

Manual | EN

TwinCAT 3

Corrected timestamps

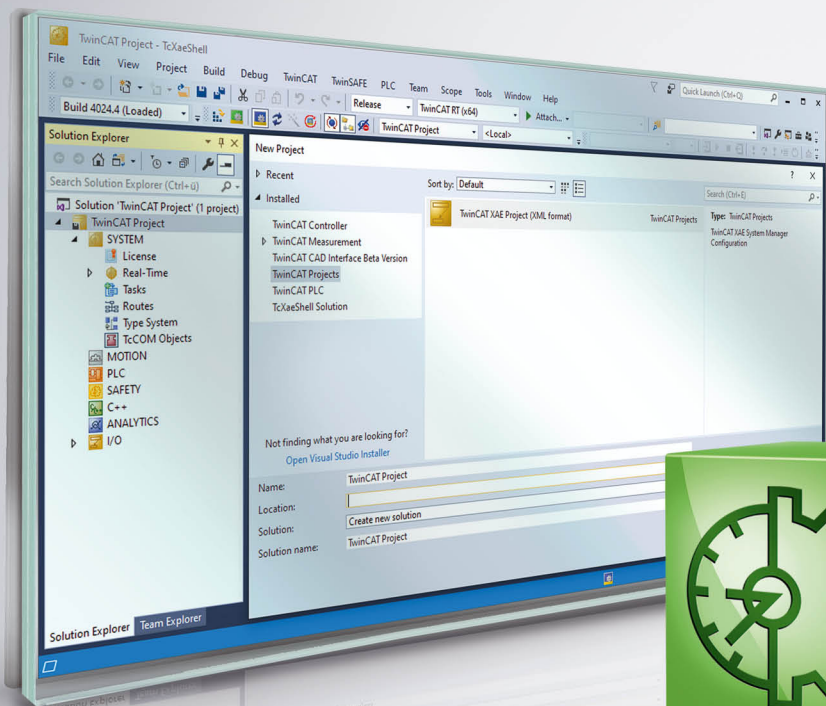


Table of contents

1 Foreword	5
1.1 Notes on the documentation.....	5
1.2 Safety instructions	6
1.3 Notes on information security	7
2 Overview	8
3 System requirements	9
4 Limitations	10
5 Technical introduction	11
5.1 Consumers	11
5.1.1 TwinCAT components as offset consumers	11
5.1.2 Application implementation	11
5.2 Provider	12
5.2.1 NTP provider.....	12
5.2.2 DC provider.....	16
5.2.3 Application implementation	18
6 Real-time API	20
6.1 Structures	20
6.1.1 Enum TimeType	20
6.2 Interfaces	20
6.2.1 ITcSetExternalTime interface	20
6.2.2 ITcExternalTime interface	22
7 ADS API	25
8 Samples	27
8.1 ADS consumer	27
8.2 PLC Consumer	27
8.3 C++ consumer	28
8.4 C++ provider	28
9 Appendix	30
9.1 FAQ	30
9.1.1 Windows as NTP client.....	30
9.1.2 Windows as NTP server	30

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.

It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.

It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the 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 prior announcement. No claims for the modification of 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 trademarks of and licensed by Beckhoff Automation GmbH.

Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

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

EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.



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

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations 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 engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

DANGER

Serious risk of injury!

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

WARNING

Risk of injury!

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

CAUTION

Personal injuries!

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

NOTE

Damage to the environment or devices

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.



Tip or pointer

This symbol indicates information that contributes to better understanding.

1.3 Notes on information security

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

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

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

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

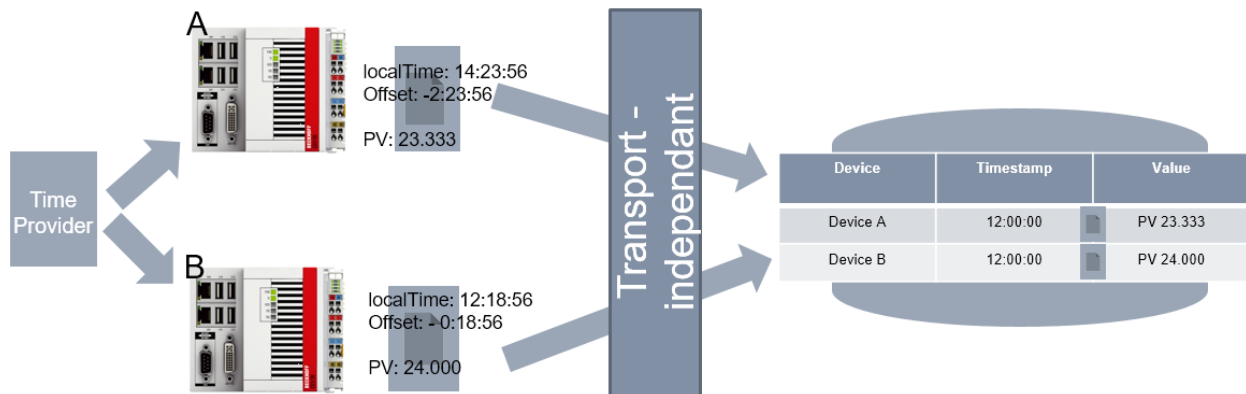
2 Overview

Controllers generate data to be collected and linked in modern, distributed systems. Since controllers start off as stand-alone devices, they have independent time bases. In a common database, it would not be possible to correlate data with respect to time.

In order to counter this problem, it has been possible for quite some time to synchronize controllers with each other, for example using the network protocol IEEE1588 or PTP.

However, in many scenarios it is sufficient to provide the data with a uniform timestamp. The controllers can be operated independently of each other, so that on the one hand the hardware costs associated with the protocols mentioned above are reduced, while on the other hand there is no technical dependency between the controllers.

This chapter describes the TwinCAT components for adapting timestamps for storing time-synchronous data.



The figure illustrates the basic idea: independent controllers obtain the local timestamp and adjust it using an offset, which is then used to store the common data.

A central component, the external time interface, is available in TwinCAT real-time for this purpose. This component

- receives the offset to the corrected time from a configured source (external time provider).
- provides the external time consumer with a corrected time, depending on the current local time.

This corrected time can then be used by different components inside and outside the real-time.

The source is typically either an NTP server or a DC time signal based on EtherCAT, which is synchronized via EL6688 through PTP (IEEE1588), for example. However, a source can also be implemented by the customer, so that other time signals can be realized as a source.

In addition to the central component in the TwinCAT real-time described above, the concept thus comprises two types of components:

1. External time providers: provide an offset for adjusting timestamps of the central component. For example, a provider obtains a timestamp via NTP (Network Time Protocol, see RFC 4330), from which it calculates an offset to the local system time and makes this available.
2. External time consumers: use an offset that they obtain from the central component. Thus a timestamp can be used in the components that leads to comparable data on remote devices. All TwinCAT components that use timestamps can be consumers, and also customer applications.

3 System requirements

Technical data	Requirement
Operating system	Windows 7/10, Windows Embedded Standard 7
Target platform	PC architecture (x86, x64)
TwinCAT version	TwinCAT 3.1 build 4024.0 or higher
Required TwinCAT setup level	TwinCAT 3 XAE, XAR
Required TwinCAT license	Any runtime license (PLC, C++)

4 Limitations

Some important limitations have to be taken into account:

- The TwinCAT system time is not changed by the external time interface described here
- The external time offsets are made available to the consumers as provided by the provider. It follows that
 - the offset must be calculated correctly by the provider.
 - no monotony can be guaranteed in the timestamps.
- The external time offsets are not saved and subsequently made available for retrieval. This means that only the current offsets are managed in the TwinCAT system.

5 Technical introduction

TwinCAT offers different interfaces for the external time provider and the external time consumer in order to utilize the concept of corrected timestamps.

On the external time consumer side, different TwinCAT components are able to use the external timestamp. In addition, there are different access options for applications.

On the external time provider side, modules are provided that can calculate and provide an offset via NTP. In addition, there is a module that can use the offset via DC. The corresponding interface for providing the offset is also offered for TwinCAT C++, so that customers can create their own external time providers.

Timestamps for different use cases

It should be noted that TwinCAT differentiates between four types of timestamps in this concept:

1. None: Local system time and no correction
2. Soft: Recommended use e.g. for NTP
3. Medium: Recommended use e.g. for IEEE1588
4. Hard: Recommended use e.g. for hardware synchronization where no drift should occur

An external time provider provides one of the possible offsets; only one provider can be defined for each type.

An external time consumer can then use any offset; all four offset types can be used as required. Thus it is possible to use different timestamps in different ensembles or operation modes. For example, a local diagnosis can take place with the local system time, while at the same time aggregated data from different systems can be corrected with the offset type Soft and stored in a common database.

The interfaces of the corrected timestamps use data types with a length of 8 bytes and are counted from 1.1.1601 in 100 ns steps.

5.1 Consumers

External time consumers are components that can correct the local system time with an offset. For this purpose, the components must select or configure an offset of type Soft, Medium or Hard and query it accordingly.

5.1.1 TwinCAT components as offset consumers

The following TwinCAT components support the approach of corrected timestamps – the respective documentation describes how this functionality can be enabled:

- TwinCAT 3 EventLogger

This list will be extended.

5.1.2 Application implementation

Applications can use the external time offsets in different components:

- Real-time PLC: The PLC can query an offset or have a local timestamp corrected accordingly.
- Real-time – C++: C++ TcCOM modules are able to query the offset and act accordingly.
- User mode – ADS device notifications: The timestamps sent with the ADS device notifications can be corrected.
- User mode – ADS Read: The corrected timestamp can be retrieved by an ADS Read. This can be used in ADS Sum commands to retrieve a timestamp along with data.

The interfaces are documented in the corresponding API chapters.

5.2 Provider

External time providers are components that determine an external time offset in relation to the local system time through an external information source and make it available in TwinCAT. This allows external time consumers to receive a corrected time, independent of the provider.

TwinCAT also supplies providers with:

- NTP providers: an implementation that queries and provides a time signal from an NTP server via (S)NTP.
- DC providers: An implementation that passes on the DC time from the EtherCAT master to TwinCAT as an offset (e.g. via IEEE1588 or PTP)
- In addition, the customer is able to provide his own providers.

5.2.1 NTP provider

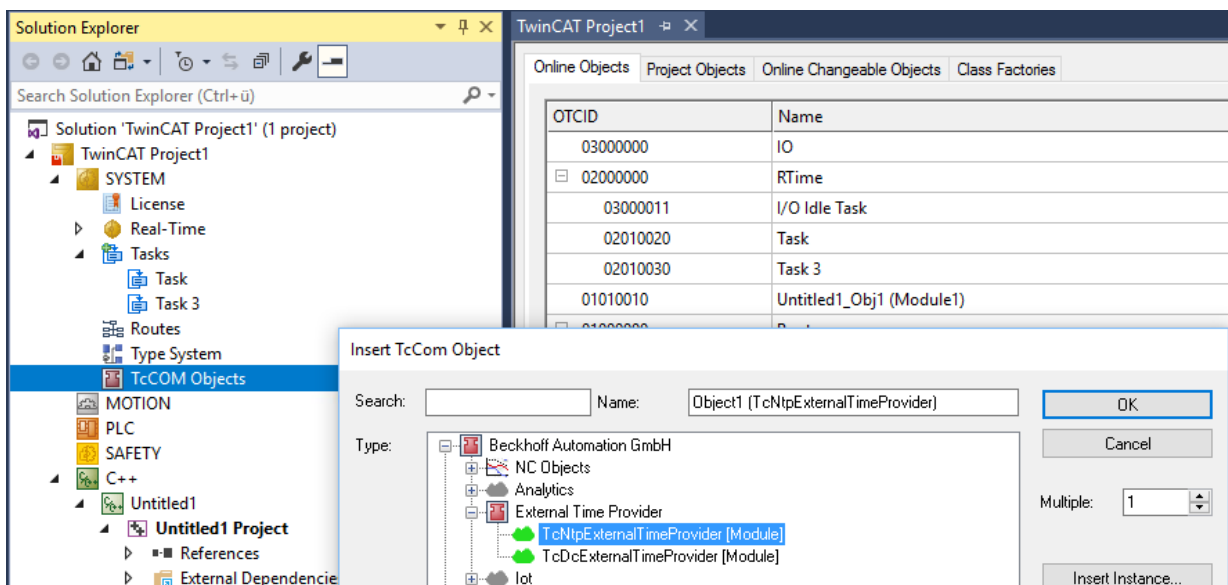
The NTP provider is an (S)NTP client that cyclically receives a time signal from an NTP server. This allows it to calculate an offset of the system time from the time signal of the NTP server and make it available accordingly.

Configuration

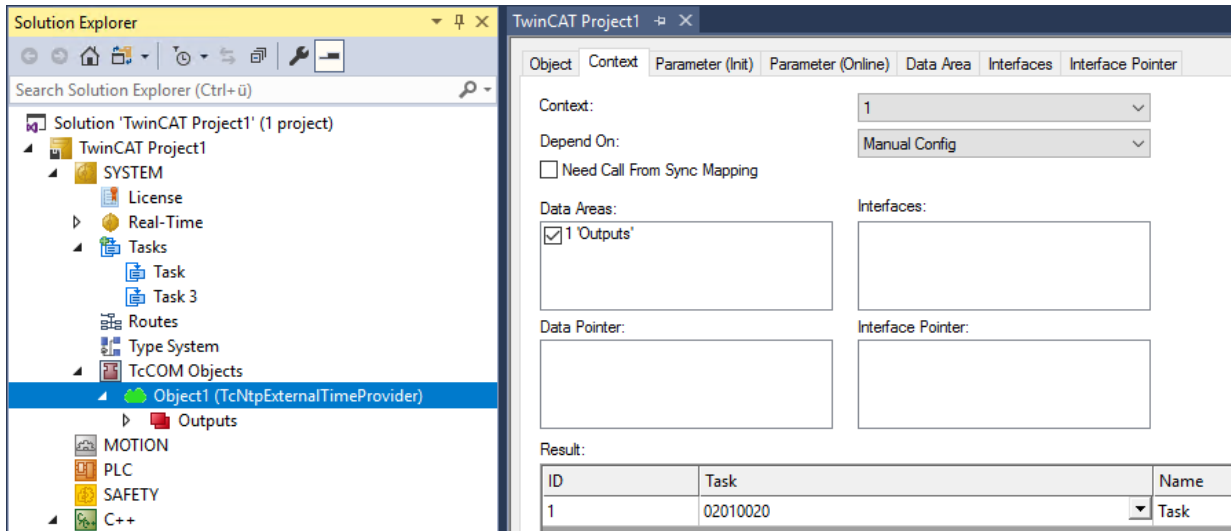
The NTP provider is implemented as TcCOM module TcNtpExternalTimeProvider. This module is commissioned as a TcCOM module as follows:

✓ TwinCAT project

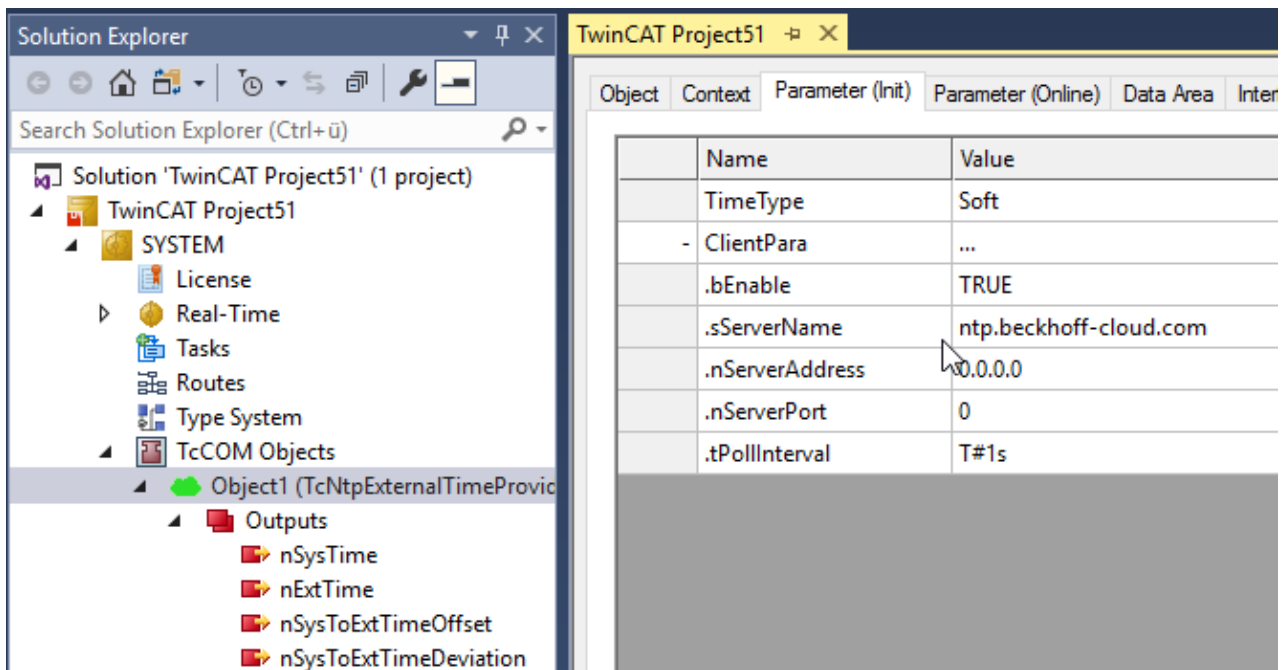
1. Insert a TcCOM module under System->TcCOM Objects and select type TcNtpExternalTimeProvider in the category External Time Provider.



- The module requires a task from which it is called. This is parameterized via the **Context** tab of the module:



⇒ The TcCOM module can be parameterized:



The configuration takes place in the **Parameter (Init)** tab. The parameters have the following meanings:

- **TimeTime**: The type of offset for which this module is to determine an offset.

Client Para:

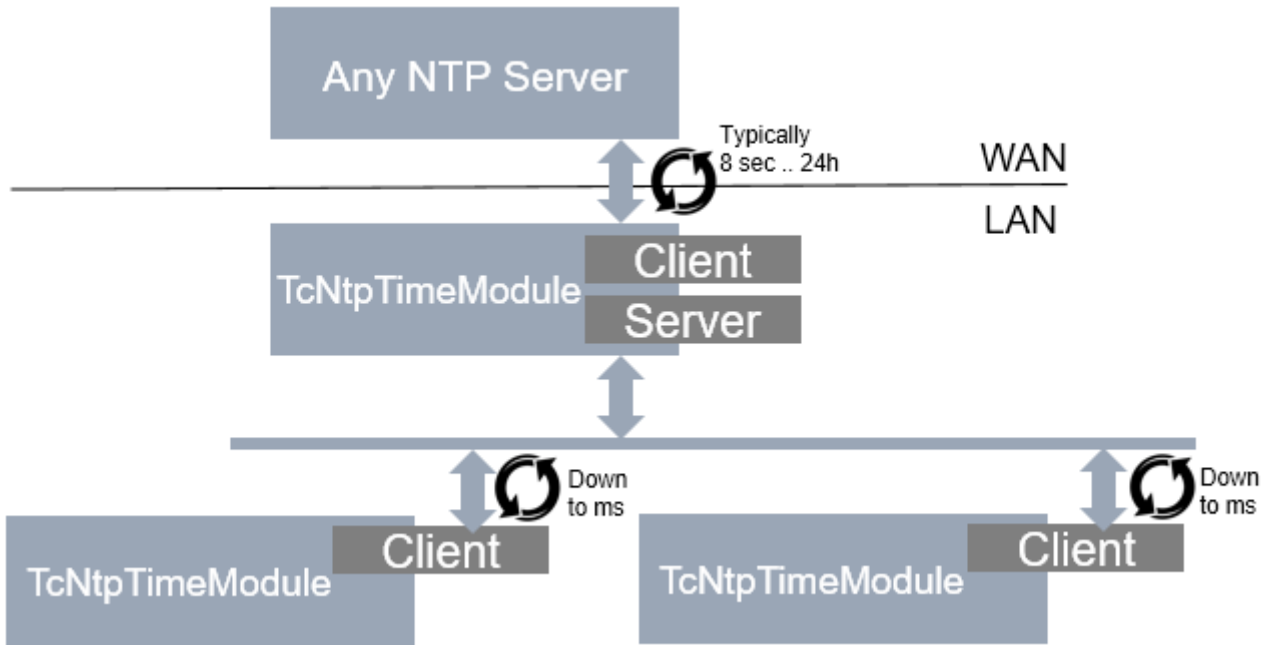
- **bEnable**: The module can be disabled to prevent NTP communication.
- **sServerName**: The name of the NTP server to be used as the source.
- **nServerAddress**: IP address of an NTP server (used if sServerName is empty).
- **nServerPort**: The UDP port of the NTP server to be used (default: 123).
- **tPollInterval**: The interval in which the NTP queries are to be started. The maximum specified by the server is taken into account, which may slow down requests.

This module passes a determined offset to TwinCAT via the [ITcSetExternalTime \[P 20\]](#) interface. In addition, outputs are available for mapping.

NTP provider as NTP server

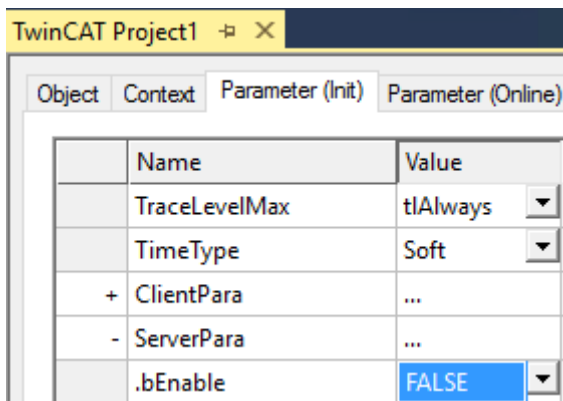
Optionally, the same module can also act as an NTP server. Thus, a time signal can be obtained from an external NTP server (as a client) and simultaneously provided to lower-level systems.

For the external server, the NTP protocol typically requires a minimum query time of 8 seconds or more. The NTP provider as NTP server, on the other hand, allows more frequent query intervals.



Server function

The server functionality is normally hidden. It can be displayed and configured via **Show Hidden Parameters**:



- **bEnable:** Enable NTP server functionality for this module. To do this, open the udp/123 port in the Windows firewall.
- **nPort:** The UDP port that is used to offer the server (default: 123).

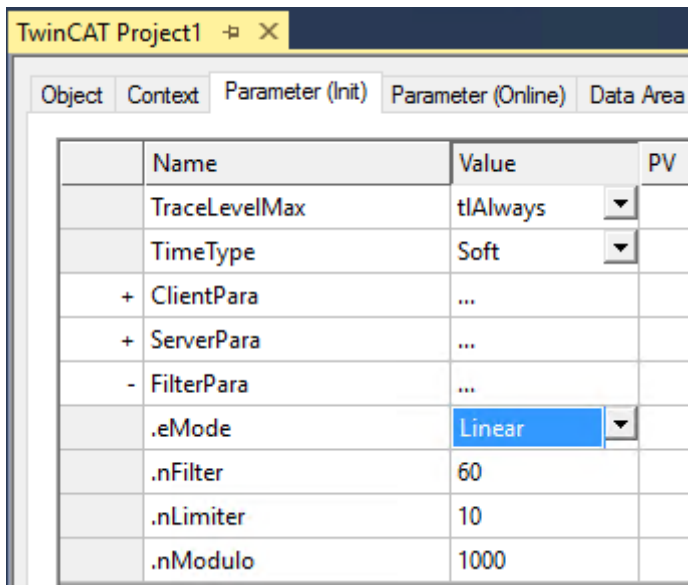
The following parameters are used to adjust the NTP information provided. By default, the parameters are set as specified in the protocol; they can be overwritten here:

- **nLeap:** Manual configuration of the Leap Indicator.
- **nStratum:** Manual configuration of the stratum.
- **nRoot:** Manual configuration of the root server information, as defined depending on the stratum.

Filter function

If offsets are determined by the NTP server query, the module can independently perform a transition from the old offset to the new offset.

This functionality is normally hidden. It can be displayed and configured via **Show Hidden Parameters**:



- **eMode**: A selection of modes. Currently, either no adjustment or a linear adjustment is made (default).

The following parameters apply if "Linear" is selected as eMode:

- **nFilter**: Number of values for which the average is taken, i.e. number of NTP responses. With a poll interval of 1 s, nFilter = 60 would effect a filter for one minute. (Default: 60).
- **nLimiter**: The offset is changed by this value at the most per cycle. If the difference between the local and external clocks were to be 100 ms and the cycle time 1 ms, it would thus take 100,000 cycles or 1.6 minutes at nLimiter = 10 until the offset has settled. (Default: 1 µs).
- **nModulo**: Rounding of the offset. Usually this should be chosen depending on the cycle time. The offset is adjusted via this modulo so that no "un-round" times are created. The DC Time will return the modulo of the cycle time; corrected with the offset, the time stamp thus remains "round". The offset/ time stamp changes as a result, but also with small jumps if an adjustment takes place. As described under nLimiter and with nModulo = 1000, the offset and thus the relative time stamp would increment every 100th cycle by 0.1 ms.

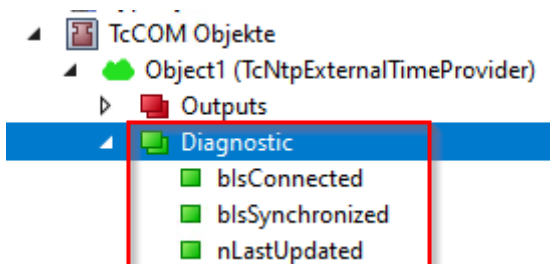
Diagnostics

Diagnostic information can be viewed under the **Parameters (Online)** tab.

Object	Context	Parameter (Init)	Parameter (Online)	Data Area	Interfaces	Interface Pointer	Name	Online	CS	Unit
-	ServerInfo		...						<input type="checkbox"/>	
			.nLeap					0		
			.nVersion					4		
			.nMode					4		
			.nStratum					2		
			.tPollIntv					T#1m4s		[ms]
			.fPollPrec					1e-07		[s]
			.fRootDelay					0.00047302968		[s]
			.fRootDisp					0.0029449912		[s]
			.sRefId					129.70.130.70		
			.nRefTime					2019-06-11T07:24:03.9653238		
			.nOrgTime					2019-06-11T07:24:05.1789999		
			.nRecTime					2019-06-11T07:24:05.4467752		
			.nTmtTime					2019-06-11T07:24:05.4468292		
			.nDstTime					2019-06-11T07:24:05.199		

For each line there is a corresponding description in the **Comment** column.

In addition, corresponding symbols are available for programmatic evaluation:



5.2.2 DC provider

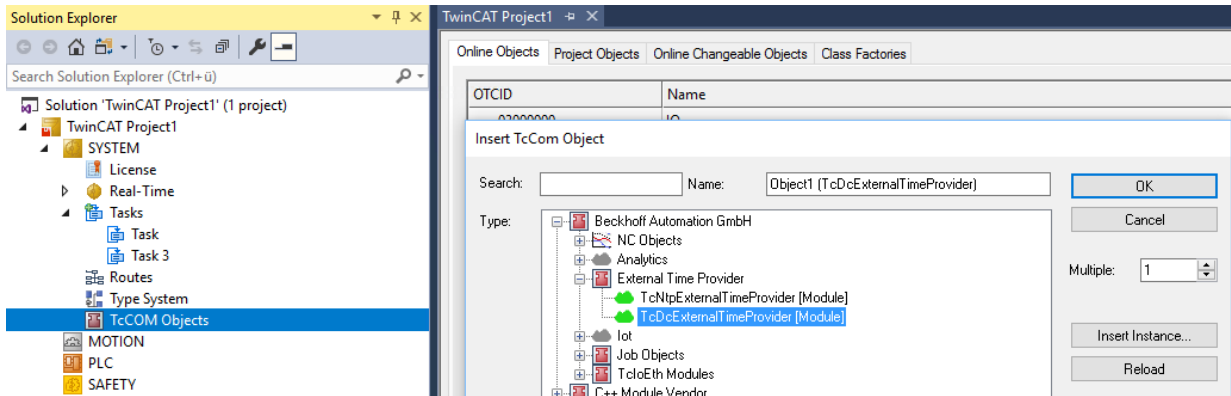
The DC provider obtains an offset through mapping from an EtherCAT master. It can be used to use time values from the I/O range as offset, such as those provided by the EtherCAT master (DC time) or an EL6695.

Configuration

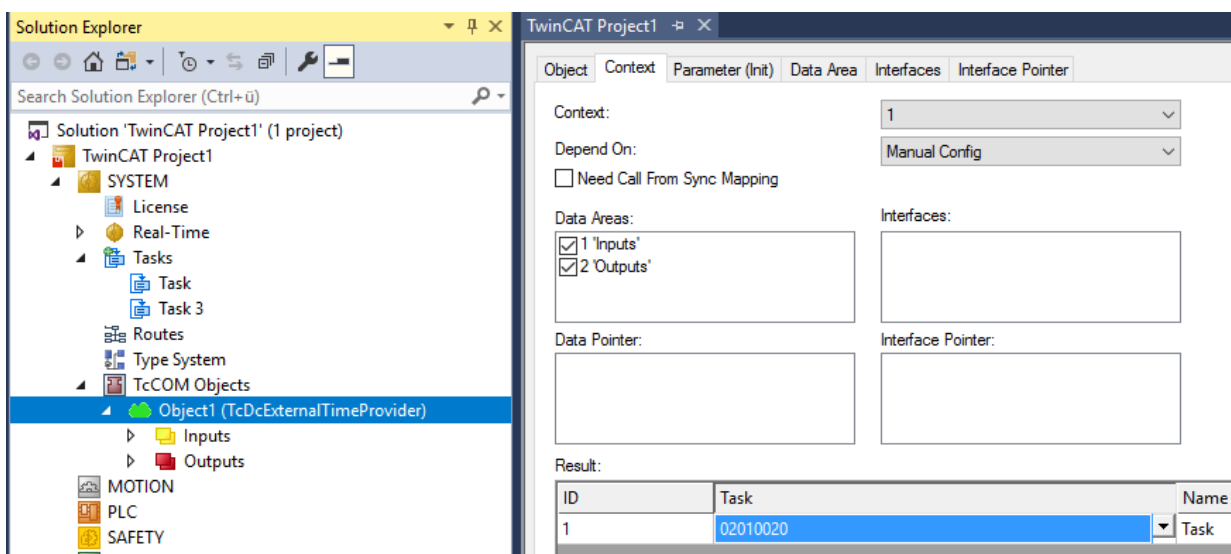
The DC provider is implemented as TcCOM module TcDcExternalTimeProvider. This module is commissioned as a TcCOM module as follows:

- ✓ TwinCAT project

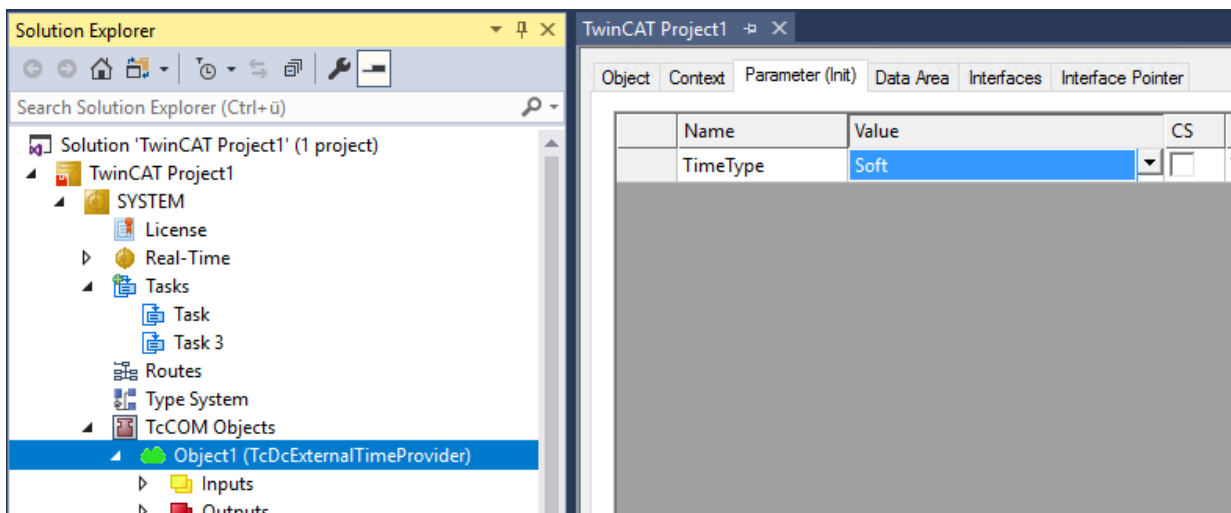
1. Insert a TcCOM module under System->TcCOM Objects and select type TcDcExternalTimeProvider in the category External Time Provider.



2. The module requires a task from which it is called. This is parameterized via the context tab of the module:



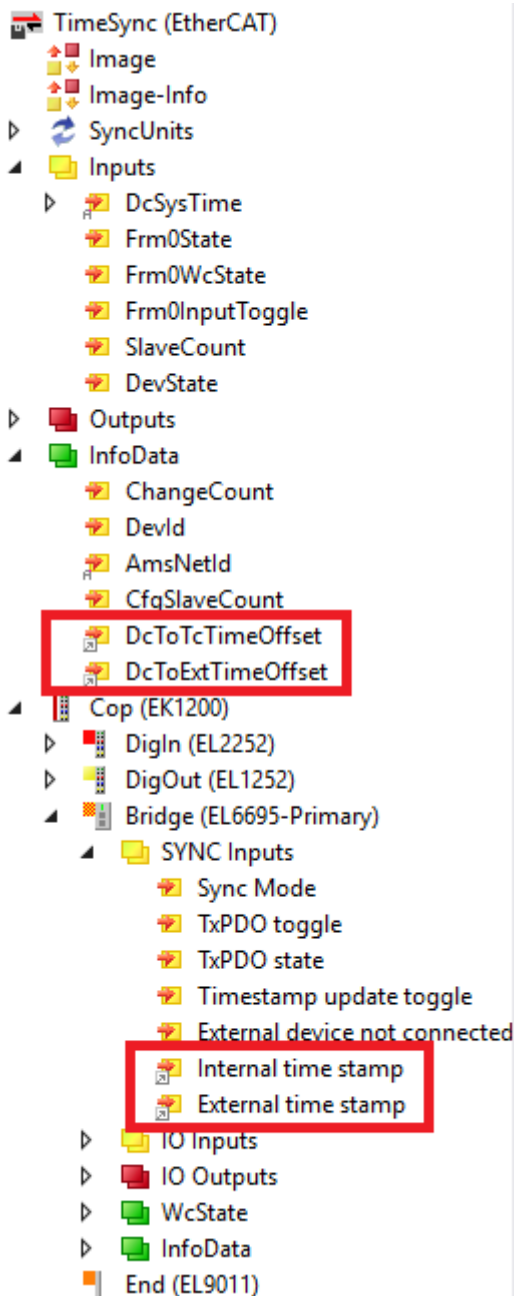
⇒ The module can be parameterized:



The configuration takes place in the Parameters (Init) tab. The parameters have the following meanings:

- TimeType: The type of offset for which this module is to determine an offset.

- This module obtains the offset itself through mapping:



This module passes a determined offset to TwinCAT via the `ITcSetExternalTime [▶ 20]` interface. In addition, outputs are available for mapping.

5.2.3 Application implementation

An application can provide its own TimeOffset provider by using the `ITcSetExternalTime` interface in TwinCAT C++.

This module provides a cyclic value for the respective offsets, if necessary.

Sequence

A module implements the following sequence

- ✓ A TcCOM module was instantiated
- 1. The module registers itself as provider of a certain type of offset (Soft/Medium/Hard) via `RegisterExternalTimeProvider`

2. SetExternalTimeOffset can be used to provide an offset cyclically, if necessary
3. The module logs off using UnregisterExternalTimeProvider

Registration ensures that an offset of only one module can be used at a time.

A more detailed description of the ITcSetExternalTime interface can be found in chapter [ITcSetExternalTime interface](#) [► 20].

6 Real-time API

At this point, interfaces and structures are documented to deal with the corrected timestamps from the real-time.

6.1 Structures

6.1.1 Enum TimeType

TwinCAT provides four different timestamps. The Enum TimeType is used to distinguish between them.

Syntax

```
enum TimeType {
  SystemTime = 0,
  ExternalTimeHard = 1,
  ExternalTimeMedium = 2,
  ExternalTimeSoft = 3, // e.g. NTP
};
```

Values

How the three external timestamp types are used in practice depends on application. The example below is merely a suggestion.

Name	Description
ExternalTimeHard	Suggested use for hard offsets that have no drift
ExternalTimeMedium	Suggested use for accurate offsets such as IEE1588
ExternalTimeSoft	Suggested use for general offsets, such as NTP

6.2 Interfaces

At this point the interfaces are described which are used for the corrected time stamps.

For the different time formats and representations there is a corresponding list in the C++ SDK.

See: Infosys [C/C++](#)

6.2.1 ITcSetExternalTime interface

The ITcSetExternalTime interface is implemented by the TcCOM object server. It can be used to provide an externally determined offset.

Syntax

```
TCOM_DECL_INTERFACE("00000067-0000-0000-e000-000000000064", ITcSetExternalTime)
struct __declspec(novtable) ITcSetExternalTime : public ITcExternalTime
```

Methods

Name	Description
RegisterExternalTimeProvider ▶ 21	Registering a provider for an offset related to TimeType
UnregisterExternalTimeProvider ▶ 21	Logging off a provider for an offset related to TimeType
SetExternalTimeOffset ▶ 21	Provide a new offset for the registered TimeType

Comments

This interface is not available for the PLC.

6.2.1.1 RegisterExternalTimeProvider method

Registering a provider for an offset related to TimeType

Syntax

```
HRESULT TCOMAPI RegisterExternalTimeProvider(OTCID oidProvider, TimeType type) = 0;
```

Parameter

oidProvider: (type: OTCID) The object ID of the provider; normally the object ID of the calling party

type: (type: [TimeType](#) [► 20]) The TimeOffset type to be registered.

Return value

Type: HRESULT

Notifies the success of registration

Description

6.2.1.2 UnregisterExternalTimeProvider method

Logging off a provider for an offset related to TimeType

Syntax

```
HRESULT TCOMAPI UnregisterExternalTimeProvider(OTCID oidProvider, TimeType type) = 0;
```

Parameter

oidProvider: (type: OTCID) The object ID of the provider; normally the object ID of the calling party

type: (type: [TimeType](#) [► 20]) The TimeOffset type to be logged off.

Return value

Type: HRESULT

Notifies the success of the deregistration

Description

6.2.1.3 SetExternalTimeOffset method

Provide a new offset for the registered TimeType

Syntax

```
HRESULT TCOMAPI SetExternalTimeOffset(OTCID oidProvider, TimeType type, __int64 offset) = 0;
```

Parameter

oidProvider: (type: OTCID) The object ID of the provider; normally the object ID of the calling party

type: (type: [TimeType](#) [► 20]) The TimeOffset type

offset: (type: `__int64`) The new offset value.

Return value

Type: HRESULT

Notifies the success.

Description

It is valid for the offset $\text{ExternalTime} = \text{Internal Time} + \text{Offset}$. I.e. if the time in TwinCAT is in the past, the offset must be greater than 0.

6.2.2 ITcExternalTime interface

The ITcExternalTime interface is implemented by the TcCOM object server. This can be used to retrieve and use an externally determined offset.

Syntax

```
TCOM_DECL_INTERFACE("00000066-0000-0000-e000-000000000064", ITcExternalTime)
struct __declspec(novtable) ITcExternalTime : public ITcUnknown
```

Methods

Name	Description
SystemTimeToExternalTime [► 22]	Calculation of a corrected timestamp in relation to the system time
ExternalTimeToSystemTime [► 23]	Calculation of the system time in relation to a corrected timestamp
GetExternalTimeOffset [► 23]	Retrieving an offset in relation to the TimeType
GetExternalTimeProvider [► 23]	Queries the ObjectID of the current provider

6.2.2.1 SystemTimeToExternalTime method

Calculation of a corrected timestamp in relation to the system time

Syntax

```
HRESULT TCOMAPI SystemTimeToExternalTime(TimeType type, __int64& time) = 0;
```

Parameter

type: (type: [TimeType](#) [► 20]) The TimeOffset type to be used for the calculation

time: (type: `__int64&`) The timestamp to be corrected by offset

Return value

Type: HRESULT

Notifies the success.

Description

6.2.2.2 ExternalTimeToSystemTime method

Calculation of the system time in relation to a corrected timestamp

Syntax

```
HRESULT TCOMAPI ExternalTimeToSystemTime(TimeType type, __int64& time) = 0;
```

Parameter

Type: (type: [TimeType](#) [[▶ 20](#)]) The TimeOffset type to be used for the calculation

time: (type: `__int64&`) The corrected timestamp, adjusted by the offset.

Return value

Type: HRESULT

Notifies the success.

Description

The offset valid at the time of the call is used to determine the local system time.

6.2.2.3 GetExternalTimeOffset method

Retrieving an offset in relation to the TimeType

Syntax

```
HRESULT TCOMAPI GetExternalTimeOffset(TimeType type, __int64& offset) = 0;
```

Parameter

type: (type: [TimeType](#) [[▶ 20](#)]) The TimeOffset type to be retrieved

offset: (type: `__int64&`) The value set to the offset.

Return value

Type: HRESULT

Notifies the success.

Description

6.2.2.4 GetExternalTimeProvider method

Queries the ObjectID of the current provider

Syntax

```
HRESULT TCOMAPI GetExternalTimeProvider(TimeType type, OTCID& oidProvider) = 0;
```

Parameter

type: (type: [TimeType](#) [[▶ 20](#)]) The TimeOffset type whose provider is to be queried.

oidProvider: (type: OTCID&) The ObjectID that is set to the ObjectID of the provider.

Return value

Type: HRESULT

Notifies the success.

Description

7 ADS API

The TimeOffsets can also be queried via ADS. There are two ways to do this

1. ADS Notification: ADS notifications contain a time stamp that contains the time at which the data was changed.
An ADS client sends an ADS command before the AddDeviceNotification. This causes the target system to register which type of corrected time stamp is required from this ADS client.
2. ADS Read: A corrected time stamp can be read out via ADS Read. This can be used to obtain a corrected time stamp in an ADS Sum command at the time when the ADS commands were executed.

Index group	Index offset	Access	Data type	Description	Note
ADSIGRP_EXT ERNALTIME 0xF088					
	ADSIOFFS_EX TERNALTIME_ SET 0x0000	R	LONG	Read the currently configured offset type for the respective ADS client (AmsNetAddr incl. client port).	The return value is type 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EX TERNALTIME_ SET 0x00__	W		Set the offset type for the ADSDevice notifications of the respective ADS client (AmsNetAddr incl. client port).	__ is type 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EX TERNALTIME_ OFFSET 0x01__	R	LONGLONG	Reading the current offset for a type.	__ is type: 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EX TERNALTIME_ OFFSET 0x01__	W	LONGLONG	Setting the current offset for a type.	__ is type: 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EX TERNALTIME_ ABSOLUTE 0x02__	R	LONGLONG	Reading the corrected time stamp.	__ is type: 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EX TERNALTIME_ PROVIDER 0x03__	R	ULONG	Reading the object ID from the TimeOffset provider.	__ is type: 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EX TERNALTIME_ SETALL 0x0400	R	LONG	Reads the type that is used if no other type is set.	The return value is type 0 = None, 1 = Hard, 2 = Medium, 3 = Soft
	ADSIOFFS_EX TERNALTIME_ SETALL 0x04__	W		Sets the type that is used if no other type is set.	__ is type 0 = None, 1 = Hard, 2 = Medium, 3 = Soft

The Defines can be found in the file "Ads.h".

The [ADS consumer sample \[▶_27\]](#) illustrates the application.

8 Samples

Various samples illustrating the use of the corrected timestamps are provided for the benefit of the user:

- [PLC Consumer \[▶ 27\]](#): A PLC program accesses corrected timestamps.
- [C++ Consumer \[▶ 28\]](#): A C++ TcCOM module accesses corrected timestamps.
- [ADS Consumer \[▶ 27\]](#): An ADS client in user mode accesses the corrected timestamps.
- [C++ Provider \[▶ 28\]](#): A C++ TcCOM module determines an offset and provides it.

The corrected timestamps are also used by other components of the TwinCAT system. A required configuration can be found with the respective components.

8.1 ADS consumer

The ADS Consumer sample retrieves corrected timestamps as described in the [ADS API \[▶ 25\]](#).

Download

Here you can access the https://infosys.beckhoff.com/content/1033/tc3_Korrigierte-Zeitstempel/Resources/zip/7705550603.zip for this sample.

- ✓ Start the TwinCAT target system with which the ADS Consumer sample is to communicate. The [PLC Consumer \[▶ 27\]](#) sample can be used.

1. Unpack the downloaded ZIP file.
 2. Open the included vcxproj file in Visual Studio.
 3. Adjust the target AmsNetID. (TcExternalTimeAdsClient.cpp, line 119)
- ⇒ The sample is ready for operation.

Description

The sample code can be found in the CPP file TcExternalTimeAdsClient.cpp

Different UseCases for receiving corrected timestamps are illustrated in the Main() method:

- Reading of the provider, the offset and the corrected timestamp from the system service for the different offsets: uncorrected(0), soft(1), medium(2), hard(3), plus an invalid value (4) to illustrate the error behavior.
- Reading the corrected timestamps from a PLC program for the different offsets.
- Reading the provider used and all providers.
- Subscribing to a variable in the PLC; the time provided via notification has a corrected timestamp. The output takes place in the AdsNotificationCallback() method.


8.2 PLC Consumer

The PLC Consumer sample retrieves a corrected timestamp from the TwinCAT system and uses it.

Download

Here you can access the https://infosys.beckhoff.com/content/1033/tc3_Korrigierte-Zeitstempel/Resources/zip/7705583115.zip for this sample.

1. Open the tszip file that it contains in TwinCAT 3 by clicking on **Open Project**
2. Select your target system.
3. Build the sample on your local machine (e.g. **Build->Build Solution**).

4. Activate the configuration by clicking on  .
⇒ The sample is ready for operation.

Description

The TcNtpExternalTimeProvider is configured under **System > TcCOMObjects**. Here you can parameterize your own NTP server under **Parameter (Init)**, if the default pool.ntp.org cannot be reached.

The PLC program essentially consists of the function block FB_TcExternalTime. It provides functions for reading a corrected timestamp from the TwinCAT system. The variable `_eTimeType` represents the type (soft, medium, hard) and can be parameterized.


In MAIN, this function block is used for the eTimeType "Soft" to ensure that the corrected time set by NTP is used.

8.3 C++ consumer

The C++ Consumer sample retrieves a corrected timestamp from the TwinCAT system and uses it.

Download

Here you can access the [https://infosys.beckhoff.com/content/1033/tc3_Korrigierte-Zeitstempel / Resources/zip/7705552907.zip](https://infosys.beckhoff.com/content/1033/tc3_Korrigierte-Zeitstempel/Resources/zip/7705552907.zip) for this sample.

1. Open the zip file that it contains in TwinCAT 3 by clicking on **Open Project**
2. Select your target system.
3. Build the sample on your local machine (e.g. **Build->Build Solution**).
4. Activate the configuration by clicking on  .
⇒ The sample is ready for operation.

Description

The TcNtpExternalTimeProvider is configured under **System > TcCOMObjects**.

Here you can parameterize your own NTP server under **Parameter (Init)**, if the default pool.ntp.org cannot be reached.

The C++ module cyclically determines a local timestamp in the CycleUpdate() method and corrects it. It can be traced in the respective steps using the debugger. The corrected timestamp is provided as a parameter (online).

The type required for this can be configured as parameter "TimeType" in the TcCOM object.


8.4 C++ provider

The C++ provider sample determines an offset and stores it in the TwinCAT system so that it can be used by the consumers.

Download

Here you can access the [https://infosys.beckhoff.com/content/1033/tc3_Korrigierte-Zeitstempel /Resources/ zip/770555211.zip](https://infosys.beckhoff.com/content/1033/tc3_Korrigierte-Zeitstempel/Resources/zip/770555211.zip) for this sample.

1. Unpack the downloaded .zip file.
2. Open the .zip file that it contains in TwinCAT 3 by clicking on **Open Project**
3. Select your target system.

4. Build the sample on your local machine (e.g. **Build->Build Solution**).
5. Activate the configuration by clicking on  .
⇒ The sample is ready for operation.

Description

The offset provider receives the offset to be provided as DataArea "ExternalTime.nOffset". This is transferred to the TwinCAT system as a TimeType medium, which can also be configured at runtime under **Parameter (Init)**.

In the CycleUpdate() method, the SetExternalTimeOffset method is used for this after a corresponding register has been created using RegisterExternalTimeProvider for a TimeType.

9 Appendix

9.1 FAQ

9.1.1 Windows as NTP client

Windows itself offers an NTP client for the system time. In addition, an NTP time can be retrieved using the following script, which is useful for debugging purposes:

```
@echo off
set /p Server=Server:
w32tm /stripchart /computer:%Server% /packetinfo /samples:10
pause
```

9.1.2 Windows as NTP server

Windows itself offers an NTP server to provide timestamps.

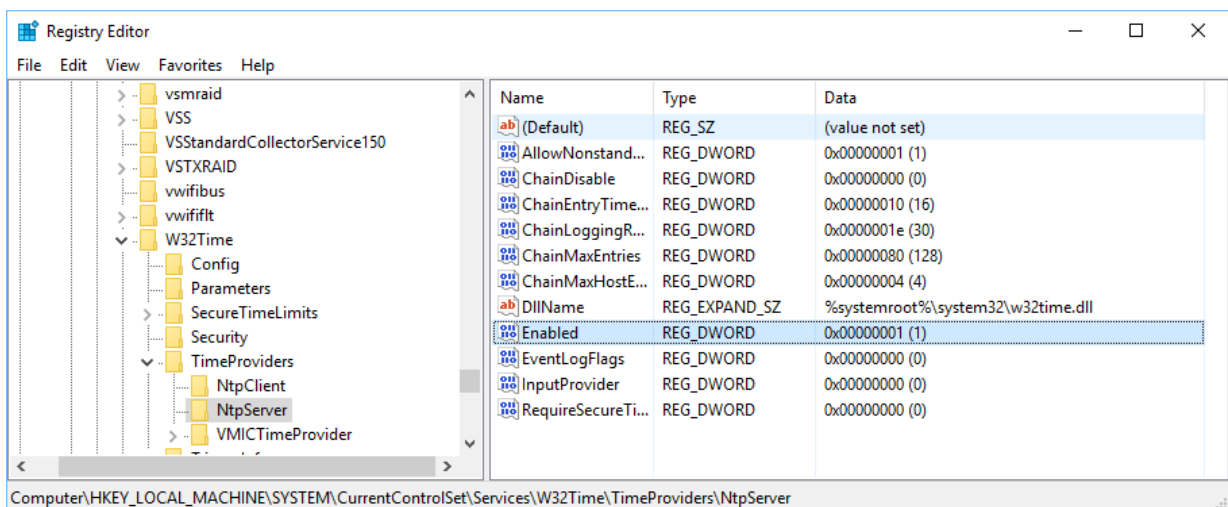
Please note that only one component can use the port for NTP (udp/123). This means that either the [TwinCAT NTP server functionality \[► 12\]](#) or the Windows NTP server can be used.

The Windows NTP server is disabled by default and can be activated later:

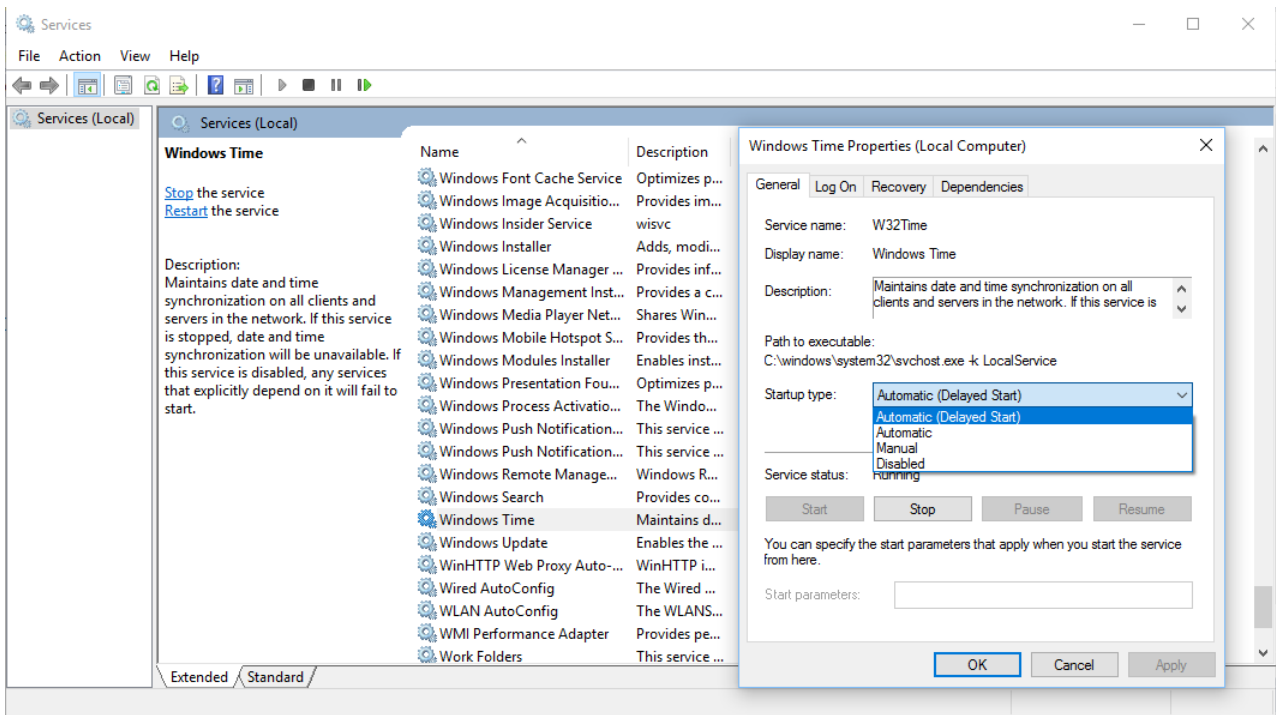
✓ Windows 7 / 10

1. The registry key is set:

```
HKLM\System\CurrentControlSet\Services\W32Time\TimeProviders\NtpServer
Enabled = 1
```



2. The Windows Time system service is started and set to Autostart, if appropriate.



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