this case the FB will enter an error state. In case you are uncertain, use RP/RPM polling instead of COV.

7.2.4 COV-Reporting

COV (Change-of-Value) reporting provides a functionality to automatically receive notifications in case of value changes. Normally only the Present_Value and Status_Flags properties are being notified. As a client, to receive COV-Notifications it is required to subscribe for COV for a period of time (even though infinite subscriptions are possible, it is good practice to use limited lifetimes only). If the subscription succeeds, a simple acknowledge is returned by the server. The current value is being reported as the initial value as soon as the device can provide it (normally within a few seconds).

The list of active COV subscriptions can be obtained from the property Active_COV_Subscriptions from the Device object.

In case the subscription is not repeated after the subscription lifetime expired, the subscription is silently removed and no COV messages are reported for this subscription anymore.

After the subscription is renewed the current value will be provided again as the initial value, this can be used as a heartbeat, especially for binary objects which do not change very often.

The number of subscriptions is not clearly specified in the BACnet standard. In addition, there are devices on the market which claim support for COV, but not for all objects. Or the number of subscriptions may be limited to just a few per object or per device. Other client processes like the BMS may use COV as well which may cause a BACnet device to run out of space for further subscriptions.

In case of BACnet error messages, especially issued by smaller MS/TP devices we recommend using a ReadProperty / ReadProperty Multiple polling instead of using COV.

7.3 Client POUs

Remote FBs for programming a BACnet client are named "FB_BACnetRM_" followed by the abbreviation given in the following table, e.g.
the FB_BACnetRM_AI represents a remote FB for accessing an analog input object in a peer device.

ACC	Accumulator	This object type represents accumulated (pulse) values.
AI	Analog Input	This object type represents physical analog input information, e.g. a sensor value.
AO	Analog Output	This object type represents physical analog output information, e.g. via a 0-10V output.
AV	Analog Value	This object type represents a (virtual) analog value information, e.g. a setpoint.
AV_Disp	Analog Value Display	This object type represents read-only analog value information.
AV_Setp	Analog Value Setpoint	This object type represents writable but not commandable analog value information, priority array and relinquish default are not provided.
AVG	Averaging	This object type represents an averaging object that provides statistical information.
BI	Binary Input	This object type represents a binary input information, e.g. the state of a lamp or a fuse.
BO	Binary Output	This object type represents a binary output information, e.g. via a switch.
BV	Binary Value	This object type represents a (virtual) binary value information, e.g. an error state.
BV_Disp	Binary Value Display	This object type represents read-only binary value information.
BV_Setp	Binary Value Setpoint	This object type represents a writable but not commandable binary value information, priority array and relinquish default are not provided.
CAL	Calendar	This object type represents calendar (date-based) information.
CMD	Command	This object type represents command information (scene information).
Device	Device	This object type represents the physical device. It contains information such as the local clock, vendor, model name and more.
EE	EventEnrollment	This object type is used to apply event monitoring in addition to intrinsic reporting, e.g. to implement warning limits.
ELOG	Eventlog	This object type represents an event log buffer, e.g. to store alarms locally.
File	File	This object type represents files, e.g. the current configuration or persistent data.
Group	Group	This object type represents a group of objects.
Loop	Control Loop	This object type represents control loops, e.g. a PI or PID loop.
Loop_Cnv	Control Loop	Like Loop object, additionally P, I and D and the respective units are provided.
MI	Multistate Input	This object type represents a physical multistate input information, e.g. a local operating modes switch.
MO	Multistate Output	This object type represents a physical multistate output information, e.g. an operating modes switch controlled by the PLC.
MV	Multistate Value	This object type represents a (virtual) multistate value information, e.g. a program parameter.
MV_Disp	Multistate Value Display	This object type represents read-only multistate value information.
MV_Setp	Multistate Value Setpoint	This object type represents writable but not commandable multistate value information, priority array and relinquish default are not provided.
NC	Notification Class	This object type represents an alarm class to notify recipients.
PC	Pulse Converter	This object type represents converted pulse information, e.g. energy consumption in kWh.
Prog	Program	This object type represents the PLC program.
SchedA	Schedule Analog	This object type represents a schedule with analog values.
SchedB	Schedule Binary	This object type represents a list of binary values.
SchedM	Schedule Multistate	This object type represents a schedule with multistate values.

TLM	Trendlog Multiple	This object type represents a trendlog object that supports multiple channels.
Tlog	Trendlog	This object type represents a trendlog object that supports a single channel.

7.4 FB Code

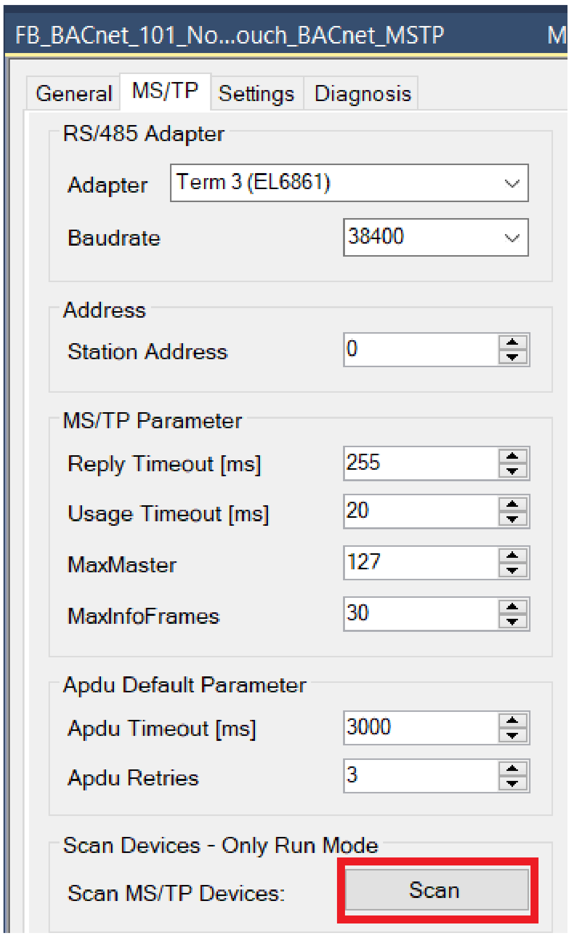
This function may be used to automatically generate the necessary function code for referencing BACnet clients in the PLC. The remote function blocks referring to the BACnet remote objects are automatically referenced in the code generation.

7.5 Calling the function FB Code

FB Code may be called from different menu items in the system manager.

7.5.1 Calling FB Code from the Scan dialog

Using the *Scan* function, BACnet devices are automatically detected using the BACnet services Who-Is / I-Am.


FB_BACnet_101_No...ouch_BACnet_MSTP

General | MS/TP | Settings | Diagnosis

RS/485 Adapter
Adapter: Term 3 (EL6861)
Baudrate: 38400

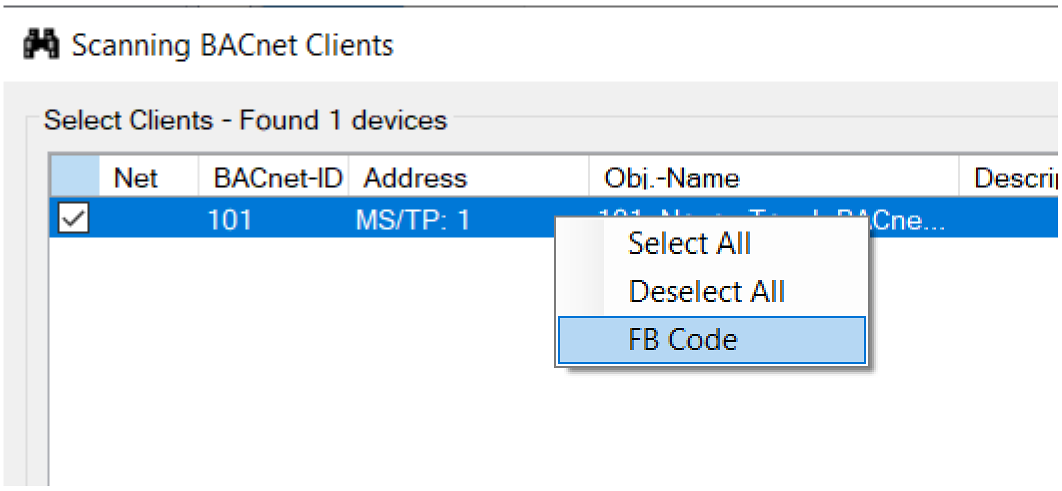
Address
Station Address: 0

MS/TP Parameter
Reply Timeout [ms]: 255
Usage Timeout [ms]: 20
MaxMaster: 127
MaxInfoFrames: 30

Apdu Default Parameter
Apdu Timeout [ms]: 3000
Apdu Retries: 3

Scan Devices - Only Run Mode
Scan MS/TP Devices: Scan

After scanning for devices, the dialog shows all available devices found in the network. Right click to a device and select FB Code to start the code generation. Make sure, a PLC project already exists.

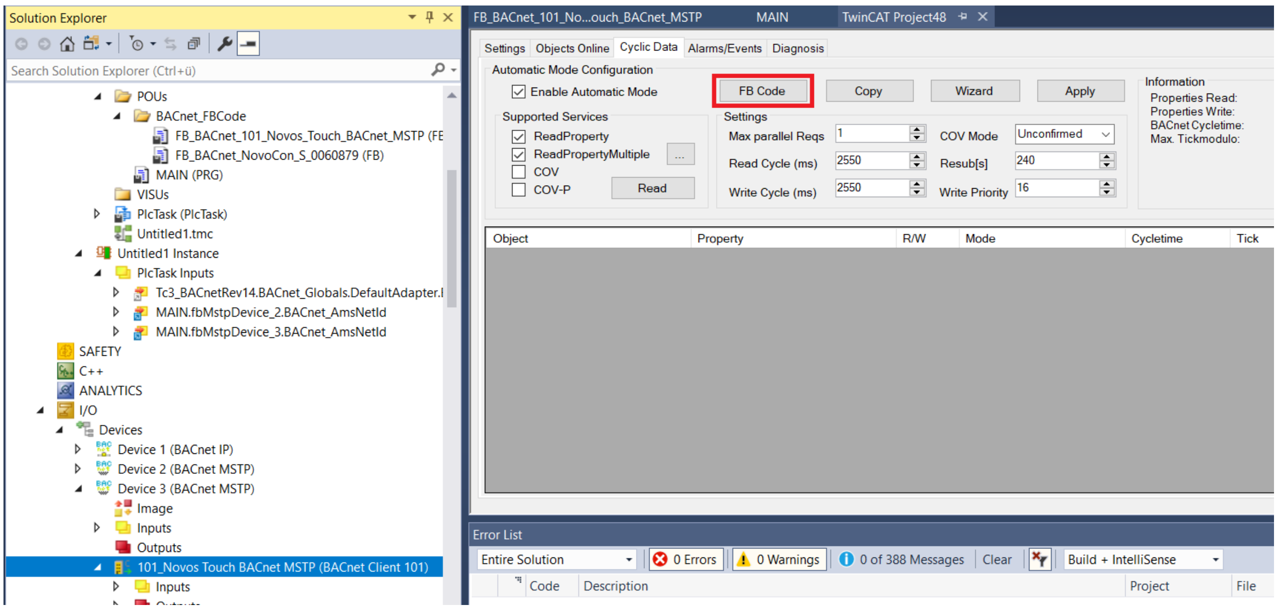


	Net	BACnet-ID	Address	Obj.-Name	Descri
✓		101	MS/TP: 1	101_N... BACne...	

7.5.2 Calling FB Code from the Cyclic Data dialog

If the device has been scanned and added to the System Manager as a BACnet client reference, the FB Code can be called via the *Cyclic Data* dialog.

Select the client device in the System Manager tree and select the *Cyclic Data* dialog.



Enable Automatic Mode: ✓; ReadProperty: ✓; ReadPropertyMultiple: ✓; COV: ☐; COV-P: ☐

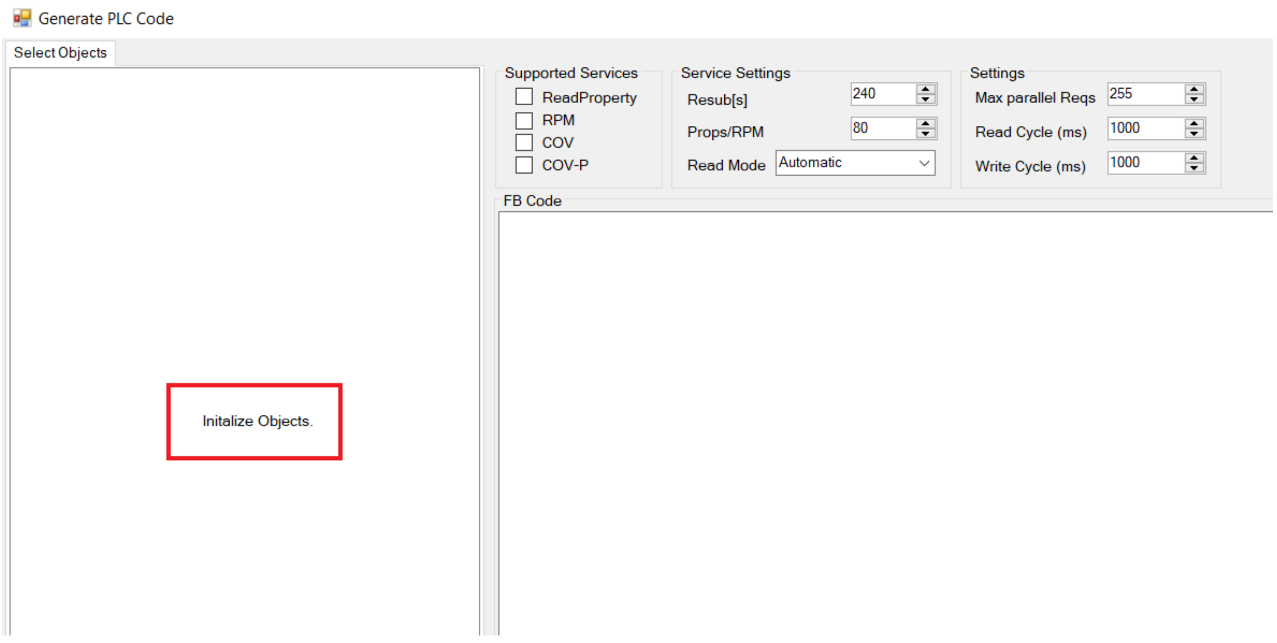
Max parallel Reqs	1	COV Mode	Unconfirmed
Read Cycle (ms)	2550	Resub[s]	240
Write Cycle (ms)	2550	Write Priority	16

7.6 The client FB Code dialog

After calling the function FB Code the dialog shown below opens the code generation window.

The BACnet objects contained in the selected device are shown in the tree in the left window.

> If using FB Code from the **Scan** dialog, searching for all objects in a device may take some time. This is shown by animated dots. Please be patient to read the entire device, especially when scanning slower MS/TP devices!


Generate PLC Code

Select Objects

Supported Services: [ ] ReadProperty, [ ] RPM, [ ] COV, [ ] COV-P

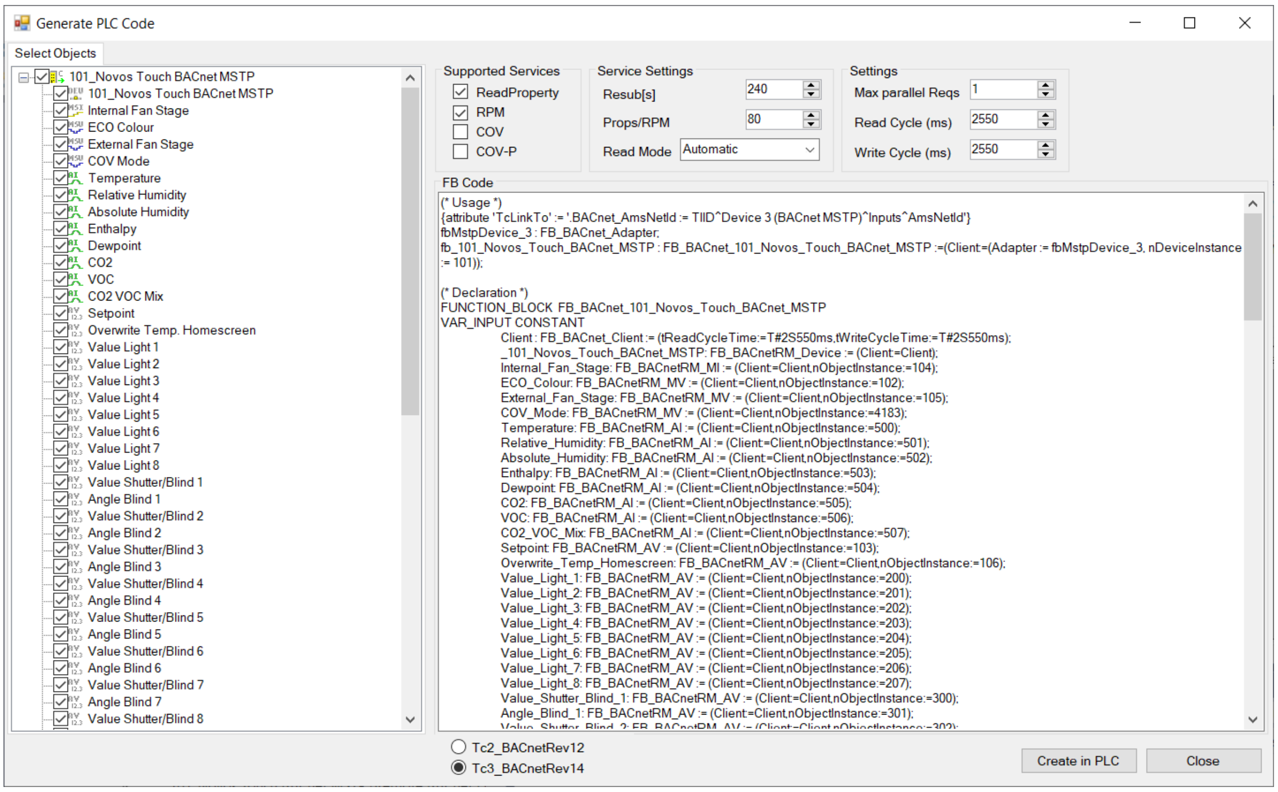
Service Settings: Resub[s] 240; Props/RPM 80; Read Mode Automatic

Settings: Max parallel Reqs 255; Read Cycle (ms) 1000; Write Cycle (ms) 1000

FB Code

Initalize Objects.

After scanning the entire device, the dialog displays all objects contained in the device.


Generate PLC Code

Select Objects

- [✓] 101_Novos Touch BACnet MSTP
  - [✓] 101_Novos Touch BACnet MSTP
  - [✓] Internal Fan Stage
  - [✓] ECO Colour
  - [✓] External Fan Stage
  - [✓] COV Mode
  - [✓] Temperature
  - [✓] Relative Humidity
  - [✓] Absolute Humidity
  - [✓] Enthalpy
  - [✓] Dewpoint
  - [✓] CO2
  - [✓] VOC
  - [✓] CO2 VOC Mix
  - [✓] Setpoint
  - [✓] Overwrite Temp. Homescreen
  - [✓] Value Light 1
  - [✓] Value Light 2
  - [✓] Value Light 3
  - [✓] Value Light 4
  - [✓] Value Light 5
  - [✓] Value Light 6
  - [✓] Value Light 7
  - [✓] Value Light 8
  - [✓] Value Shutter/Blind 1
  - [✓] Angle Blind 1
  - [✓] Value Shutter/Blind 2
  - [✓] Angle Blind 2
  - [✓] Value Shutter/Blind 3
  - [✓] Angle Blind 3
  - [✓] Value Shutter/Blind 4
  - [✓] Angle Blind 4
  - [✓] Value Shutter/Blind 5
  - [✓] Angle Blind 5
  - [✓] Value Shutter/Blind 6
  - [✓] Angle Blind 6
  - [✓] Value Shutter/Blind 7
  - [✓] Angle Blind 7
  - [✓] Value Shutter/Blind 8

Supported Services: [✓] ReadProperty, [✓] RPM, [ ] COV, [ ] COV-P

Service Settings: Resub[s] 240; Props/RPM 80; Read Mode Automatic

Settings: Max parallel Reqs 1; Read Cycle (ms) 2550; Write Cycle (ms) 2550

FB Code

```
(* Usage *)
{attribute 'TcLinkTo' := '.BACnet_AmsNetId := TIID^Device 3 (BACnet MSTP)^Inputs^AmsNetId'}
fbMstpDevice_3 : FB_BACnet_Adapter;
fb_101_Novos_Touch_BACnet_MSTP : FB_BACnet_101_Novos_Touch_BACnet_MSTP :=(Client:=(Adapter := fbMstpDevice_3, nDeviceInstance
:= 101));

(* Declaration *)
FUNCTION_BLOCK FB_BACnet_101_Novos_Touch_BACnet_MSTP
VAR_INPUT CONSTANT
        Client : FB_BACnet_Client := (tReadCycleTime:=T#2S550ms,tWriteCycleTime:=T#2S550ms);
        _101_Novos_Touch_BACnet_MSTP: FB_BACnetRM_Device := (Client:=Client);
        Internal_Fan_Stage: FB_BACnetRM_MI := (Client:=Client,nObjectInstance:=104);
        ECO_Colour: FB_BACnetRM_MV := (Client:=Client,nObjectInstance:=102);
        External_Fan_Stage: FB_BACnetRM_MV := (Client:=Client,nObjectInstance:=105);
        COV_Mode: FB_BACnetRM_MV := (Client:=Client,nObjectInstance:=4183);
        Temperature: FB_BACnetRM_AI := (Client:=Client,nObjectInstance:=500);
        Relative_Humidity: FB_BACnetRM_AI := (Client:=Client,nObjectInstance:=501);
        Absolute_Humidity: FB_BACnetRM_AI := (Client:=Client,nObjectInstance:=502);
        Enthalpy: FB_BACnetRM_AI := (Client:=Client,nObjectInstance:=503);
        Dewpoint: FB_BACnetRM_AI := (Client:=Client,nObjectInstance:=504);
        CO2: FB_BACnetRM_AI := (Client:=Client,nObjectInstance:=505);
        VOC: FB_BACnetRM_AI := (Client:=Client,nObjectInstance:=506);
        CO2_VOC_Mix: FB_BACnetRM_AI := (Client:=Client,nObjectInstance:=507);
        Setpoint: FB_BACnetRM_AV := (Client:=Client,nObjectInstance:=103);
        Overwrite_Temp_Homescreen: FB_BACnetRM_AV := (Client:=Client,nObjectInstance:=106);
        Value_Light_1: FB_BACnetRM_AV := (Client:=Client,nObjectInstance:=200);
        Value_Light_2: FB_BACnetRM_AV := (Client:=Client,nObjectInstance:=201);
        Value_Light_3: FB_BACnetRM_AV := (Client:=Client,nObjectInstance:=202);
        Value_Light_4: FB_BACnetRM_AV := (Client:=Client,nObjectInstance:=203);
        Value_Light_5: FB_BACnetRM_AV := (Client:=Client,nObjectInstance:=204);
        Value_Light_6: FB_BACnetRM_AV := (Client:=Client,nObjectInstance:=205);
        Value_Light_7: FB_BACnetRM_AV := (Client:=Client,nObjectInstance:=206);
        Value_Light_8: FB_BACnetRM_AV := (Client:=Client,nObjectInstance:=207);
        Value_Shutter_Blind_1: FB_BACnetRM_AV := (Client:=Client,nObjectInstance:=300);
        Angle_Blind_1: FB_BACnetRM_AV := (Client:=Client,nObjectInstance:=301);
```

( ) Tc2_BACnetRev12 [✓ Tc3_BACnetRev14]

7.6.1 The Select Objects window

This window shows the objects contained in the device. Objects which should not be included in the code generation may be disabled by using the checkboxes.

7.6.2 Supported Services

This window displays the supported services of the peer device.

ReadProperty

Cyclic polling is performed to determine the property values. Read Property reads only individual properties. Read Property Polling is therefore very slow due to its principle.

RPM = Read Property Multiple

RPM is a cyclic polling that polls multiple values at once to determine the property values. This process is faster than polling with Read Property. In some cases, the *PropsRPM* value must be lowered to a lower value, especially for devices with limited resources.

COV = Change of Value

This setting uses COV to poll the property values (Present Value and Status Flags).

Note: Some devices, especially MS/TP devices, may support COV only for some, not all, objects contained in the device. In this case, deselect COV from the supported services and, if necessary, add COV support in the PLC code only to the objects that support COV. In the data sheet of the BACnet device you will find the corresponding objects. If you are not sure, disable COV and use RP/RPM polling instead.

COV-P = Change of Value Property

With this service TwinCAT subscribes as a client to individual properties. Note: Not all BACnet devices support this service!

7.6.3 Service Settings

Resub (s)

This setting specifies the interval to resubscribe for Change-of-Value. This value also specifies the heartbeat (every subscription causes a value update).

Props/RPM

This setting specifies the number of properties per ReadPropertyMultiple request. Please note, some BACnet devices may not be able to handle higher quantities of properties per request. In this case this value may be decreased for a proper communication.

Read Mode

Automatic

Unconfirmed COV

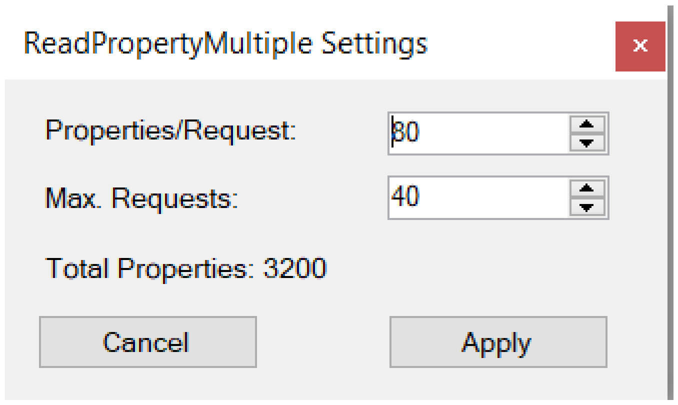
Confirmed COV

Unsubscribed COV

Read Property

RPM=Read Property Multiple

The calculation of the number of ReadPropertyMultiple requests is determined by the settings in the ReadProperty Multiple Settings dialog.


ReadPropertyMultiple Settings

Properties/Request: 80

Max. Requests: 40

Total Properties: 3200

The first item specifies the maximum number of properties requested in a single RPM-request.

The second item specifies the maximum number of RPM requests.

Please note: These settings are optimized for Beckhoff peer devices and may be reduced in case other devices cannot handle large amounts of requests or properties per request.

7.6.4 The Settings window

Max parallel Requests

This setting specifies how many requests may be sent before waiting for the response of the peer device. Please note that some devices may not support a large number of parallel requests. If you are not sure, use the value 1 and watch for possible timeout messages in the error log window.

Read Cycle (ms)

This value specifies the read request interval in milliseconds.

Write Cycle (ms)

This value specifies the write interval in milliseconds.

7.6.5 The FB Code window

This window displays the code generated from the object information of the selected device in structured text (ST).

This window is divided into three sections.

The *(* Usage *)* section shows how to create instances of this function block and adds a TcLinkTo... attribute in the case of MS/TP

Sample:

```
{attribute 'TcLinkTo' := '.BACnet_AmsNetId := TIID^Device 3 BACnet MSTP)^Inputs^AmsNetId'}
fbMstpDevice_3 : FB_BACnet_Adapter;
fb_101_Novos_Touch_BACnet_MSTP : FB_BACnet_101_Novos_Touch_BACnet_MSTP :=(Client:=(Adapter :=
fbMstpDevice_3, nDeviceInstance := 101));
```

The *(* Declaration *)* section shows the variables of the function blocks used.

Sample:

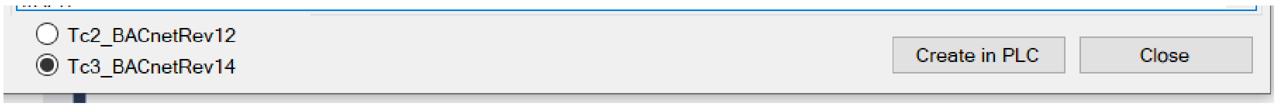
```
FUNCTION_BLOCK FB_BACnet_101_Novos_Touch_BACnet_MSTP
VAR_INPUT CONSTANT
    Client : FB_BACnet_Client:=
            (tReadCycleTime:=T#2S550ms,tWriteCycleTime:=T#2S550ms);
    _101_Novos_Touch_BACnet_MSTP: FB_BACnetRM_Device:=
            (Client:=Client);
    Internal_Fan_Stage: FB_BACnetRM_MI :=
            (Client:=Client,nObjectInstance:=104);
    ECO_Colour: FB_BACnetRM_MV :=
            (Client:=Client,nObjectInstance:=102);
    External_Fan_Stage: FB_BACnetRM_MV :=
            (Client:=Client,nObjectInstance:=105);
END_VAR
```

The *(* Code *)* section shows the instance calls of the function blocks.

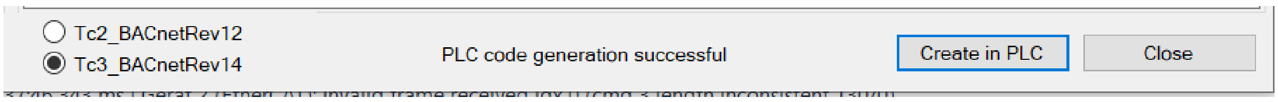
Sample:

```
Client();
_101_Novos_Touch_BACnet_MSTP();
Internal_Fan_Stage();
ECO_Colour();
External_Fan_Stage();
```

The footer of this dialog contains a switch that influences the code generation. It is possible to generate the code for BACnet revision 12 using the automapping comments or for BACnet revision 14 using the PLC code.



Selecting Create in PLC starts the code generation. After successful creation, a message is displayed in the footer. Click Close to close the dialog and return to the TwinCAT System Manager.



PLC code generation successful

After this step it is only necessary to place the function block call (e.g. in the POU MAIN) as shown above.

The contents of the FB Code window can also be copied to the clipboard for further editing.

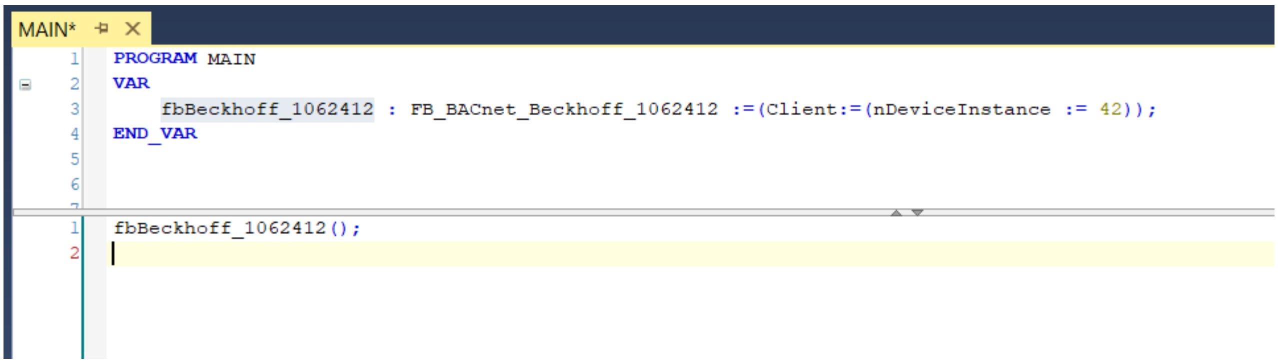
7.6.6 Using the FB created by FB_Code

The code generated by `FB_Code` is not called automatically. The first lines in the variable declaration of the generated FB contain the usage information.

Sample BACnet IP

```
(* Usage :
fbBeckhoff_1062412 : FB_BACnet_Beckhoff_1062412 :=(Client:=(nDeviceInstance := 42));
*)
```

Cyclic call, e.g. in POU Main



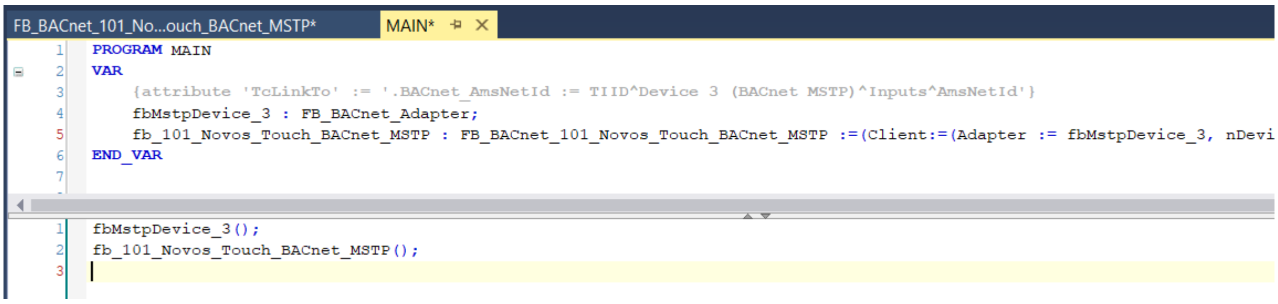
```
PROGRAM MAIN
VAR
    fbBeckhoff_1062412 : FB_BACnet_Beckhoff_1062412 :=(Client:=(nDeviceInstance := 42));
END_VAR
```

```
fbBeckhoff_1062412();
```

Sample BACnet MS/TP

```
(* Usage :
{attribute 'TcLinkTo' := '.BACnet_AmsNetId := TIID^Device 3 (BACnet MSTP)^Inputs^AmsNetId'}
fbMstpDevice_3 : FB_BACnet_Adapter;
fb_101_Novos_Touch_BACnet_MSTP : FB_BACnet_101_Novos_Touch_BACnet_MSTP :=(Client:=(Adapter :=
fbMstpDevice_3, nDeviceInstance := 101));
*)
```

Cyclic call, e.g. in POU Main



```
PROGRAM MAIN
VAR
    {attribute 'TcLinkTo' := '.BACnet_AmsNetId := TIID^Device 3 (BACnet MSTP)^Inputs^AmsNetId'}
    fbMstpDevice_3 : FB_BACnet_Adapter;
    fb_101_Novos_Touch_BACnet_MSTP : FB_BACnet_101_Novos_Touch_BACnet_MSTP :=(Client:=(Adapter := fbMstpDevice_3, nDevi
END_VAR
```

```
fbMstpDevice_3();
fb_101_Novos_Touch_BACnet_MSTP();
```

Please note that the MS/TP adapter FB must be called cyclically in addition to the FB representing the MS/TP device.

7.6.7 Using FB_Code to create device templates

If there are several identical devices, `FB_Code` can be used to create a template. Multiple instances of the generated FB can be accessed via the device instance number of the peer device.

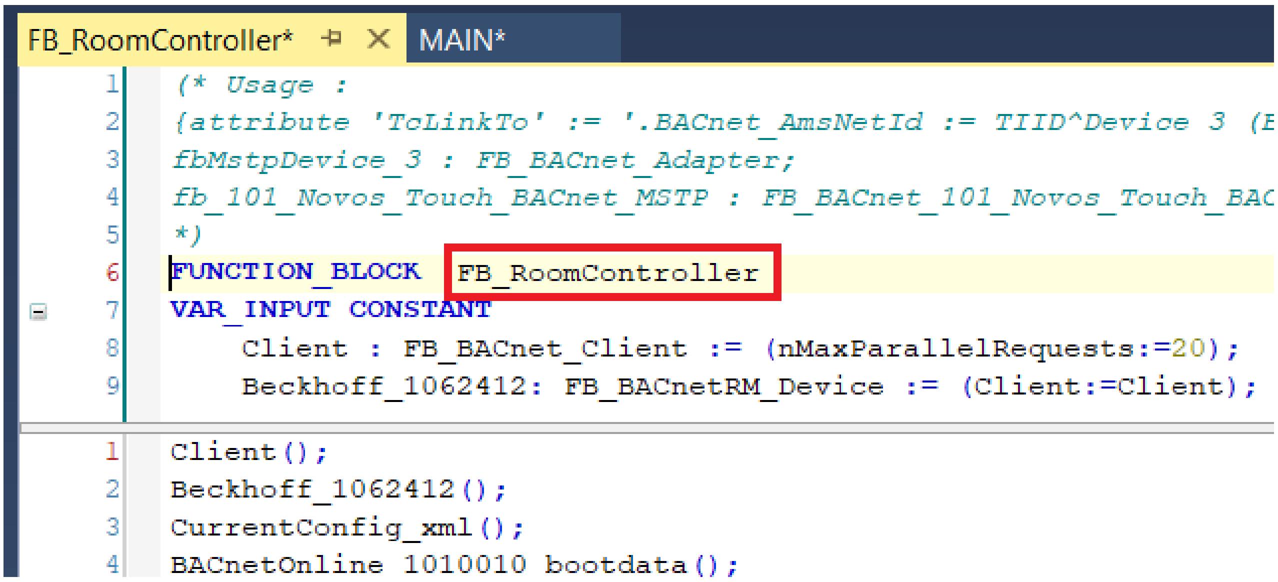
In the following sample, the generated FB has been renamed to represent a more general FB implementation. It is recommended to use the refactoring function of TwinCAT to avoid naming conflicts.

Three instances are declared in MAIN and called cyclically.

Please note that you call the adapter FB only once!

Sample:

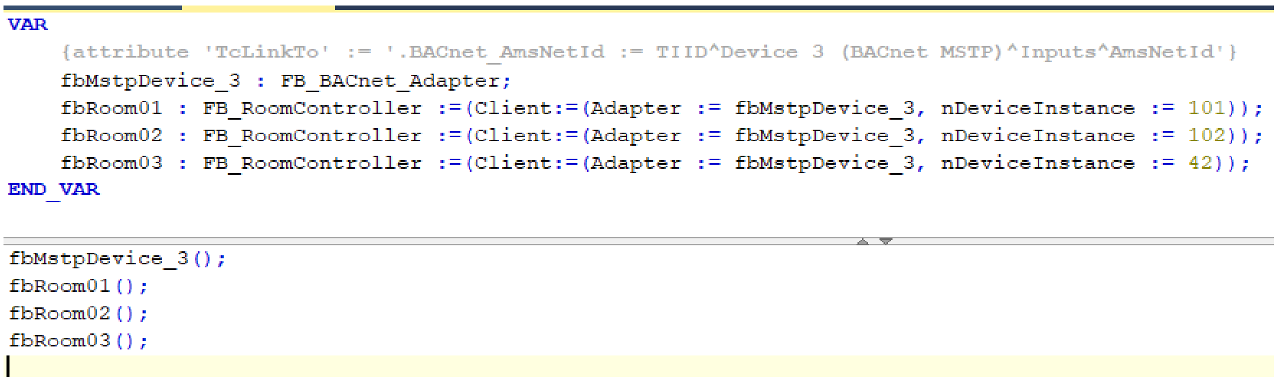
Rename `FB_Code`



```
(* Usage :
{attribute 'TcLinkTo' := '.BACnet_AmsNetId := TIID^Device 3 (E
fbMstpDevice_3 : FB_BACnet_Adapter;
fb_101_Novos_Touch_BACnet_MSTP : FB_BACnet_101_Novos_Touch_BAC
*)
FUNCTION_BLOCK FB_RoomController
VAR_INPUT CONSTANT
    Client : FB_BACnet_Client := (nMaxParallelRequests:=20);
    Beckhoff_1062412: FB_BACnetRM_Device := (Client:=Client);
```

```
Client();
Beckhoff_1062412();
CurrentConfig_xml();
BACnetOnline_1010010_bootdata();
```

Cyclic calls in MAIN:



```
VAR
    {attribute 'TcLinkTo' := '.BACnet_AmsNetId := TIID^Device 3 (BACnet MSTP)^Inputs^AmsNetId'}
    fbMstpDevice_3 : FB_BACnet_Adapter;
    fbRoom01 : FB_RoomController :=(Client:=(Adapter := fbMstpDevice_3, nDeviceInstance := 101));
    fbRoom02 : FB_RoomController :=(Client:=(Adapter := fbMstpDevice_3, nDeviceInstance := 102));
    fbRoom03 : FB_RoomController :=(Client:=(Adapter := fbMstpDevice_3, nDeviceInstance := 42));
END_VAR
```

```
fbMstpDevice_3();
fbRoom01();
fbRoom02();
fbRoom03();
```

7.7 Client variables

The client FBs provide various information about the status of the peer connection.

Adapter:

eDevState: Describes the state machine in the connection phase. Should be eComplete if all steps were successful.

bEthLink: TRUE if the connection has been established.

bGateway: TRUE if the IP address setting contains gateway information (IT router).

_bHasStarted: TRUE if the state machine is started.

_nUpdateCount: This value is incremented in each cycle.

Client:

bAutoResetObjError: If set to TRUE, the client state machine is automatically reset in case of communication interruption.

bSuppRpm: TRUE, if the peer device supports ReadPropertyMultiple.

bSuppCov: TRUE means that the peer device supports ChangeOfValue.

bSuppCovP: TRUE means that the peer device supports ChangeOfValueProperty.

bReady: The FB is initialized and ready to request data from the peer device.

bConnected: TRUE means that a connection to the peer device has been successfully established.

If the client does not use automatic mode, these variables determine which services are used to retrieve the values: bSuppRpm, bSuppCov, bSuppCovP

7.8 Remote schedule objects

BACnet schedule objects do not have a specific data type. The data type used for the schedule is determined by the Weekly-Schedule, Exception-Schedule, Schedule-Default and ListofObjectPropertyReferences properties. To reference external schedule objects, the corresponding function block must be selected manually (FB_BACnetRM_SchedA for analog schedule, FB_BACnetRM_SchedB for binary schedule, FB_BACnetRM_SchedM for multistate schedule).

7.9 Acyclic read

To read properties from the object of a peer device, the library provides two functions for acyclic reading:

FB_BACnetRM_ReadProperty

FB_BACnetRM_ReadPropertyEx

The BACnet object of the peer device is referenced by a pointer to the FB remote instance (variable iObject). Compared to the basic function block FB_BACnetRM_ReadProperty, the function block FB_BACnetRM_ReadPropertyEx provides two additional variables for specifying the object type and instance number (variables eObjType and nObjInst).

7.9.1 Example FB_BACnetRM_ReadProperty

This sample shows how to use the FB_BACnetRM_ReadProperty function block to read the value of the high_limit property from the analog input object, instance number 1 in the peer device instance number 42. Please note that the object referenced by iObject must be called cyclically.

```
VAR
    fbClient : FB_BACnet_Client := (nDeviceInstance :=
2,tReadCycleTime:=T#10S,nMaxParallelRequests:=255);
    fbDevice : FB_BACnetRM_Device := (Client:=fbClient);
    fbAI  : FB_BACnetRM_AI  := (Client:=fbClient,nObjectInstance:=1);

    fbRead : FB_BACnetRM_ReadProperty := (Client := fbClient);
    bReadHighLimit : BOOL;
    fHighLimit : REAL;
END_VAR
------------------------------------------------------------------
fbClient();
fbDevice();
fbAI();

// Read HighLimit using FB_BACnetRM_ReadProperty
fbRead.bExecute := bReadHighLimit;
IF fbRead.bExecute THEN
    bReadHighLimit := FALSE;
    fbRead.pData:= ADR( fHighLimit );
    fbRead.nData:= SIZEOF( fHighLimit );
    fbRead.ePropID:= E_BACnet_PropertyIdentifier.PropHighLimit;
    fbRead.iObject := fbAI;
END_IF
fbRead();
```

7.9.2 Example FB_BACnetRM_ReadPropertyEx

The following example shows the use of the `FB_BACnetRM_ReadPropertyEx` function block.

```
VAR
    fbClient : FB_BACnet_Client := (nDeviceInstance :=
42,tReadCycleTime:=T#10S,nMaxParallelRequests:=255);
    fbDevice : FB_BACnetRM_Device := (Client:=fbClient);

    fbReadEx : FB_BACnetRM_ReadPropertyEx := (Client := fbClient);
    bReadLowLimitEx : BOOL;
    fLowLimitEx : REAL;
END_VAR
------------------------------------------------------------------
fbClient();
fbDevice();

// Read LowLimit using FB_BACnetRM_ReadPropertyEx
fbReadEx.bExecute := bReadLowLimitEx;
IF fbReadEx.bExecute THEN
    bReadLowLimitEx := FALSE;
    fbReadEx.pData:= ADR( fLowLimitEx );
    fbReadEx.nData:= SIZEOF( fLowLimitEx );
    fbReadEx.ePropID:= E_BACnet_PropertyIdentifier.PropLowLimit;
    fbReadEx.nObjInst:= 1;
    fbReadEx.eObjType:= E_BACnet_ObjectType.ObjAnalogInput;
END_IF
fbReadEx();
```

7.10 Acyclic write

To write properties to the object of a peer device, the library provides two functions for acyclic writing:

`FB_BACnetRM_WriteProperty`

`FB_BACnetRM_WritePropertyEx`

The BACnet object of the peer device is referenced by a pointer to the FB remote instance (variable iObject). Compared to the basic function block `FB_BACnetRM_WriteProperty`, the function block `FB_BACnetRM_WritePropertyEx` provides two additional variables for specifying the object type and instance number (variables `eObjType` and `nObjInst`).

7.10.1 Example FB_BACnetRM_WriteProperty

This example shows how to use the function block FB_BACnetRM_WriteProperty to write the value of the out_of_service property of the object binary output, instance number 0 in the peer device instance number 42. Please note that the object referenced by iObject must be called cyclically.

```
VAR
    fbClient : FB_BACnet_Client := (nDeviceInstance :=
42,tReadCycleTime:=T#10S,nMaxParallelRequests:=255);
    fbDevice : FB_BACnetRM_Device := (Client:=fbClient);
    fbBO : FB_BACnetRM_BO := (Client:=fbClient,nObjectInstance:=0);
    fbWrite : FB_BACnetRM_WriteProperty := (Client := fbClient);
    bWriteOoS : BOOL;
    bOutofService : BOOL;
END_VAR
------------------------------------------------------------------
fbClient();
fbDevice();
fbBO();

// Write OutOfService using FB_BACnetRM_WriteProperty
fbWrite.bExecute := bWriteOoS;
IF fbWrite.bExecute THEN
    bWriteOoS  := FALSE;
    fbWrite.pData:= ADR( bOutOfService );
    fbWrite.nData:= SIZEOF( bOutOfService );
    fbWrite.ePropID:= E_BACnet_PropertyIdentifier.PropOutOfService;
    fbWrite.iObject := fbBO;
END_IF
fbWrite();
```

7.10.2 Example FB_BACnetRM_WritePropertyEx

The following example shows the usage of function block FB_BACnetRM_WritePropertyEx.

```
VAR
    fbClient : FB_BACnet_Client := (nDeviceInstance :=
42,tReadCycleTime:=T#10S,nMaxParallelRequests:=255);
    fbDevice : FB_BACnetRM_Device := (Client:=fbClient);

    fbWriteEx : FB_BACnetRM_WritePropertyEx := (Client := fbClient);
    bWriteOoSEx : BOOL;
    bOutofServiceEx : BOOL;
END_VAR
------------------------------------------------------------------
fbClient();
fbDevice();

// Write OutOfService using FB_BACnetRM_WritePropertyEx
fbWriteEx.bExecute := bWriteOoSEx;
IF fbWriteEx.bExecute THEN
    bWriteOoSEx := FALSE;
    fbWriteEx.pData:= ADR( bOutOfServiceEx );
    fbWriteEx.nData:= SIZEOF( bOutOfServiceEx );
    fbWriteEx.ePropID:= E_BACnet_PropertyIdentifier.PropOutOfService;
    fbWriteEx.nObjInst:= 1;
    fbWriteEx.eObjType:= E_BACnet_ObjectType.ObjBinaryOutput;
END_IF
fbWriteEx();
```

7.11 Monitoring a client connection

To monitor a client connection, it is necessary to call the FB_BACnetRM_Device function block cyclically.

The current status of the connection can be taken from the FB_BACnet_Client function block.

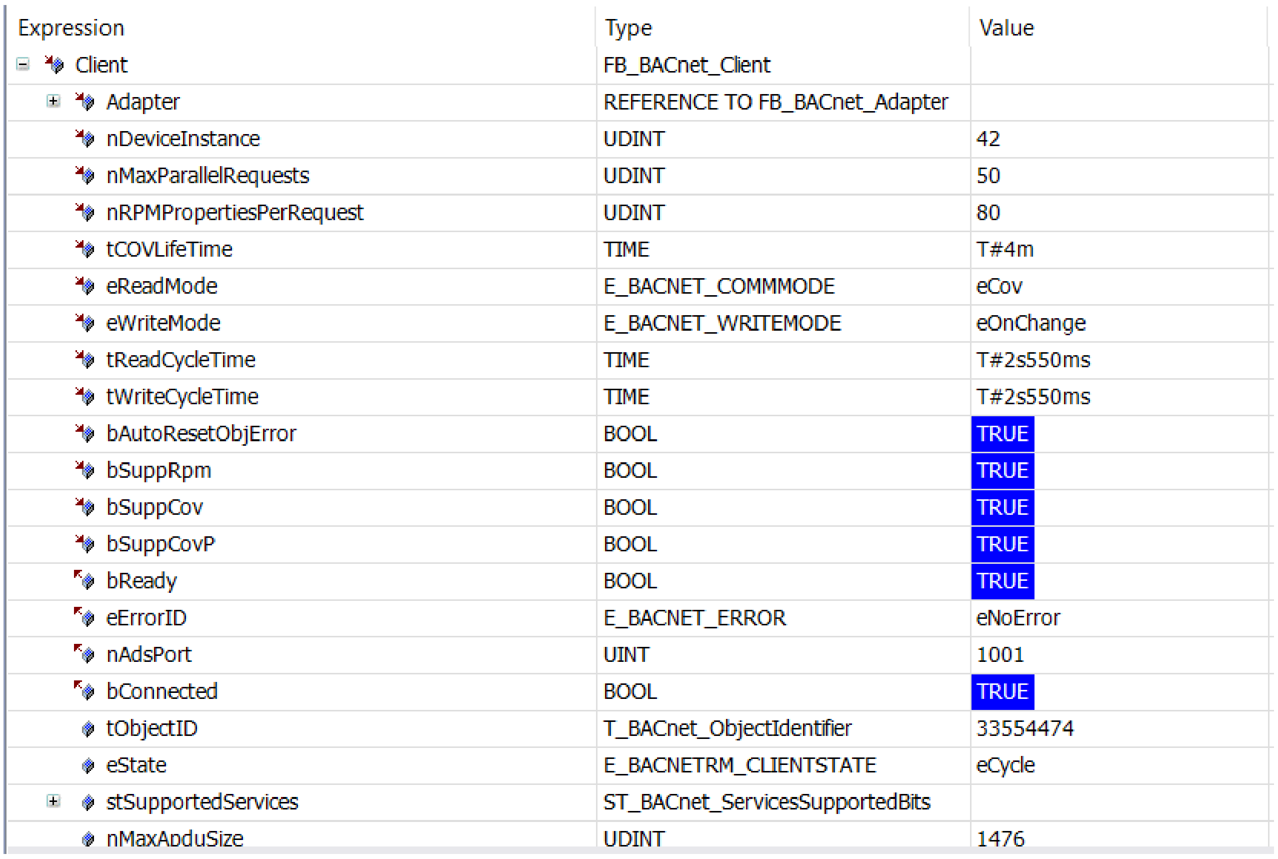
If the connection is interrupted, e.g. by disconnecting the peer device or by a defective network cable, the BACnet supplement attempts to re-establish the connection. If after these attempts (variable nErrorCnt in function block FB_BACnetRM_Device) the connection is still interrupted, it is assumed that the device or the network connection is no longer present. This process may take 30 seconds or longer to safely detect the interrupted connection.

In this case the variable eSysState of the FB_BACnetRM_Device takes the value eNoCommunication and bOperational is set to FALSE. The variables bReady and bConnected of the function block FB_BACnet_Client are set to FALSE, the variable eState takes the value eInit.

The BACnet supplement continues to attempt to connect to the peer device. Once the connection is re-established, the state machine behaves as it did when it was first started.

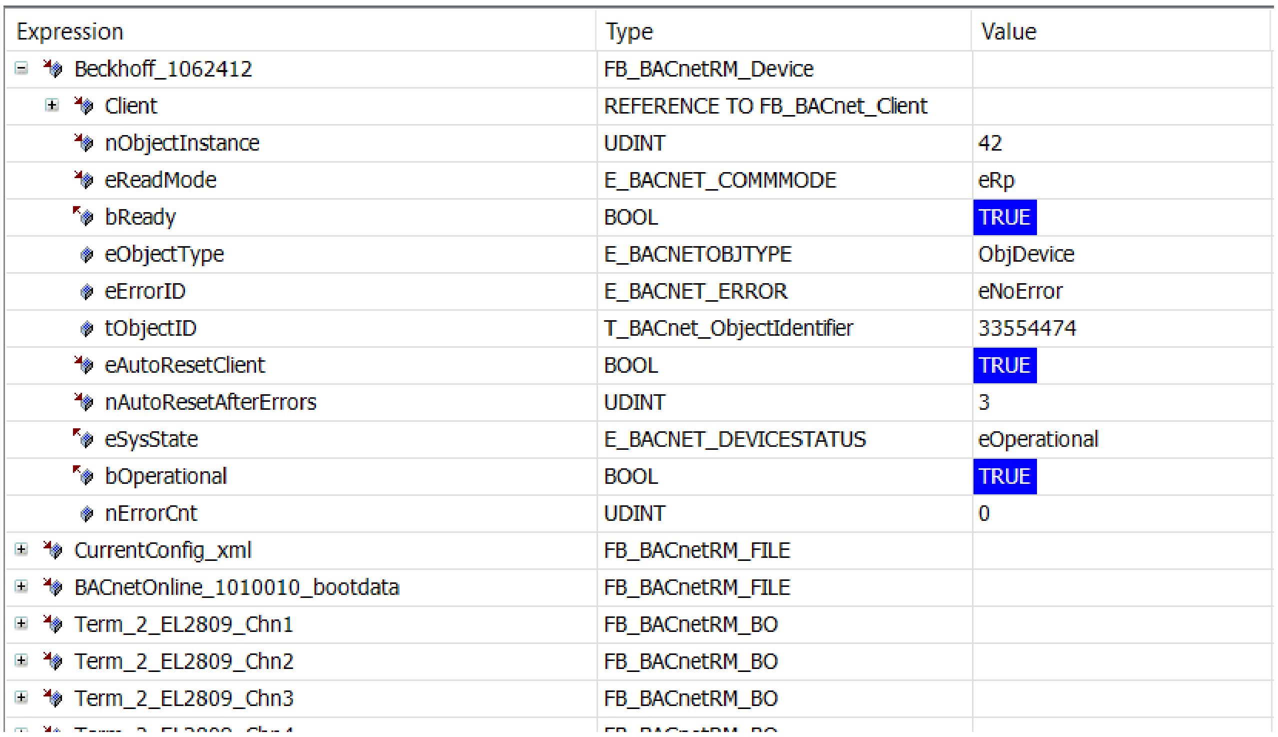
7.11.1 Example: Connection is successfully established

FB_BACnet_Client



Expression	Type	Value
Client	FB_BACnet_Client	
Adapter	REFERENCE TO FB_BACnet_Adapter	
nDeviceInstance	UDINT	42
nMaxParallelRequests	UDINT	50
nRPMPropertiesPerRequest	UDINT	80
tCOVLifeTime	TIME	T#4m
eReadMode	E_BACNET_COMMMODE	eCov
eWriteMode	E_BACNET_WRITEMODE	eOnChange
tReadCycleTime	TIME	T#2s550ms
tWriteCycleTime	TIME	T#2s550ms
bAutoResetObjError	BOOL	TRUE
bSuppRpm	BOOL	TRUE
bSuppCov	BOOL	TRUE
bSuppCovP	BOOL	TRUE
bReady	BOOL	TRUE
eErrorID	E_BACNET_ERROR	eNoError
nAdsPort	UINT	1001
bConnected	BOOL	TRUE
tObjectID	T_BACnet_ObjectIdentifier	33554474
eState	E_BACNETRM_CLIENTSTATE	eCycle
stSupportedServices	ST_BACnet_ServicesSupportedBits	
nMaxApduSize	UDINT	1476

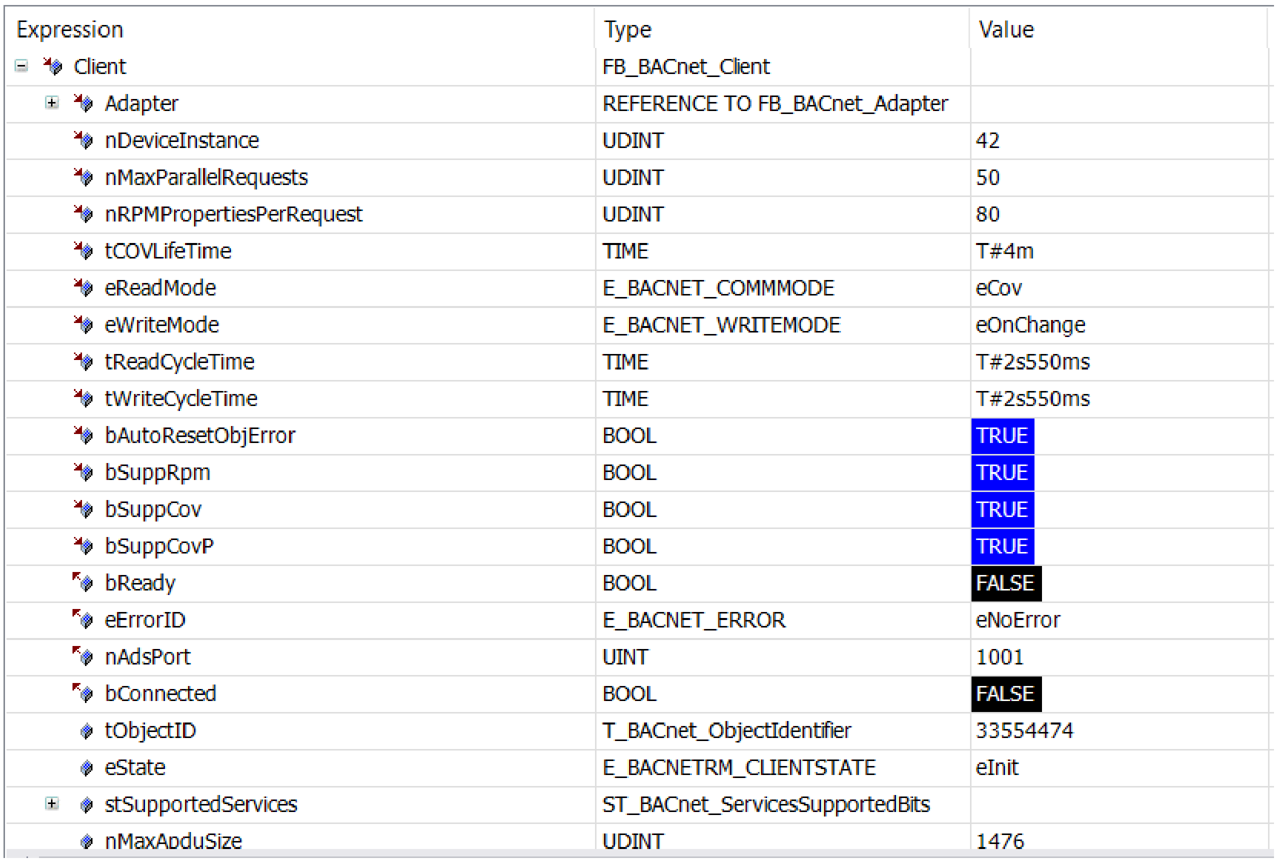
FB_BACnetRM_Device



Expression	Type	Value
Beckhoff_1062412	FB_BACnetRM_Device	
Client	REFERENCE TO FB_BACnet_Client	
nObjectInstance	UDINT	42
eReadMode	E_BACNET_COMMMODE	eRp
bReady	BOOL	TRUE
eObjectType	E_BACNETOBJTYPE	ObjDevice
eErrorID	E_BACNET_ERROR	eNoError
tObjectID	T_BACnet_ObjectIdentifier	33554474
eAutoResetClient	BOOL	TRUE
nAutoResetAfterErrors	UDINT	3
eSysState	E_BACNET_DEVICESTATUS	eOperational
bOperational	BOOL	TRUE
nErrorCnt	UDINT	0
CurrentConfig_xml	FB_BACnetRM_FILE	
BACnetOnline_1010010_bootdata	FB_BACnetRM_FILE	
Term_2_EL2809_Chn1	FB_BACnetRM_BO	
Term_2_EL2809_Chn2	FB_BACnetRM_BO	
Term_2_EL2809_Chn3	FB_BACnetRM_BO	

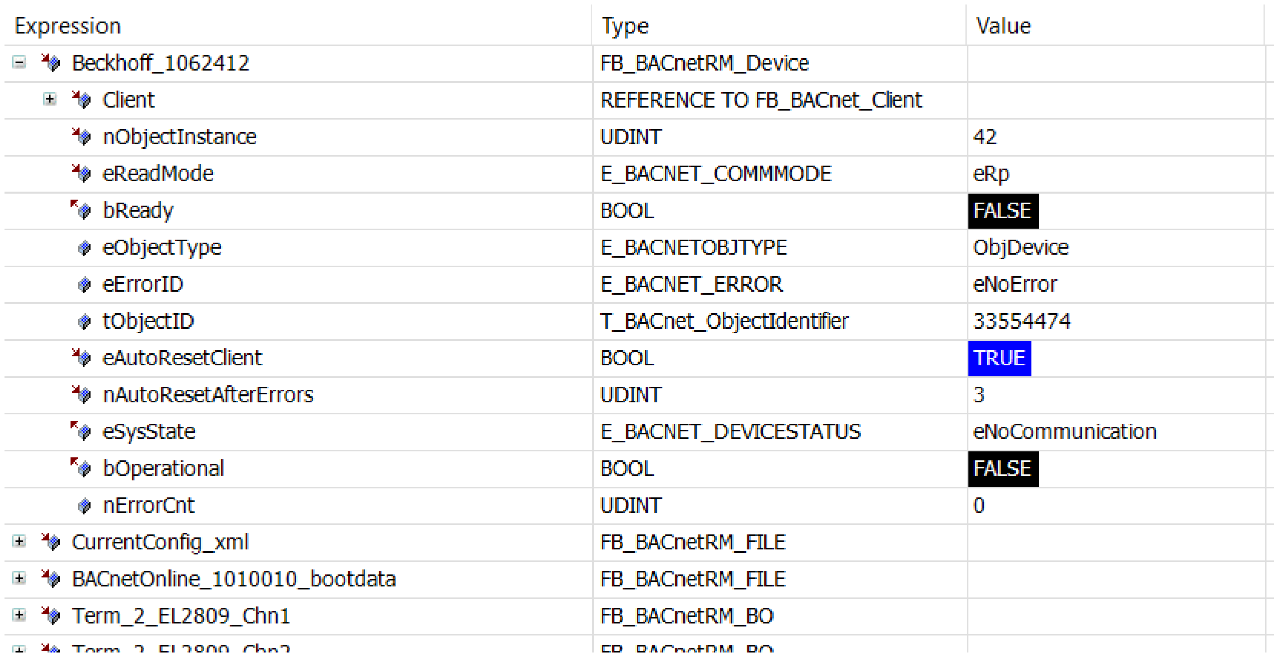
7.11.2 Example: Connection is interrupted

FB_BACnet_Client



Expression	Type	Value
Client	FB_BACnet_Client	
Adapter	REFERENCE TO FB_BACnet_Adapter	
nDeviceInstance	UDINT	42
nMaxParallelRequests	UDINT	50
nRPMPropertiesPerRequest	UDINT	80
tCOVLifeTime	TIME	T#4m
eReadMode	E_BACNET_COMMMODE	eCov
eWriteMode	E_BACNET_WRITEMODE	eOnChange
tReadCycleTime	TIME	T#2s550ms
tWriteCycleTime	TIME	T#2s550ms
bAutoResetObjError	BOOL	TRUE
bSuppRpm	BOOL	TRUE
bSuppCov	BOOL	TRUE
bSuppCovP	BOOL	TRUE
bReady	BOOL	FALSE
eErrorID	E_BACNET_ERROR	eNoError
nAdsPort	UINT	1001
bConnected	BOOL	FALSE
tObjectID	T_BACnet_ObjectIdentifier	33554474
eState	E_BACNETRM_CLIENTSTATE	eInit
stSupportedServices	ST_BACnet_ServicesSupportedBits	
nMaxApduSize	UDINT	1476

FB_BACnetRM_Device



Expression	Type	Value
Beckhoff_1062412	FB_BACnetRM_Device	
Client	REFERENCE TO FB_BACnet_Client	
nObjectInstance	UDINT	42
eReadMode	E_BACNET_COMMMODE	eRp
bReady	BOOL	FALSE
eObjectType	E_BACNETOBJTYPE	ObjDevice
eErrorID	E_BACNET_ERROR	eNoError
tObjectID	T_BACnet_ObjectIdentifier	33554474
eAutoResetClient	BOOL	TRUE
nAutoResetAfterErrors	UDINT	3
eSysState	E_BACNET_DEVICESTATUS	eNoCommunication
bOperational	BOOL	FALSE
nErrorCnt	UDINT	0
CurrentConfig_xml	FB_BACnetRM_FILE	
BACnetOnline_1010010_bootdata	FB_BACnetRM_FILE	
Term_2_EL2809_Chn1	FB_BACnetRM_BO	

7.12 Use of ReadPropertyMultiple

If several properties are to be read from a connected BACnet device, it is recommended to use the service **R**ead**P**roperty**M**ultiple (RPM). In contrast to the use of the ReadProperty service, which can only read individual properties, ReadPropertyMultiple combines several queries into one telegram and therefore works considerably more efficiently.

The following example shows the use of RPM. However, it requires that connected devices support this service. The three read commands shown in this example are combined into a single RPM.

Variables

```
    {attribute 'TcLinkTo' := '.BACnet_AmsNetId := TIID^Device 1 (BACnet MSTP)^Inputs^AmsNetId'}
    fbMstpDevice_1 : FB_BACnet_Adapter;

    fbClient : FB_BACnet_Client := (Adapter := fbMstpDevice_1,nDeviceInstance := 116005,
tReadCycleTime:=T#10S, nMaxParallelRequests:=20);
    fbDevice : FB_BACnetRM_Device := (Client:=fbClient);

    fbTon : TON;

    fbReadEx1 : FB_BACnetRM_ReadPropertyEx := (Client := fbClient);
    fbReadEx2 : FB_BACnetRM_ReadPropertyEx := (Client := fbClient);
    fbReadEx3 : FB_BACnetRM_ReadPropertyEx := (Client := fbClient);

    bRead : BOOL;

    fPVSupplyTemp : REAL;
    fPVReturnTemp : REAL;
    fDesignFlow : REAL;
```

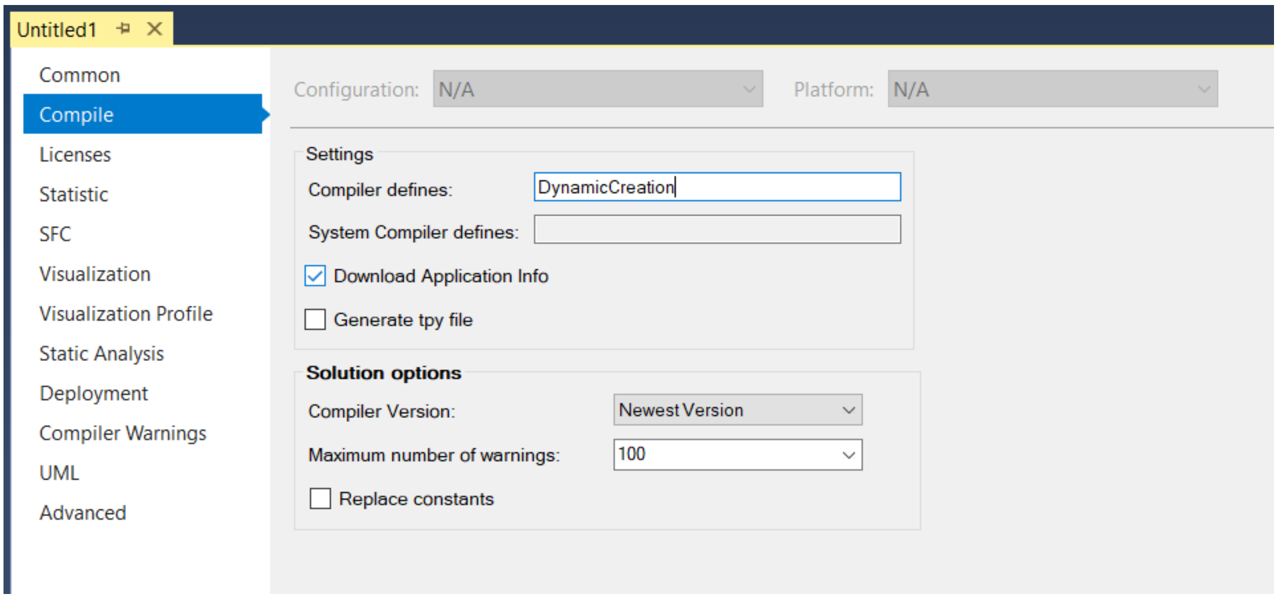
Code

```
fbMstpDevice_1();
fbClient();
fbDevice();

fbTon( IN:= NOT fbTon.Q, PT:= T#5S );
IF fbTon.Q THEN
    bRead := TRUE;
END_IF

fbReadEx1.bExecute := bRead;
fbReadEx2.bExecute := bRead;
fbReadEx3.bExecute := bRead;

IF bRead THEN
    bRead := FALSE;

    fbReadEx1.pData:= ADR( fPVSupplyTemp );
    fbReadEx1.nData:= SIZEOF( fPVSupplyTemp );
    fbReadEx1.ePropID:= E_BACnet_PropertyIdentifier.PropPresentValue;
    fbReadEx1.nObjInst:= 1;
    fbReadEx1.eObjType:= E_BACnet_ObjectType.ObjAnalogInput;

    fbReadEx2.pData:= ADR( fPVReturnTemp );
    fbReadEx2.nData:= SIZEOF( fPVReturnTemp );
    fbReadEx2.ePropID:= E_BACnet_PropertyIdentifier.PropPresentValue;
    fbReadEx2.nObjInst:= 2;
    fbReadEx2.eObjType:= E_BACnet_ObjectType.ObjAnalogInput;

    fbReadEx3.pData:= ADR( fDesignFlow );
    fbReadEx3.nData:= SIZEOF( fDesignFlow );
    fbReadEx3.ePropID:= E_BACnet_PropertyIdentifier.PropPresentValue;
    fbReadEx3.nObjInst:= 0;
    fbReadEx3.eObjType:= E_BACnet_ObjectType.ObjAnalogValue;
END_IF

fbReadEx1();
fbReadEx2();
fbReadEx3();
```

8 Dynamic Object Manager

Using the Dynamic Object Manager allows to create or delete BACnet objects at runtime.

To use this feature, the pragma “*DynamicCreation*” needs to be set in the compiler settings as shown below:



Configuration: N/A
Platform: N/A

Settings
Compiler defines: DynamicCreation
System Compiler defines:
☑ Download Application Info
☐ Generate tpy file

Solution options
Compiler Version: Newest Version
Maximum number of warnings: 100
☐ Replace constants

8.1 FB_BACnet_DynObjectManager

The `FB_BACnet_DynObjectManager` provides the ability to dynamically create or delete objects at runtime. Typical use cases are local visualizations or the creation of BACnet objects based on configuration files.

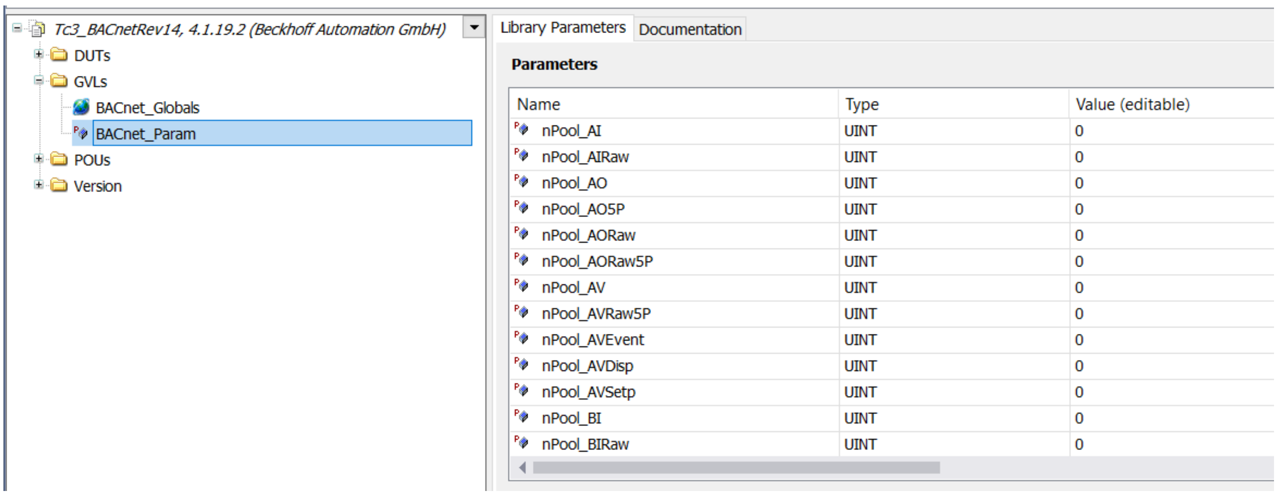
It is possible to use the Dynamic Object Manager together with statically created objects.
However, please note that the object instance numbers must be unique!

8.2 Cyclic calls

The Dynamic Object Manager and all dynamically created objects must be called cyclically (like all other objects once and only once per cycle). Setting the `bCycleObjects := TRUE` flag enables cyclic calls of the created objects. In this case, no further cyclic calls may be made (except calling the instance of `FB_BACnet_DynObjectManager`).

8.3 Predefined object pool

Dynamic objects are created with the `__NEW` operator. If the number of objects to be created is known or can be estimated at the time of the first PLC start, the library provides a predefined object pool. This is more memory efficient than using the `__NEW` operator. The elements named `nPool_XX` in the `BACnet_Param` section of the library GVLs can be set up to save resources.



Parameters

Name	Type	Value (editable)
nPool_AI	UINT	0
nPool_AIRaw	UINT	0
nPool_AO	UINT	0
nPool_AO5P	UINT	0
nPool_AORaw	UINT	0
nPool_AORaw5P	UINT	0
nPool_AV	UINT	0
nPool_AVRaw5P	UINT	0
nPool_AVEvent	UINT	0
nPool_AVDisp	UINT	0
nPool_AVSetp	UINT	0
nPool_BI	UINT	0
nPool_BIRaw	UINT	0

For example, if 42 instances of a `FB_BACnet_AV` are used, the variable `nPoolAV` can be set to 42 as the default value. If more objects are created than are available in the pool, the creation process will continue to use the `__NEW` operator.

8.4 Example

The following code shows how to use the dynamic creation/deletion feature.

Variables:

```
PROGRAM DYN_OBJECTS
VAR
    fbDynObject : FB_BACnet_DynObjectManager := (bCycleObjects := TRUE);

    bCreate : BOOL;
    bDelete : BOOL;

    DynView : POINTER TO FB_BACnet_View;
    DynAV01 : POINTER TO FB_BACnet_AV;
    DynBV01 : POINTER TO FB_BACnet_BV;
    nCounter : UDINT;
END_VAR
```

Code:

```
// Management FB has to be called in every cycle
fbDynObject();

IF (fbDynObject.Ready) THEN
    IF (bCreate) THEN
        bCreate := FALSE;
        // [Variant 1] Create standard object types:
        IF (fbDynObject.CreateObject(
                DynView,
                E_BACnet_CreateObjType.eStructuredView,
                BACnet_Globals.nBACnetInstId_Auto,
                'DynView',
                'Dynamic View',
                0))
        THEN
            // [Optional] set object properties...
        END_IF
        IF (fbDynObject.CreateObject(
                DynAV01,
                E_BACnet_CreateObjType.eAnalogValue,
                BACnet_Globals.nBACnetInstId_Auto,
                '\/DynAV01',
                '\/Dynamic AV 1',
                DynView))
        THEN
            // [Optional] set object properties...
            DynAV01^.eUnit := E_BA_Unit.eTemperature_DegreesCelsius;
        END_IF
        IF (fbDynObject.CreateObject(
                DynBV01,
                E_BACnet_CreateObjType.eBinaryValue,
                BACnet_Globals.nBACnetInstId_Auto,
                '\/DynBV01',
                '\/Dynamic BV 1',
                DynView))
        THEN
            // [Optional] set object properties...
            DynBV01^.sInactiveText := 'Off';
            DynBV01^.sActiveText := 'On';
        END_IF
    END_IF

    IF (bDelete) THEN
        bDelete := FALSE;
        // [Variant 1] Delete all objects conveniently via object manager:
        fbDynObject.Reset();

        // [Variant 2] Delete all objects manually:
         (*
        fbDynObject.DeleteObject(DynAV01);
        fbDynObject.DeleteObject(DynBV01);
        fbDynObject.DeleteObject(DynView);
        *)
    END_IF

    // Sample PLC code:
    // > Take care of valid object pointers!
    IF (fbDynObject.CreatedObjects > 0) THEN
        nCounter := (nCounter+1);
        // Simulate changing value:
        DynAV01^.bEnPgm := TRUE;
        DynAV01^.fValPgm := (TO_REAL(nCounter MOD 1000)/100);

        // [Variant 2] Call created objects manually
        IF (NOT fbDynObject.bCycleObjects) THEN
            DynView^();
            DynAV01^();
            DynBV01^();
        END_IF
    END_IF
END_IF
```

8.5 Complete initialization of the dynamic objects

The initialization of the dynamically created objects is usually done automatically a few cycles after the last object was created.

If, however, the initialization is to be completed by PLC program (because, for example, reading out a configuration file could take longer), the Dynamic Object Manager can be called in such a way that the end of the initialization takes place from the PLC program.
For this purpose, the second parameter `bAutoFinishInit` must be set to the value FALSE, e.g.
`fbDynMngr : FB_BACnet_DynObjectManager := (bCycleObjects := TRUE, bAutoFinishInit := FALSE);`

To complete the initialization, the method `FinishInit` must then be called, e.g.
`fbDynMngr.FinishInit();`

8.6 Creating and deleting own BACnet function blocks (FB)

If own BACnet FBs are to be instantiated in addition to the FBs available in the Tc3_BACnetRev14 library, these must be deleted in a method `FB_exit` when the PLC program is terminated so that the dynamically allocated memory is released again.

Dynamically created FB instances of function blocks from the Tc3_BACnetRev14 library are automatically deleted and removed from the memory via the method `FB_exit` of the Dynamic Object Manager.

The following example shows the dynamic creation of FB instances of own FBs (the implementations themselves are not shown in this example) as well as the deletion of these instances in the method `FB_exit`.

Care must be taken to ensure that the correct type of FB is used for the enable.

Variable MAIN

```
PROGRAM MAIN
VAR
    fbDynObj : FB_DYN_OBJECTS;
END_VAR
```

Code MAIN

```
fbDynObj();
```

Variables FB_DynObj

```
FUNCTION_BLOCK FB_DYN_OBJECTS
VAR
    DynMgmt : FB_BACnet_DynObjectManager := (bCycleObjects := TRUE, bAutoFinishInit := FALSE);

    bCreate : BOOL := TRUE;
    bDelete : BOOL;

    TestFbBVOwn : POINTER TO FB_BACnet_BV_Event;
    TestFbAVOwn : POINTER TO FB_BACnet_AV_EventSetp;
END_VAR
```

Code FB_DynObj

```
DynMgmt();
IF (DynMgmt.Ready) THEN
    IF (bCreate) THEN
        bCreate := FALSE;

        TestFbBVOwn := __NEW( FB_BACnet_BV_Event );
        IF (DynMgmt.CreateObjectEx(TestFbBVOwn, BACnet_Globals.nBACnetInstId_Auto, '\/TestBV own', 
'\/TestBV own', 0)) THEN
            // Initialize properties:
            TestFbBVOwn^.sInactiveText := 'AUS';
            TestFbBVOwn^.sActiveText := 'EIN';
        END_IF

        TestFbAVOwn := __NEW( FB_BACnet_AV_EventSetp );
    IF (DynMgmt.CreateObjectEx(TestFbAVOwn, BACnet_Globals.nBACnetInstId_Auto, '\/TestEvent own', 
'\/TestEvent own', 0)) THEN
            // Initialize properties:
            TestFbAVOwn^.fHighLimit := 470;
            TestFbAVOwn^.fLowLimit := -100;
    END_IF
        DynMgmt.FinishInit();
    END_IF

    IF (bDelete) THEN
        bDelete := FALSE;
    FB_exit(FALSE);
    END_IF
END_IF
```

Variables of the FB_DynObj.FB_exit method

```
METHOD FB_exit : BOOL
VAR_INPUT
    bInCopyCode : BOOL; // if TRUE, the exit method is called for exiting an instance that is copied 
afterwards (online change).
END_VAR
```

Code of the FB_DynObj.FB_exit method

```
DynMgmt.RemoveObjectEx(TestFbAVOwn);
__DELETE(TestFbAVOwn);
DynMgmt.RemoveObjectEx(TestFbBVOwn);
__DELETE(TestFbBVOwn);
```

9 Samples

9.1 Variable Declaration

The default values of the BACnet properties should be specified in the variable declaration and should only be written conditionally at runtime (so that write access by BACnet is still possible). In the following example, the *Description* property is modified at runtime.

To use the BACnet EngineeringUnit (unit of measurement) the library Tc3_BA2_Common must be included.

Variables

```
fbAv : FB_BACnet_AV := (
            sObjectName := 'X51LU01xAMH_-SW31',
            sDescription := 'SetpointVariable Ventilator',
            eUnit := E_BA_Unit.ePressure_Pascals,
            fRelinquishDefault := 250
        );
bDescriptionChanged : BOOL;
```

Code

```
if bDescriptionChanged then
    fbAV.sDescription := 'TEST';
    bDescriptionChanged := FALSE;
END_IF
fbAv();
```

9.2 BACnet properties

This example shows the presetting of further BACnet properties, such as the status texts or **C**hange-**o**f-**V**alue settings (COV). Also shown is how function blocks can be called using a for loop.

For multistate objects, the number of stages is defined via the stage texts (`Property State_Text`). The default value is a limit of 12 states. This number can be changed in the Global Variables in the parameter list `BACnet_Param`.

Variables

```
// optional unit, range and COV properties
fbAi : FB_BACnet_AI := (
            eUnit := E_BA_Unit.eTemperature_DegreesCelsius,
            fCovIncrement := 2.0,
            fMinPresValue := 0.0,
            fMaxPresValue := 100.0
        );

// optional state text information
fbBi : FB_BACnet_BI := (
            sInactiveText := 'DOWN',
            sActiveText := 'UP'
        );

// number of states determined by aStateText
fbMi : FB_BACnet_MI := (
            aStateText := ['AUTO', 'Low', 'Medium', 'High', 'Turbo']
        );

// array of BACnet FBs
afbAV : ARRAY[0..499] of FB_BACnet_AV;
nCount : INT;
```

Code

```
fbAi();
fbBi();
fbMi();

FOR nCount := 0 to 499 do
```

```
    afbAV[nCount]();
END_FOR
```

9.3 Link with the 'TcLinkTo' attribute

This example shows how the 'TcLinkTo' attribute can be used for linking with EtherCAT or K-bus terminals. The FB variants with the suffix IO (K-bus) and ECAT (EtherCAT Terminals) are available.

The 'TcLinkToOSO' attribute in the EL3068 terminal example also shows how the underrange and overrange states can be mapped to the terminal channel state (nRawState).

In general, the state of the terminal channel is mapped via the variable nRawState. The variable nRawECatState is used to additionally map the EtherCAT status of the communication with the terminal.

The TIID path can be taken from the properties of the respective terminal channel in the development environment and copied from there.

Variables

```
// general structured view object
View : FB_BACnet_View := (sObjectName:='IoBus_0_Device_3_EtherCAT');

// EtherCAT terminals

(* Term 2 (EL1809) *)
{attribute 'TcLinkTo' :=
    '.bRawVal :=TIID^Device 3 (EtherCAT)^Term 1 (EK1100)^Term 2 (EL1809)^Channel 1^Input;
    .nRawECatState:=TIID^Device 3 (EtherCAT)^Term 1 (EK1100)^Term 2 (EL1809)^InfoData^State'}
Term_2_EL1809_Chn1 : FB_BACnet_BI_ECAT := (iParent:=View, sObjectName:='Term_2_EL1809_Chn1',
nIoBusNr:=0, sDeviceType:='EL1809 16Ch. Dig. Input 24V, 3ms');

(* Term 3 (EL2809) *)
{attribute 'TcLinkTo' :=
    '.bRawVal :=TIID^Device 3 (EtherCAT)^Term 1 (EK1100)^Term 3 (EL2809)^Channel 1^Output;
    .nRawECatState:=TIID^Device 3 (EtherCAT)^Term 1 (EK1100)^Term 3 (EL2809)^InfoData^State'}
Term_3_EL2809_Chn1 : FB_BACnet_BO_ECAT := (iParent:=View, sObjectName:='Term_3_EL2809_Chn1',
nIoBusNr:=0, sDeviceType:='EL2809 16Ch. Dig. Output 24V, 0.5A');

(* Term 4 (EL3068) *)
{attribute 'TcLinkTo' :=
    '.nRawVal :=TIID^Device 3 (EtherCAT)^Term 1 (EK1100)^Term 4 (EL3068)^AI Standard Channel
1^Value;
    .nRawECatState:=TIID^Device 3 (EtherCAT)^Term 1 (EK1100)^Term 4 (EL3068)^InfoData^State'}
{attribute 'TcLinkToOSO' :=
    '.nRawState:=<0,1,0>TIID^Device 3 (EtherCAT)^Term 1 (EK1100)^Term 4 (EL3068)^AI Standard Channel
1^Status^Underrange;
    .nRawState:=<1,1,0>TIID^Device 3 (EtherCAT)^Term 1 (EK1100)^Term 4 (EL3068)^AI Standard Channel
1^Status^Overrange'}
Term_4_EL3068_Chn1 : FB_BACnet_AI_ECAT := (iParent:=View, sObjectName:='Term_4_EL3068_Chn1',
nIoBusNr:=0, sDeviceType:='EL3068 8Ch. Ana. Input 0-10V');

(* Term 5 (EL4132) *)
{attribute 'TcLinkTo' :=
    '.nRawVal :=TIID^Device 3 (EtherCAT)^Term 1 (EK1100)^Term 5 (EL4132)^Channel 1^Output;
    .nRawECatState:=TIID^Device 3 (EtherCAT)^Term 1 (EK1100)^Term 5 (EL4132)^InfoData^State'}
Term_5_EL4132_Chn1 : FB_BACnet_AO_ECAT := (iParent:=View, sObjectName:='Term_5_EL4132_Chn1',
nIoBusNr:=0, sDeviceType:='EL4132 2Ch. Ana. Output +/-10V');


// Bus Terminals (K-Bus)

(* Term 2 (KL1104) *)
{attribute 'TcLinkTo' :=
    '.bRawVal :=TIID^Device 3 (EtherCAT)^Box 6 (BK1150)^Term 2 (KL1104)^Channel 1^Input'}
Term_2_KL1104_Chn1 : FB_BACnet_BI_IO := (iParent:=View, sObjectName:='Term_2_KL1104_Chn1',
nIoBusNr:=0, sDeviceType:='KL 1104, 4 Ch. Input (24V, 3.0ms)');

(* Term 3 (KL2408) *)
{attribute 'TcLinkTo' :=
    '.bRawVal :=TIID^Device 3 (EtherCAT)^Box 6 (BK1150)^Term 3 (KL2408)^Channel 1^Output'}
Term_3_KL2408_Chn1 : FB_BACnet_BO_IO := (iParent:=View, sObjectName:='Term_3_KL2408_Chn1',
nIoBusNr:=0, sDeviceType:='KL 2408, 8 Ch. Output (24V, 0.5 A, 3 A max)');

(* Term 4 (KL3208-0010-4) *)
{attribute 'TcLinkTo' :=
    '.nRawVal :=TIID^Device 3 (EtherCAT)^Box 6 (BK1150)^Term 4 (KL3208-0010-4)^Channel 1^Data In;
    .nRawState:=TIID^Device 3 (EtherCAT)^Box 6 (BK1150)^Term 4 (KL3208-0010-4)^Channel 1^State'}
Term_4_KL3208_0010_4_Chn1 : FB_BACnet_AI_IO := (iParent:=View,
sObjectName:='Term_4_KL3208_0010_4_Chn1', nIoBusNr:=0, sDeviceType:='KL 3208-0010, 4 of 8 Ch. ana.
Input PT1000, Ni1000 (RTD)');

(* Term 6 (KL4022) *)
{attribute 'TcLinkTo' :=
    '.nRawVal :=TIID^Device 3 (EtherCAT)^Box 6 (BK1150)^Term 6 (KL4022)^Channel 1^Data Out'}
Term_6_KL4022_Chn1 : FB_BACnet_AO_IO := (iParent:=View, sObjectName:='Term_6_KL4022_Chn1',
nIoBusNr:=0, sDeviceType:='KL 4022, 2 Ch. ana. Output (4...20mA)');
```

Code

```
View();
Term_2_EL1809_Chn1();
Term_3_EL2809_Chn1();
Term_4_EL3068_Chn1();
Term_5_EL4132_Chn1();
Term_2_KL1104_Chn1();
Term_3_KL2408_Chn1();
Term_4_KL3208_0010_4_Chn1();
Term_6_KL4022_Chn1();
```

9.4 Property selection and write protection

This example shows how properties of BACnet objects can be write-protected. Furthermore, the example shows how unneeded properties can be removed from objects.

By default, all properties (which may be writeable according to the BACnet standard) are executed as writeable. All properties possible in the respective object type are also included.

Properties that are to be read-only are listed in the array `aWriteProtected`. Properties that are not used and thus should be removed from the object are listed in the array `aDisabled`.

Related properties (these are marked in the standard with a footnote of the same name) must be present in their entirety or removed in their entirety. In the example below, the properties `ChangeOfStateCount`, `ChangeOfStateTime` and `TimeOfStateCountReset` are disabled.

Variables

```
fbBi : FB_BACnet_BI := (
            sObjectName := 'Example Binary Input Object',
            sDescription := 'Objectname and Description properties are read-only',
            stSettings := (
                aDisabled := [
                    E_BACnetPropIdentifier.PropChangeOfStateCount,
                    E_BACnetPropIdentifier.PropChangeOfStateTime,
                    E_BACnetPropIdentifier.PropTimeOfStateCountReset
            ],
            aWriteProtected := [
                    E_BACnetPropIdentifier.PropObjectName,
                    E_BACnetPropIdentifier.PropDescription
            ]
        )
    );
```

Code

```
fbBi();
```

9.5 Priority controller

This example shows the use of the function blocks ending with `_5P` (5 priorities). From the available 16 priorities of the BACnet standard, 5 priorities were selected, which are sufficient for building automation projects in most cases:

LifeSafety (1): For example, emergency shutdown

Critical Equipment Control (5): For example, frost protection

Manual Local Operator (7): For example on-site operation at the control cabinet

Manual Operator (8): For example operator at the BACnet management and operating level

PLC (15): Priority of the PLC program

The number in brackets indicates the default value of the priority. This can be changed globally in the `BACnet_Param` section of the Tc3_BACnetRev14 library.

The following boolean flags are available for controlling the priority:

bEnSfty: Override of the LifeSafety priority by the PLC program. The value is given by the variables `fValSfty` (analog), `bValSfty` (binary) and `nValSfty` (multistate).

bEnCrit: Override of the critical equipment priority. The value is given by the variables `fValCrit` (analog), `bValCrit` (binary) and `nValCrit` (multistate).

bEnManLoc: Override of the manual operator priority. The value is given by the variables `fValManLoc` (analog), `bValManLoc` (binary) and `nValManLoc` (multistate).

bEnPgm: PLC priority override. The value is given by the variables `fValPgm` (analog), `bValPgm` (binary) and `nValPgm` (multistate).

bEnManualOperator: Override of the operator priority. The value is given by the variables `fValManualOperator` (analog), `bValManualOperator` (binary) and `nValManualOperator` (multistate).

This priority level is used in BACnet projects for access by the MBE (management operating equipment) and should therefore only be used if this access is not to be made on the part of BACnet or in the absence of an MBE.

Variables

```
// Analog objects supporting 5 priorities
fbAO5P : FB_BACnet_AO_5P;
fbAOIO5P : FB_BACnet_AO_IO5P;
fbAORaw5P : FB_BACnet_AO_RAW5P;
fbAV5P : FB_BACnet_AV_5P;

// Binary objects supporting 5 priorities
fbBO5P : FB_BACnet_BO_5P;
fbBOIO5P : FB_BACnet_BO_IO5P;
fbBORaw5P : FB_BACnet_BO_RAW5P;
fbBV5P : FB_BACnet_BV_5P;

// Multistate objects supporting 5 priorities
fbMO5P : FB_BACnet_MO_5P;
fbMOIO5P : FB_BACnet_MO_IO5P;
fbMORaw5P : FB_BACnet_MO_RAW5P;
fbMV5P : FB_BACnet_MV_5P;
```

Code

```
fbAO5P();
fbAOIO5P();
fbAORaw5P();
fbAV5P();
fbBO5P();
fbBOIO5P();
fbBORaw5P();
fbBV5P();
fbMO5P();
fbMOIO5P();
fbMORaw5P();

// example access to critical equipment control priority
fbMV5P.bEnCrit := TRUE;
```

```
fbMV5P.nValCrit := 3; // select state no. 3
fbMV5P();
```

9.6 Reset priorities

In some cases it may be necessary to reset priority levels described via BACnet from the PLC or visualization (i.e. write the value NULL to this priority level).
This can be done on the server side with the call `WritePropertyNull`.

It should be noted that for binary objects, priority level 6 is reserved for the minimum switch-on and switch-off times and therefore cannot be written.

Variables

```
fbBV                :     FB_BACnet_BV;
nCount              :     INT;
bEmptyPrioArray     :     BOOL;
nRet                :     DINT;
```

Code

```
fbBV();
IF bEmptyPrioArray THEN
    bEmptyPrioArray := FALSE;
    FOR nCount := 1 TO 16 DO
        IF nCount = 6 THEN
            CONTINUE;
        END_IF
        nRet := fbBV.WritePropertyNull( E_BACnetPropIdentifier.PropPresentValue, bPrio :=
TO_BYTE( nCount) );
    END_FOR
END_IF
```

9.7 Control loops and loop objects

For the mapping of control loops the BACnet standard provides the Loop object. The control parameters, such as P, I, D, limits of the control range or the operating direction are passed on to the controller function block as properties of the loop object type. The type of the controller itself is not specified in the BACnet standard.

The library Tc3_BA2_Common contains the implementation of the PID controller (FB_BA_PIDCtrl), which is mapped via the function blocks `FB_BACnet_Loop` and `FB_BACnet_LoopRef` respectively.

`FB_BACnet_Loop` implements a controller whose setpoint, output value and actual value are mapped via function block-internal variables.

`FB_BACnet_LoopRef` uses a reference to an Analog Value object as the setpoint, a reference to an Analog Output object as the output value, and a reference to an Analog Input object for the actual value.

While the function block type `FB_BACnet_Loop` is sufficient for most applications, the function block type `FB_BACnet_LoopRef` allows the operator of the management and operating level to access the controller values (setpoint, output value and actual value) via BACnet. However, this increases the number of BACnet objects per control loop to 4.

The following examples show the realization of a PI controller.

Variables

```
// control loop using internal variables
fbLoopInternal : FB_BACnet_Loop := (
                        bEn := TRUE,
                        sDescription := 'Loop using internal control parameters',
                        eOutputUnit := E_BA_Unit.eOther_Percent,
                        eAction := E_BA_Action.eReverse,
                        fProportionalConstant := 5.0,
                        fIntegralConstant := 180,
                        fSetpoint := 20
        );
fCtrlVal : REAL := 18;
```

```
// control loop using external BACnet objects
fbLoopRef_Setpt : FB_BACnet_AV_Setp := (
                        fValue := 20 );
fbLoopRef_CtrlVar : FB_BACnet_AI := (
                        fVal := 18 );
fbLoopRef_Y : FB_BACnet_AO := ();
fbLoopRef : FB_BACnet_Loop_Ref := (
                        bEn := TRUE,
                        sDescription := 'Loop using reference objects',
                        stControlledVariableReference :=
                            F_BACnet_Reference( fbLoopRef_CtrlVar, PropPresentValue),
                        stSetpointReference :=
                            F_BACnet_Reference( fbLoopRef_Setpt, PropPresentValue),
                        stManipulatedVariableReference :=
                            F_BACnet_Reference( fbLoopRef_Y, PropPresentValue),
                        eOutputUnit := E_BA_Unit.eOther_Percent,
                        eAction := E_BA_Action.eReverse,
                        fProportionalConstant := 5.0,
                        fIntegralConstant := 180
        );
```

Code

```
// internal control loop
fbLoopInternal.fCtrlVar := fCtrlVal;
fbLoopInternal();

// control loop using external object references
fbLoopRef_Setpt();
fbLoopRef_CtrlVar();
fbLoopRef_Y();
fbLoopRef();
```

9.8 Set up alarm receiver

BACnet allows alarm receivers (e.g. the MBE) to register in one or more Notification Class objects in the recipient list (Recipientlist).

If the receivers are already defined at the time of PLC programming, they can also be specified by the PLC program.

Either the Device instance number can be used as a reference to the receiver. In this case TwinCAT resolves the actual address with a Who-Is telegram (the response I-Am of the receiver contains the actual address).

Alternatively, this address can be specified directly in the recipient list, but this option is rarely used in practice.

The following example shows a Notification Class NC1 without receiver specifications (i.e. the receivers register themselves via BACnet). Notification Class 2 contains an example of a receiver referenced by the Device instance number as well as a receiver referenced by the IP address, UDP port and BACnet network number.

The order of the alarm types in the arrays `aEventEnable` and `aEventMessageTextsConfig` as well as `aAckRequired` and `aPriority` corresponds to the order in the BACnet standard:

TO_OFFNORMAL: Coming alarm / event

TO_FAULT: Sensor error, encoder malfunction

TO_NORMAL: Return to normal range

Variables

```
// simple Notification class object with empty recipient list
fbBV1 : FB_BACnet_BV := (
    nNotificationClass := 1,
    aEventEnable := [TRUE,TRUE,TRUE],
    bAlarmValue := TRUE,
    aEventMessageTextsConfig := [ 'ALARM', 'FAULT', 'NORMAL' ]
);
fbNC01_Standard : FB_BACnet_NC := (
    nObjectInstance := 1,
    nNotificationClass := 1,
    sDescription := 'NC01 Standard',
    aAckRequired := [ TRUE, TRUE, FALSE ],
    aPriority := [ 10, 11, 12 ]
);


// Notification class object with pre-defined recipient (Notification Sink module)
fbBV2 : FB_BACnet_BV := (
    nNotificationClass := 2,
    aEventEnable := [TRUE,TRUE,TRUE],
    bAlarmValue := TRUE
);
fbNC02_Recipient : FB_BACnet_NC := (
    nObjectInstance := 2,
    nNotificationClass := 2,
    sDescription := 'NC02 RecipientTest',
    aAckRequired := [ TRUE, TRUE, TRUE ],
    aPriority := [ 224, 223, 222 ],
    aRecipientList := [
         (
            stValidDays := (bMonday:=TRUE, bTuesday:=TRUE, bWednesday:=TRUE, bThursday:=TRUE,
bFriday:=TRUE),
            stFromTime := F_BA_ToSTTime(T#0H),
            stToTime := F_BA_ToSTTime(T#23H59M59S),
            stRecipient := F_BACnet_DeviceRecipient(nDeviceInstance:=42),
            nProcessId := 10000, // Notification Sink module
            bIssueConfirmed := FALSE,
            stEventTransitions := (bToOffNormal:=TRUE, bToFault:=TRUE, bToNormal:=TRUE )
        ),
        (
            stValidDays := (bSunday:=TRUE, bSaturday:=TRUE),
            stFromTime := F_BA_ToSTTime(T#7H),
            stToTime := F_BA_ToSTTime(T#15H30M),
            stRecipient := F_BACnet_EthernetRecipient(
                nIPAddress1:=192,168,10,200,
                nPort:=47808,
                nNetworkNr:=444
            ),
            nProcessId := 30100,
            bIssueConfirmed := TRUE,
            stEventTransitions := (bToOffNormal:=TRUE)
        )
    ]
);
```

Code

```
fbBV1();
fbNC01_Standard();

fbBV2();
fbNC02_Recipient();
```

9.9 Receiving alarms and events from other devices

This example shows the use of an Eventlog object as a receiver for alarms and events of an external BACnet device. The Eventlog object of the alarm receiver is entered in the notification class of the alarm transmitter. The assignment is made via the process identifier.

Device 1: Alarm transmitter

Variables

```
// Notification Class object in the alarm generating device
fbNC01 : FB_BACnet_NC := (
    nObjectInstance := 1,
    nNotificationClass := 1,
    sDescription := 'NC01',
    aAckRequired := [ TRUE, TRUE, TRUE ],
    aPriority := [ 224, 223, 222 ],
    aRecipientList :=
    [
        (
            stValidDays := (
                bMonday:=TRUE,
                bTuesday:=TRUE,
                bWednesday:=TRUE,
                bThursday:=TRUE,
                bFriday:=TRUE,
                bSaturday:=TRUE,
                bSunday:=TRUE
            ),
            stFromTime := F_BA_ToSTTime(T#0H),
            stToTime := F_BA_ToSTTime(T#23H59M59S),
            stRecipient := F_BACnet_DeviceRecipient(nDeviceInstance:=12345),
            nProcessId := 42,
            bIssueConfirmed := FALSE,
            stEventTransitions := (bToOffNormal:=TRUE, bToFault:=TRUE, bToNormal:=TRUE)
        )
    ]
);
```

Device 2: Alarm receiver (Device 12345)

Variables

```
fbELogBuf : FB_BACnet_ELogBuf := (
    sObjectName := 'Event Log for external alarms',
    bLogEnable := TRUE,
    nProcessId := 42
);
```

9.10 Prewarning limits

With the help of the EventEnrollment object BACnet allows the monitoring of properties from BACnet objects. A typical application in building automation is prewarning limits, which draw attention to an alarm that may follow later.

The following example shows how the object-internal alarm detection (Instrinsic Reporting) is combined with an additional pair of prewarning limits (Algorithmic Change Reporting) and how these states (prewarning and alarm) can be reported via two different Notification Class objects.

Here, the BACnet object `fbAV`, whose Present Value is specified from the REAL variable fRealValue, serves as the trigger for the warnings and alarms.

Variables

```
fRealValue : REAL := 50;

// Analog Value object using Intrinsic Reporting
fbAv : FB_BACnet_AV := (
            sObjectName := 'AV using Intr. Reporting',
            aEventEnable := [ TRUE, TRUE, TRUE ],
            eNotifyType := E_BACnet_NotifyType.eAlarm,
            bLowLimitEnable := TRUE,
            fLowLimit := 15.0,
            bHighLimitEnable := TRUE,
            fHighLimit := 87.0,
            fDeadband := 3,
            nTimeDelay := 2, // waits 2 seconds before TO_OFFNORMAL
            nTimeDelayNormal := 4, // wait 4 seconds before TO_NORMAL
            bEventDetectionEnable := TRUE,
            nNotificationClass := 10, // alarm class
            aEventMessageTextsConfig := [ 'Alarm', 'Fault', 'Warning' ],
            bEnPgm := TRUE,
            eUnit := E_BA_Unit.eOther_Percent
);

// Additional warning limits using Algorithmic Change Reporting
fbEE : FB_BACnet_EE := (
            sObjectName := 'Event Enrollment',
            nNotificationClass := 20,
            eNotifyType := E_BACnet_NotifyType.eNotifyEvent,
            aEventEnable := [ TRUE, TRUE, TRUE ],
            aEventMessageTextsConfig := [ 'Warning', 'Fault', 'Normal' ],
            stEventParameter := (
            eEventType := E_BACnet_EventType.eOutOfRange,
            stEventArgs := (
                stOutOfRange := (
                    nTimeDelay := 0,
                    fLowLimit := 25.0,
                    fHighLimit := 82.0,
                    fDeadband := 0.0
                    )
                )
            ),
            stObjectPropertyReference := F_BACnet_Reference(fbAv,PropPresentValue)
);

// Notification Classes 10=alarms, 20=warnings
fbNC10 : FB_BACnet_NC := (
            sObjectName := 'NC10',
            sDescription := 'Alarms',
            nObjectInstance := 10,
            nNotificationClass := 10,
            aAckRequired := [ TRUE, TRUE, TRUE ],
            aPriority := [ 10, 11, 12 ]
);
fbNC20 : FB_BACnet_NC := (
            sObjectName := 'NC20',
            sDescription := 'Warnings',
            nObjectInstance := 20,
            nNotificationClass := 20,
            aAckRequired := [ FALSE, TRUE, FALSE ],
            aPriority := [ 100, 110, 120 ]
);
```

Code

```
fbAv.fValPgm := fRealValue;
fbAv();
fbEE();
fbNC10();
fbNC20();
```

9.11 Calendar and schedule functions

For the implementation of schedule functions the BACnet standard provides the schedule object. It allows the use of recurring weekly programs (Weekly_Schedule) as well as the definition of exceptions (Exception_Schedule). Exceptions can either be entered directly in the schedule object or these are determined on the basis of a Calendar object (which must be located in the same device). A single date, a date range or a combined type of day, week and month can be used to specify an exception.

The weekly schedule is noted using the array `aWeek`.
Calendar-based exceptions are noted using the array `aCalendar`.
Exceptions contained directly in the schedule are noted using the array `aException`.

Help functions are available for specifying the dates, e.g. `F_BA_DateVal`.

For dates, according to the BACnet standard, the year starts from 1900. The Month element can contain odd (13) and even (14) months in addition to the regular month information. The day of the month may include odd (33) and even (34) days of the month in addition to the regular day information, as well as the last day (32) of the month.

When specifying specific dates, it is important that the day of the week matches the specified date.

Functions are also available for defining the time/value pairs of the schedule, e.g. `F_BACnet_SchedWeekly3xA`. Here, 3x indicates the number (3 entries). The indicator *A* stands for analog values (REAL), the indicator *B* for binary values (BOOL) and *M* for multistate states (integer).

If the schedule is to directly influence one or more properties from BACnet objects, these references can be specified in the `aObjectPropertyReferences` element. If this specification is missing, an empty list is created in the property ListofObjectPropertyReferences and the schedule does not access external objects/ properties. This can be used if the state of the schedule is to be monitored within the PLC, but no BACnet objects are to be directly influenced.

Variables

```
// Calendar object examples
fbCAL01 : FB_BACnet_Cal := (
            sObjectName := 'Calendar 1',
            sDescription := 'demonstrates a date-list for each choice',
            aDateList := [
            (
                eType := E_BA_DateValChoice.eDate,
                uDate := F_BA_DateVal(2021,E_BA_Month.eJanuary,19)
            ),
            (
                eType := E_BA_DateValChoice.eDateRange,
                uDate := F_BA_DateRangeVal(nFromYear:=2021,E_BA_Month.eJanuary,19,
nToYear:=2021,E_BA_Month.eJanuary,21)
            ),
            (
                eType := E_BA_DateValChoice.eWeekNDay,
                uDate := F_BA_WeekNDayVal(E_BA_Weekday.eFriday, E_BA_Week.eWeek1,
E_BA_Month.eFebruary)
            )
        ]
);
fbCAL02 : FB_BACnet_Cal := (
    sObjectName := 'Calendar 2'
);
fbCAL03 : FB_BACnet_Cal := (
    sObjectName := 'Calendar 3'
);

// schedule object for analog scheduling
fbAnalogOutput : FB_BACnet_AO;
fbSchedA : FB_BACnet_SchedA := (
    sObjectName := 'Schedule Analog',
    aObjectPropertyReferences :=
    [
          (iObject := fbAnalogOutput, ePropertyId := PropPresentValue)
    ],

    aWeek := F_BACnet_SchedWeekly3xA(E_BA_Weekday.eMonday, E_BA_Weekday.eFriday, T#0H, 0.0, T#6H,
5.0, T#20H, 0.0),

    aCalendar := [(
        iRefCalendar := fbCAL01,
        aEntry := F_BACnet_SchedEntry1xA(T#0H, 2.0)
    ),
    (
        iRefCalendar := fbCAL02,
        aEntry := F_BACnet_SchedEntry1xA(T#10H, 3.4)
    ),
    (
        iRefCalendar := fbCAL03,
        aEntry := F_BACnet_SchedEntry3xA(T#0H, 5, T#6H, 7, T#20H, 8)
    )],

    aException := [
    (
        eType := E_BA_DateValChoice.eDate,
        uDate := F_BA_DateVal(2020,E_BA_Month.eApril,10),
        aEntry := F_BACnet_SchedEntry1xA(T#0H, 0.0)
    ),
    (
        eType := E_BA_DateValChoice.eDateRange,
        uDate := F_BA_DateRangeVal(nFromYear:=2020,E_BA_Month.eApril,10,
nToYear:=2021,E_BA_Month.eMay,11),
        aEntry := F_BACnet_SchedEntry3xA(T#0H, 0.0, T#6H, 5.0, T#20H, 0.0)
    ),
    (
        eType := E_BA_DateValChoice.eWeekNDay,
        uDate := F_BA_WeekNDayVal(E_BA_Weekday.eFriday, E_BA_Week.eWeek1, E_BA_Month.eFebruary),
        aEntry := F_BACnet_SchedEntry3xA(T#0H, F_BA_NullA(), T#6H, 5.0, T#20H, F_BA_NullA())
    ),
    (
        eType := E_BA_DateValChoice.eDate,
        uDate := F_BA_DateVal(2019,E_BA_Month.eJune,20),
        aEntry := [
        (
            eState := E_BA_SchedEntryState.eValue,
            stTime := F_BA_ToSTTime(T#10H),
            uValue := F_BA_RVal(1.0)
        ),
        (
            eState := E_BA_SchedEntryState.eNull,
            stTime := F_BA_ToSTTime(T#11H)
        )
    ]
)]);

// schedule object for binary scheduling
bScheduledValue : BOOL;
fbSchedB : FB_BACnet_SchedB := (
    sObjectName := 'Schedule Bool',
    aWeek := F_BACnet_SchedWeekly3xB(E_BA_Weekday.eMonday, E_BA_Weekday.eFriday, T#0H, FALSE, T#6H, 
TRUE, T#20H, FALSE),
    aCalendar := [
    (
        iRefCalendar := fbCAL01,
        aEntry := F_BACnet_SchedEntry1xB(T#0H, FALSE)
    ),
    (
        iRefCalendar := fbCAL02,
        aEntry := F_BACnet_SchedEntry1xB(T#10H, TRUE)
    ),
    (
        iRefCalendar := fbCAL03,
        aEntry := F_BACnet_SchedEntry3xB(T#0H, F_BA_NullB(), T#6H, TRUE, T#20H, FALSE)
)]);

// schedule object for multistate scheduling
nMultistateValue : UDINT;
bSchedM_AssignCalReference : BOOL;
fbSchedM : FB_BACnet_SchedM :=
    (
        sObjectName := 'Schedule Multistate',
        aWeek := F_BACnet_SchedWeekly3xM(E_BA_Weekday.eMonday, E_BA_Weekday.eFriday, T#0H, 1, T#6H, 
2, T#20H, 1),
        nScheduleDefault := 3
);
```

Code

```
fbCAL01();
fbCAL02();
fbCAL03();

fbAnalogOutput();
fbSchedA();

fbSchedB();
bScheduledValue := fbSchedB.bPresVal;

// example how to assign a calendar reference at runtime
IF( bSchedM_AssignCalReference ) THEN
    bSchedM_AssignCalReference := FALSE;
    fbSchedM.aCalendar[1].iRefCalendar := fbCAL01;
    fbSchedM.aCalendar[1].aEntry := F_BACnet_SchedEntry1xM(T#11H, 2);
    fbSchedM.bWriteException := TRUE;
END_IF
fbSchedM();
nMultistateValue := fbSchedM.nPresVal;
```

9.12 Logging objects

BACnet allows the use of three different objects for log data storage:

Trendlog: This object type represents the recorded log data of a single data source, e.g. the Present Value of an Analog Input object.

Here, recording can be done by polling (i.e., at a fixed recording interval in 1/100s), by **C**hange **o**f **V**alue (COV), or by using a Boolean trigger.

Trendlog Multiple: This object type allows simultaneous recording of data from multiple sources. In principle, recording is only possible via polling or trigger, but not via COV. In practice, this object type is rarely used in BACnet projects.

Eventlog: This object type allows the storage of BACnet event messages (events and alarms). If a Notification Class object with the same instance number exists in the server for the instance number of an Eventlog object, the Notification Class object is automatically configured as the data source of the Eventlog object. Alarms reported via this notification class are therefore automatically stored in the log memory of the Eventlog object.

The following example shows the use of these three object types.

Variables

```
    fAnalogValue : REAL;
    fbTon : TON;
    fbAv : FB_BACnet_AV := (
                            sObjectName := 'Object to simulate changes and generate alarms',
                            fLowLimit := 1.0,
                            bLowLimitEnable := TRUE,
                            fHighLimit := 8.0,
                            bHighLimitEnable := TRUE,
                            bEnPgm := TRUE,
                            bEventDetectionEnable := TRUE,
                            nNotificationClass := 42,
                            aEventEnable := [TRUE,TRUE,TRUE],
                            eUnit := E_BA_Unit.eElectrical_Volts,
                            nTimeDelay := 0,
                            nTimeDelayNormal := 0,
                            fDeadband := 0.0
                            );

    fbNC42 : FB_BACnet_NC := (
                            nObjectInstance := 42,
                            nNotificationClass := 42,
                            sDescription := 'NC42',
                            aAckRequired := [ TRUE, TRUE, TRUE ],
                            aPriority := [ 1, 2, 3 ]
                            );

    fbELog_NC42 : FB_BACnet_ELogBuf := (
                            sObjectName := 'Event Log for NC42',
                            nObjectInstance := 42,
                            bLogEnable := TRUE
                            );

    fbTLog : FB_BACnet_TLog := (
                            sObjectName := 'Trend',
                            nNotificationClass := 42,
                            aEventEnable := [ TRUE, TRUE, FALSE ],
                            stStartTime := F_BA_ToSTDateTime(DT#2010-01-01-00:00),
                            bLogEnable := TRUE,
                            eLoggingType := E_BA_LoggingType.ePolled,
                            nLogInterval := 3 * 100, // log every 3 seconds, value in 1/100 seconds!
                            stObjectPropertyReference := F_BACnet_Reference(fbAv, PropPresentValue)
                            );
    fbTLogCov : FB_BACnet_TLog := (
                            sObjectName := 'Trend Cov',
                            bLogEnable := TRUE,
                            stObjectPropertyReference := F_BACnet_Reference(fbAv, PropPresentValue),
                            eLoggingType := E_BA_LoggingType.eCOV,
                            nCOVResubscriptionInterval := 600,
                            stClientCOV := (
                                eChoice := E_BACnet_ClientCOVChoice.eCovReal,
                                fIncrement := 5.0
                                )
                            );

    fbTLogBuf : FB_BACnet_TLogBuf := (
                            sObjectName := 'Trend with PLC buffer',
                            bLogEnable := TRUE,
                            eLoggingType := E_BA_LoggingType.eTriggered,
                            stObjectPropertyReference := F_BACnet_Reference(fbAv, PropPresentValue)
                        );

    fbTLogM : FB_BACnet_TLM := (
                            sObjectName := 'Trend Multiple',
                            bLogEnable := TRUE,
                            eLoggingType := E_BA_LoggingType.ePolled,
                            nLogInterval := 50, // log every 0.5 seconds, value in 1/100 seconds!
                            aObjectPropertyReferences := [
                                (iObject := fbAv, ePropertyId := PropPresentValue),
                                (iObject := fbAv, ePropertyId := PropStatusFlags),
                                (iObject := fbAv, ePropertyId := PropEventState) ]
                        );
```

Code

```
// generate value changes every second
fbTon( IN:= NOT fbTon.Q, PT:=T#1S );
IF fbTon.Q THEN
    fAnalogValue := fAnalogValue + 1.0;
    IF fAnalogValue > 10.0 THEN
    fAnalogValue := 0;
    END_IF
END_IF
fbAv.fValPgm := fAnalogValue;
fbAv();

// Notification Class
fbNC42();

// Event Log with buffer in PLC
fbELog_NC42();

// Trend Log every 3 seconds
fbTLog();

// COV based Trend Log
fbTLogCov();

// Trigger based Trend Log with buffer in PLC
fbTlogBuf();

// Trendlog Multiple
fbTLogM();
```

9.13 Processing of the log memory in the PLC

BACnet defines the CHOICE data type for various properties. This represents a selection of different possible property values.

To be able to process these data types in the PLC, the PLC data type UNION is used together with an enumeration. The enumeration contains the information which element of the union must be accessed.

Log memory of the Trendlog object

The following example shows the processing of the log memory of a Trendlog object.

An object of the type `FB_BACnet_TLogBuf` is used for this purpose. This provides a log memory whose contents can be read out by the PLC. The variable `aLogBuffer` is available as an array of the type `T_BACnet_TLogBuffer`. The individual elements of the log buffer are of type `ST_BA_TrendEntry` with the following elements:

dtTime: Contains the date and time of the log entry in BACnet format (`ST_BA_Date` and `ST_BA_Time`).

eType: Contains the information about the type of the log entry:

`eBinary`: The log entry contains a value of type BinaryPV. In the element uValue the value of the variable bVal is to be processed.

`eAnalog`: The log entry contains a value of type REAL. In the element uValue the value of the variable fVal is to be processed.

`eMultistate`: The log entry contains a value of type UDINT. In the element uValue the value of the variable udiVal is to be processed.

`eEvent`: The log entry contains an event. Four bits are available in the stEvent structure:

`bStart`: Recording has been started.

`bStop`: Recording has been stopped.

`bBufferPurged`: The log memory has been cleared.

`bInterrupted`: Recording was interrupted (e.g. due to a network interruption)

In addition, the four status flags are available in the stState structure element:

`bInAlarm`: The object is in an active alarm state.

`bFault`: The object is in a faulty state, e.g. encoder malfunction.

`bOverridden`: The object has been overridden by an internal mechanism, e.g. by the PLC.

`bOutOfService`: The object was put out of operation. Values are simulated.

Log memory of the Eventlog object

The following example shows the processing of the log memory of an Eventlog object.

An object of the type `FB_BACnet_ELogBuf` is used for this purpose. This provides a log memory whose contents can be read out by the PLC. The variable `aLogBuffer` is available as an array of the type `T_BACnet_ELogBuffer`. The individual elements of the log buffer are of type `ST_BACnet_EventLogEntry` with the following elements:

dtTime: Contains the date and time of the log entry in BACnet format (`ST_BA_Date` and `ST_BA_Time`).

eType: Contains the information about the type of the log entry:

`eStatus`: The log entry contains the information of the StatusFlags. Four bits are available in the stStatus structure:

`bInAlarm`: The object is in an active alarm state.

`bFault`: The object is in a faulty state, e.g. encoder malfunction.

`bOverridden`: The object has been overridden by an internal mechanism, e.g. by the PLC.

`bOutOfService`: The object was put out of operation. Values are simulated.

`eTimesync`: A time synchronization message has been received. The fTimeSync element contains the delta to the previous time.

`eNotification`: The log entry contains an event message. In the structure `stNotification` the parameters of the event message are available, e.g. the process ID, the triggering device and object, the alarm priority, etc.

9.14 Primitive Value Objects

Primitive Value objects can be used to represent simple data types, such as strings, integer values, or for date or time specifications.

For dates, according to the BACnet standard, the year starts from 1900. The Month element can contain odd (13) and even (14) months in addition to the regular month information. The day of the month may include odd (33) and even (34) days of the month, as well as the last day (32) of the month, in addition to the regular day information.

When specifying specific dates, it is important that the day of the week matches the specified date.

For clock times, the range is 0-23 hours, 0-59 minutes, 0-59 seconds, and 0-99 hundredths of a second.

The value 255 stands as a placeholder for any value (e.g. every year or every hour) for the objects of the type *Pattern*. For the enumeration types, the enumerated value *Unspecified* can be used alternatively.

Variables

```
fbPositiveInteger : FB_BACnet_INT;
fbLargeAnalog : FB_BACnet_LAV;
fbCharacterString : FB_BACnet_String;
fbUnsignedInteger : FB_BACnet_UINT;
```

```
nValue : INT := 15;
fValue : LREAL := 42.3;
sValue : STRING := 'TwinCAT BACnet';
uiValue : UINT := 12345;

fbDate : FB_BACnet_Date;
fbDatePattern : FB_BACnet_DateP;
fbDateTime : FB_BACnet_DateTime;
fbDatetimePattern : FB_BACnet_DateTimeP;
fbTime : FB_BACnet_Time;
fbTimePattern : FB_BACnet_TimeP;

// specific date
stDate : ST_BA_Date := ( nYear := 122, eMonth := E_BA_MONTH.eDecember, nDay := E_BA_DAY.eDay02,
eDayOfWeek := E_BA_WEEKDAY.eFriday );

// every year christmas eve (regardsless of the day of week)
stDatePattern : ST_BA_Date := ( nYear := 255, eMonth := E_BA_MONTH.eDecember, nDay :=
E_BA_DAY.eDay24, eDayOfWeek := E_BA_WEEKDAY.Unspecified );

// specific date and specific time
stDateTime : ST_BA_DateTime := (
stDate := ( nYear := 122, eMonth := E_BA_MONTH.eDecember, nDay := E_BA_DAY.eDay02, eDayOfWeek :=
E_BA_WEEKDAY.eFriday ),
stTime := ( nHour := 17, nMinute := 53, nSecond := 42, nHundredths := 19 ) );

// every year where the 1st of May is a Monday each hour / minute at 11 seconds
stDateTimePattern : ST_BA_DateTime:= (
stDate := ( nYear := 255, eMonth := E_BA_MONTH.eMay, nDay := E_BA_DAY.eDay01, eDayOfWeek :=
E_BA_WEEKDAY.eMonday ),
stTime := ( nHour := 255, nMinute := 255, nSecond := 11, nHundredths := 0 ) );

// specific time
stTime : ST_BA_Time := ( nHour := 11, nMinute := 42, nSecond := 38, nHundredths := 45 );

// every hour at minute 42
stTimePattern : ST_BA_Time := ( nHour := 255, nMinute := 42, nSecond := 0, nHundredths := 0 );
```

Code

```
fbPositiveInteger.nValue := nValue;
fbPositiveInteger();

fbLargeAnalog.fValue := fValue;
fbLargeAnalog();

fbCharacterString.sValue := sValue;
fbCharacterString();

fbUnsignedInteger.nValue := uiValue;
fbUnsignedInteger();

fbDate.stValue := stDate;
fbDate();

fbDatePattern.stValue := stDatePattern;
fbDatePattern();

fbDateTime.stValue := stDateTime;
fbDateTime();

fbDatetimePattern.stValue := stDateTimePattern;
fbDatetimePattern();

fbTime.stValue := stTime;
fbTime();

fbTimePattern.stValue := stTimePattern;
fbTimePattern();
```

9.15 Structured View objects

Structured View objects allow the operator view to be mapped to the objects present in a BACnet device, often using a data point addressing description.

With the exception of the top level (root level), all elements are connected to the respective parent object with the help of the iParent element.

The \/ (backslash and slash) characters can be used to create a text composition with the respective parent object.

For the properties Objectname, Description and EventMessageTextsConfig the separator is used, which is defined in the BACnet_Param of the library instance.

To use the function `STRING_TO_UTF8` the library Tc2_Utilities must be integrated.

Variables

```
// use \/ to concat to the string provided by parent node
// countries (root level)
fbGermany : FB_BACnet_View := (
    eNodeType := E_BACnet_NodeType.eOrganizational,
    sObjectName := 'Germany',
    sDescription := 'BECKHOFF offices in Germany' );

fbSwitzerland : FB_BACnet_View := (
    eNodeType := E_BACnet_NodeType.eOrganizational,
    sObjectName := 'Switzerland',
    sDescription := 'BECKHOFF offices in Switzerland');

fbSpain : FB_BACnet_View := (
    eNodeType := E_BACnet_NodeType.eOrganizational,
    sObjectName := 'Spain',
    sDescription := 'BECKHOFF offices in Spain');


// view structure Verl
fbVerl : FB_BACnet_View := (
    iParent := fbGermany,
    eNodeType := E_BACnet_NodeType.eOrganizational,
    sObjectName := '\/Verl',
    sDescription := '\/Verl offices');

fbEiserstr : FB_BACnet_View := (
    iParent := fbVerl,
    eNodeType := E_BACnet_NodeType.eOrganizational,
    sObjectName := '\/Eiserstr',
    sDescription := '\/Eiserstr offices');

fbFirstFloor : FB_BACnet_View := (
    iParent := fbEiserstr,
    eNodeType := E_BACnet_NodeType.eOrganizational,
    sObjectName := '\/Floor 2',
    sDescription := '\/Hardware Development');

fbCabinet : FB_BACnet_View := (
    iParent := fbFirstFloor,
    eNodeType := E_BACnet_NodeType.eCollection,
    sObjectName := '\/Controllers',
    sDescription := '\/Cabinet 07');

fbAv01 : FB_BACnet_AV := (
    sObjectName := '\/AV01',
    sDescription := '\/Analog Value01',
    iParent := fbCabinet );

fbController : FB_BACnet_View := (
    iParent := fbCabinet,
    eNodeType := E_BACnet_NodeType.eDevice,
    sObjectName := '\/C6015',
    sDescription := '\/Edge Device');

fbAv02 : FB_BACnet_AV := (
    sObjectName := '\/AV02',
    sDescription := '\/Analog Value02',
    iParent := fbController );


// view structure Madrid
fbMadrid : FB_BACnet_View := (
    iParent := fbSpain,
    eNodeType := E_BACnet_NodeType.eOrganizational,
    sObjectName := '\/Madrid',
    sDescription := '\/Madrid offices');

fbAv03 : FB_BACnet_AV := (
    sObjectName := '\/AV03',
    sDescription := '\/Test äöüÄÖÜß',
    iParent := fbMadrid );

bChangeAv03 : BOOL;
sTest : STRING(255);
```

Code

```
// Germany
fbGermany();
fbVerl();
fbEiserstr();
fbFirstFloor();
fbCabinet();
fbAv01();
fbController();
fbAv02();

// Switzerland
fbSwitzerland();

// Spain
fbSpain();
fbMadrid();

IF bChangeAv03 THEN
    bChangeAv03 := FALSE;
    sTest := 'ßÄÖÜäöü This is a test';
    // requires library Tc2_Utilities
    STRING_TO_UTF8( ADR( fbAv03.sDescription ), ADR( sTest ), SIZEOF( fbAv03.sDescription) );
END_IF

fbAv03();
```

9.16 Array initialization

If the property values of BACnet FBs are to be initialized as an array with identical values, the syntax described below can be used. The Server and iParent properties are assigned to all elements of the AI, AO and AV arrays. This example also shows the cyclic call using a for loop.

Variables

```
FUNCTION_BLOCK FB_BACnetServer
VAR
    Adapter : FB_BACnet_Adapter;
    Server : FB_BACnet_Server := (Adapter := Adapter);

    View_AI : FB_BACnet_View := (Server := Server, sObjectName := 'AI_Objects', sDescription :=
'Collector AI Objects');

    View_AO: FB_BACnet_View := (Server := Server, sObjectName := 'AO_Objects', sDescription :=
'Collector AO Objects');

    View_AV : FB_BACnet_View := (Server := Server, sObjectName := 'AV_Objects', sDescription :=
'Collector AV Objects');

    AI : ARRAY[1..MAX_OBJECTS] OF FB_BACnet_AI := [MAX_OBJECTS((Server := Server, iParent :=
View_AI))];

    AO : ARRAY[1..MAX_OBJECTS] OF FB_BACnet_AO := [MAX_OBJECTS((Server := Server, iParent :=
View_AO))];

    AV : ARRAY[1..MAX_OBJECTS] OF FB_BACnet_AV := [MAX_OBJECTS((Server := Server, iParent :=
View_AV))];

    iFor : INT;
END_VAR
VAR CONSTANT
    MAX_OBJECTS : INT := 50;
END_VAR
```

Code

```
Adapter();
Server();
View_AI();
View_AO();
View_AV();

FOR iFor := 1 TO MAX_OBJECTS DO
    AI[iFor]();
    AO[iFor]();
    AV[iFor]();
END_FOR
```

10 Support and Service

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

Download finder

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

The downloads are available in various formats.

Beckhoff's branch offices and representatives

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

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

You will also find further documentation for Beckhoff components there.

Beckhoff Support

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

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

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

Beckhoff Service

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

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

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

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

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

More Information:
www.beckhoff.com/tf8020

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