

Application Description • 08/2014

Programming an OPC DA .NET Client with C# for the SIMATIC NET OPC Server (COM/DCOM)

SIMATIC NET OPC Server

Warranty and Liability

Note

The Application Examples are not binding and do not claim to be complete with regard to configuration, equipment or any contingencies. The Application Examples do not represent customer-specific solutions; they are only intended to provide support for typical applications. You are responsible for ensuring that the described products are used correctly. These Application Examples do not relieve you of the responsibility of safely and professionally using, installing, operating and servicing equipment. When using these Application Examples, you recognize that we cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these Application Examples at any time and without prior notice. If there are any deviations between the recommendations provided in this Application Example and other Siemens publications – e.g. catalogs – the contents of the other documents have priority.

We do not accept any liability for the information contained in this document.

Any claims against us – based on whatever legal reason – resulting from the use of the examples, information, programs, engineering and performance data etc., described in this Application Example will be excluded. Such an exclusion will not apply in the case of mandatory liability, e.g. under the German Product Liability Act (“Produkthaftungsgesetz”), in case of intent, gross negligence, or injury of life, body or health, guarantee for the quality of a product, fraudulent concealment of a deficiency or breach of a condition which goes to the root of the contract (“wesentliche Vertragspflichten”). The compensation for damages due to a breach of a fundamental contractual obligation is, however, limited to the foreseeable damage, typical for the type of contract, except in the event of intent or gross negligence or injury to life, body or health. The above provisions do not imply a change in the burden of proof to your disadvantage.

Any form of duplication or distribution of these Application Examples or excerpts hereof is prohibited without the expressed consent of Siemens Industry Sector.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens’ products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit <http://www.siemens.com/industrialsecurity>.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit <http://support.automation.siemens.com>.

Table of Contents

Warranty and Liability	2
1 Task.....	4
1.1 Overview.....	4
1.2 Requirements.....	5
2 Solution.....	6
2.1 Solution overview	6
2.2 Description of the core functionality	7
2.3 Hardware and software components used.....	9
2.4 Alternative solutions	10
3 Basics	12
3.1 Principal application model of the OPC DA interface.....	12
3.2 Differences between synchronous and asynchronous read and write jobs	14
3.3 Dividing OPC items into OPC groups	18
3.4 Identifying OPC items created in an OPC client	19
3.5 Function mechanisms of .NET and inclusion of components of the previous programming world.....	21
3.5.1 Programming model of the old world	21
3.5.2 Programming model of the .NET world.....	22
3.5.3 Integration of COM components in .NET applications	23
3.6 Basics on S7 communication	25
4 Functional Mechanisms of this Application	28
4.1 COMDA Client API	30
4.2 Simple Client COM DA.....	32
4.3 S7 program.....	35
5 Configuration and Setting of the OPC Server.....	39
5.1 Configuration of the OPC server in STEP7 V11	39
5.2 Configuring the S7 connections	44
5.3 Check the settings.....	49
6 Installation and Commissioning	51
6.1 Hardware and software installation	51
6.2 Loading the PC station via STEP 7 V1x.....	53
6.3 Importing the XDB file into the Station Configuration Editor	56
6.4 Installation of the OPC client on the PC/PG.....	59
6.5 Loading the simulation to the S7 stations	60
7 Operating the Application.....	61
7.1 Overview.....	61
7.2 Using the block services.....	63
8 Glossary	64
9 Related Literature	65
9.1 Bibliographic references.....	65
9.2 Internet link specifications	66
10 History.....	66

1 Task

1.1 Overview

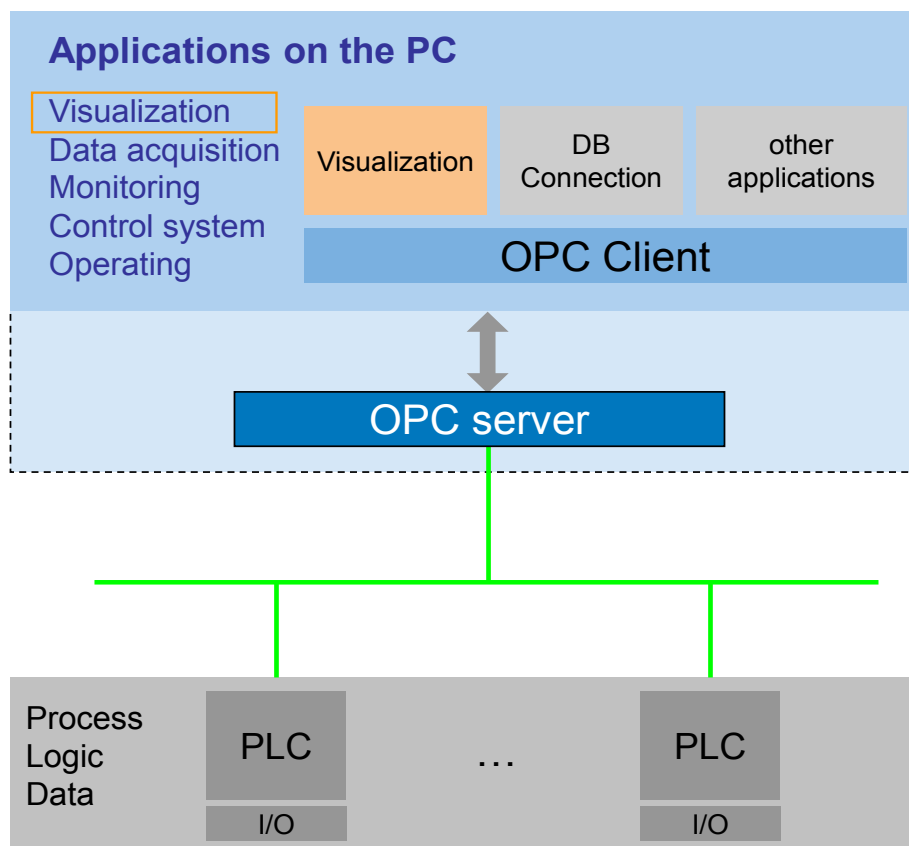
Introduction

This application example shows the coupling of a production process to a Windows-based PC with a data exchange via OPC. With this principle of operation, for example, separate, specialized user interfaces and process visualization or data acquisition can be realized.

Overview of the automation task

The figure below provides an overview of the automation task.

Figure 1-1



Description of the automation task

A process is simulated with two different PLCs. The data from this process is to be displayed, recorded and modified. This application shows how process data can be displayed on a PC – using the .NET programming environment – irrespective from which process control this data is.

1.2 Requirements

Requirements for operating and monitoring software for visualizing

The software is to enable the fast and simple generation of an interface. For this purpose, the following requirements have to be met:

- Expandable library with interface controls.
- Use of Windows standard controls.
- Simple, reusable connection of these controls to the data.

Requirements to the data interface between visualizing and controlling

- Connection to the process data via Industrial Ethernet as well as the SIMATIC NET OPC Server V8.x.
- Use of OPC DataAccess Custom interfaces RCW.
- Symbolic and absolute addressing of process data.
- Asynchronous/synchronous reading and writing of individual process data.
- Writing and reading of large data volumes via block services.
- Implementation of a structure for error handling.

Requirements to the development environment to be used

The current Windows development environment is to be used:

- Use of Microsoft Visual Studio® .NET 2010 SP1.
- Use of the .NET programming language Visual C#

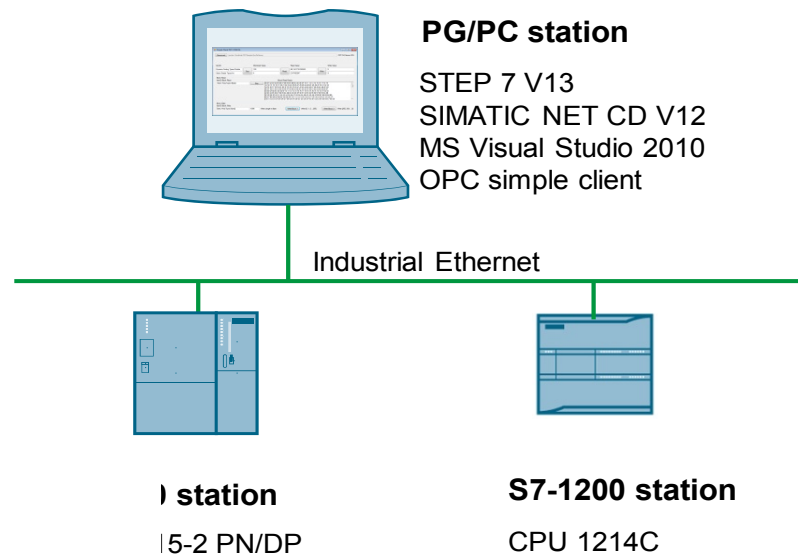
2 Solution

2.1 Solution overview

Schematic layout

The following scheme shows the most important components of the solution:

Figure 2-1



S7 station

The controller side consists of two S7 stations: a CPU 315-2 PN/DP and an S7-1214C

PC station

A PC station is connected via a standard Ethernet network card to a S7-300 controller and a S7-1200. On the PC station the SIMATIC NET OPC server and the OPC client is running. A very simply designed client (Simple OPC Client) shows you all basic functions for getting started.

The functionality of these sample clients will be explained in the next section.

Advantages

This application offers the following advantages:

- A complete C# programming example which shows you all important OPC mechanisms.
- A reusable, expandable class library in which the most important OPC methods for individual applications are encapsulated.
- A simplification of the general ".NET RCW" of the OPC Foundation to an easily usable and expandable library for ".NET".

Delimitation

This application does not contain a complete description of the following topics:

- ".NET" Framework
- C#
- OPC specification
- Deeper level COM mechanisms

Basic knowledge of these topics is assumed.

Assumed knowledge

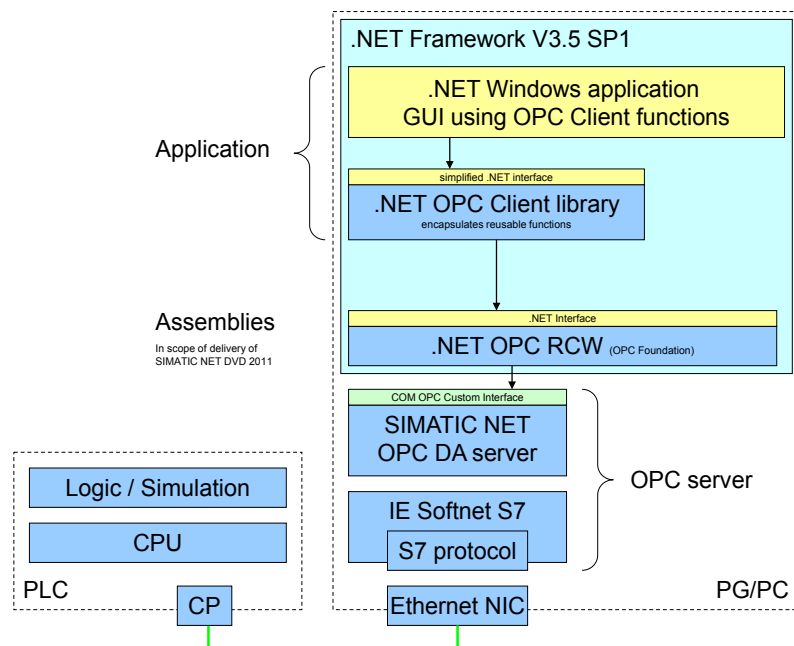
Basic knowledge in the area of object-oriented programming, as well as in the COM environment is assumed. Further knowledge in UML (Unified Modelling Language) is an advantage.

2.2 Description of the core functionality

Software components involved

The following figure shows the involved software components. Precise descriptions can be found in chapter 4.

Figure 2-2



PC/PG

A C# OPC client has been realized on the PC/PG for visualizing.

For connecting to the process, the OPC client uses the OPC Runtime Callable Wrapper (abbr.: RCW) that is automatically installed together with the SIMATIC NET OPC server.

The SIMATIC NET OPC server establishes the connection to the controller via the SIMATIC NET SOFTNET-S7 connection.

Control

The controller provides the data to be visualized.
 For this purpose, a simple S7 program for simulating various data types is implemented.

Created software components

- C# OPC client
- STEP7 simulation program

General application procedure

The OPC SimpleClient shows access to OPC data points and the use of the tags as well as block services of the OPC server. The client library is expandable:

Table 2-1

Action	User interface
<p>The following functions are implemented in the OPC SimpleClient: Connection to the OPC server:</p> <ul style="list-style-type: none"> • Connecting with the OPC server. • Creating groups and items. • Monitoring values. • Reading/writing values. • Reading/writing block. 	

Note The actions are performed via the operator elements of the user interface.

2.3 Hardware and software components used

The application was created with the following components:

Hardware components for the controller

Table 2-2

Component	No.	Article number	Note
PS307 5A	1	6ES7307-1EA00-0AA0	
CPU 315-2 PN/DP	1	6ES7315-2EH14-0AB0	or a comparable S7-300 CPU
S7-1200 PM 1207	1	6EP1332-1SH71	
CPU 1214C DC/DC/DC	1	6ES7 214-1AE30-0XB0	or a comparable S7-1200 CPU.
Standard switch	1	Depending on product	

Hardware components for the PC

Table 2-3

Component	No.	Article number	Note
Power PG	1	6ES7751-0EA31-0LB3	or similar PC/PG
NDIS-capable network card	1	Depending on product	Integrated in Power PG

Standard software components

Table 2-4

Component	No.	Article number	Note
STEP 7 Professional V13	1	6ES7822-4AA03-0YA5	
WinCC Professional	1	6AV2103-0DA00-0AM0	Adjust the article number according to the required power tags.
SIMATIC NET IE SOFTNET-S7 (V12)	1	6GK1704-1LW12-0AA0 6GK1704-1CW12-0AA0	LW=8 S7 connections (Lean), CW=64 S7 connections
Microsoft Visual Studio 2010	1	Express Edition Standard Edition Professional Edition	Obtainable in the Microsoft store (http://emea.microsoftstore.com)
.NET Framework 3.5	1	Free download at http://www.microsoft.com/	Installed by SIMATIC NET

Sample files and projects

The following list includes all files and projects that are used in this example.

Table 2-5

Component	Note
21043779_OPCCClient_RCW_CODE.zip	The file contains the archived STEP7 simulation program for STEP7 V11 and STEP7 V13, the setup for installing the OPC clients (without .NET Framework) and the complete source code files
21043779_OPCCClient_RCW_DOKU_V2_1_e.pdf	This document

2.4 Alternative solutions

OPC solutions in various programming languages

For coupling of process and operator control & monitoring, OPC is used as standard mechanism. Depending on the preferred programming language, there are various solutions that differ mainly in view of the speed of programming, type of target application and their requirements. Furthermore, it should be distinguished what OPC interfaces are to be used: the classic COM-based interfaces OPC DA2, DA3, A&E or the combined OPC UA interface.

Decision criteria for using different programming languages

The following table shows you the most important decision criteria which are decisive for selecting the used programming language.

Table 2-6

Deciding criteria	Description
Custom interface	Is it possible to use the custom interface of the OPC DA interface?
Automation interface	Is it possible to use the automation interface of the OPC DA interface?
Simple implementation	How suitable is the language to implement the code relatively simply?
Error handling	How well is the language suitable for realizing professional error handling?
Performance	How well is the language suitable for developing performant OPC applications?
Parallelity	How well suitable is the language to process tasks (from different threads) in parallel?

Comparing different programming languages

The following table compares the different programming languages using the above described criteria. The variant realized in this example is highlighted in blue in this table.

Table 2-7

Criterion	VBA (MSOffice)	VB V6.0	VC++	.NET languages (managed)
Custom interface	-	-	✓	✓(with RCW)

2 Solution

Automation interface	✓	✓	✓ (*)	✓(*) (with RCW)
Simple implementation	++	++	+ (**)	++
Error handling	-	-	++	++
Performance	-	-	++	+
Parallellity	-	-	++	+

(*) should not be used

(**) whilst using the Active Template Library (ATL)

Deciding criteria for using OPC interfaces

The following table shows you the most important decision criteria which are crucial for selecting the used OPC interface:

Table 2-8

Deciding criteria	Description
COM/DCOM OPC DA 2/3	Is the use of OPC DA interface with reading/writing/monitoring sufficient for the application?
COM/DCOM OPC A&E	Is the processing of events and process alarms pursued (for the future)?
OPC UA	Is a combined function (read/write/alarms/methods/objects/types) with expanded functions required?
Security	Should remote connections be established and operated securely across firewall boundaries?
Platforms	Is the application only to communicate with Windows-based systems (no Linux, Android, iOS)?
Functionality, flexibility	What functionality is provided by the interface? In how far are the functions to be expanded?
Combination with other manufacturers	How suitable is the interface for integrating the systems of other manufacturers, what remains, what is to be configurable?

Comparison of various OPC libraries for .NET languages

In order to simplify the implementation of OPC Client application in the .NET languages, there are various libraries for different application areas. The following table compares the libraries based on the above described criteria.

Table 2-9

Criterion	OPC Connector	OPCClient.API	OPCda.RCW	OPCUa.RCW
COM DA Interface	✓	✓	✓	-
COM AE Interface	-	-	-	-
OPC UA Interface	✓ (*)	✓ (*)	-	✓
Security	- (**)	- (**)	- (**)	++
Platforms	-	-	-	✓
Functionality, flexibility	-	+	++	+++
Other manufacturer	- (***)	- (***)	✓	✓

(*) only DA functions

(**) security only via DCOM

(***) classic OPC DA and OPC UA parallel, limited to SimaticNET OPC server

3 Basics

3.1 Principal application model of the OPC DA interface

Introduction

This section only briefly discusses the application model of the OPC DA interface. A detailed description of the object model is available in the OPC DA specification of the OPC foundation.

OPC Client

The OPC client tells the OPC server that it wants to have access to certain tags by the OPC client creating references to the respective interfaces.

Further interfaces available via the "IOPCGroupStateMgt" group(s) interface enable read and write access to these tags. The following access methods are possible:

- Synchronous reading / writing
- Asynchronous reading / writing
- Monitoring tags (reporting value changes).

Note

Single threaded OPC clients are blocked for synchronous read / write jobs. For asynchronous read / write jobs, the OPC client remains operable ("responsive").

For more information see chapter 3.2.

OPC server

The OPC server is the central communication unit between an OPC client and the respective controller.

Via COM/DCOM mechanisms, it provides standardized interfaces towards the OPC client which allow each COM-capable application to access tags of any controller.

In this application, the DataAccess interfaces for generating the logic connections between OPC items and process tags are decisive.

The OPC server is connected to the respective controller via the implementation of the communication protocols.

Connection between OPC server and controller

The values of the tags are read out via the connections configured in the OPC server using the respective protocol blocks. In this example, the S7 protocol is used that contains two principle transmission mechanisms, the tag services and the block services. The connections are configured in the TIA portal (see chapter 5.2).

Reading tags from the controller

You can read out and update the tags in two different ways:

Table 3-1

Service	Description
Tag service	For tag services, one or several process tags are specified by means of absolute or symbolic addressing. Thereby, the tags to be monitored can be polled cyclically from the OPC server (→Polling, see Glossary). Nothing has to be programmed in the controller here. The communication is performed via system-internal processes.
Block service	If the tags to be monitored are transferred to the OPC server in a program controlled manner by means of larger data blocks, then this is referred to as block service. The controller calls a communication block (BSEND) in its execution program in order to actively trigger the data transmission. The OPC server receives the data in its receive buffer. In doing so, not the process tags themselves are addressed, but rather the data areas, such as, for example, a data block. Cyclic "monitoring" of the receive buffer by the OPC server only makes sense here for receive items (e.g. "receive").

Notes on the services

- In this application tag services and block services (only S7-300) are used.
- The absolute addressing only differs from the symbolic addressing by the ITEM_ID.
- The tag and block services on the OPC client side are only differentiated by its ITEM_ID.
- Here, "block services" only refers to the services ("SEND/RECEIVE", "BSEND/BRECEIVE") provided by the SIMATIC NET OPC server.

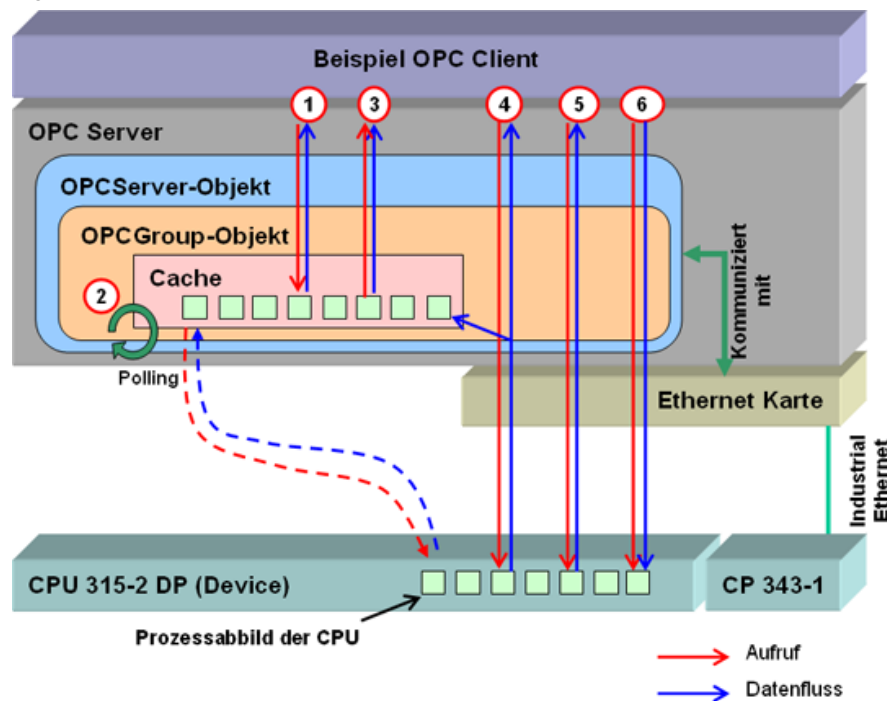
3.2 Differences between synchronous and asynchronous read and write jobs

The differences between synchronous and asynchronous reading and writing of OPC items are discussed below. We will in particular explain the difference between accesses to the "DEVICE" and the "CACHE".

Function principle of synchronous/asynchronous read and write jobs

The following image gives you an overview of the function principles of the respective calls. The function principle is identical with the PROFINET interface.

Figure 3-1



Explanations on the figure

- The red arrows show the call direction, the blue arrows the resulting data flow.
- Data is exchanged between the OPC server and the controller using an Industrial Ethernet card and the internal interface of a PN-CPU or a CP343-1.
- Data exchange between OPC client and OPC server occurs as inter-process communication (here: COM).
- "Cache" refers to a temporary buffer which the OPC server generates for a particular group. It contains a local image of the process tags defined for this group (which in return are managed by the group as OPC items).
- "Device" refers to the process tags on the controller.

Explanation of the processes

The following table explains the respective calls and data flows. In the process, the following consequences are pointed out:

Table 3-2

No.	Action	Remarks
1	Synchronous read job on the CACHE	The value of the OPC item is read with the value currently located in the CACHE.
2	Updating the CACHE	<p>The OPC server updates the entire CACHE with the values from the DEVICE after the time period defined with the "requestedUpdateRate".</p> <p>Note:</p> <p>The requestedUpdateRate or also the returned update time by the OPC server ("revisedUpdateRate") does not have to match the actual update time. Please note that the actual update time is by no means shorter than the revisedUpdateRate.</p>
3	OnDataChange Event	<p>For OPC items parameterized as "active", the OPC server has registered a data change within its CACHE (new polled value <> cached value). These changes are reported to the OPC client and the new value is copied to the CACHE.</p> <p>Note:</p> <p>This mechanism is ideal for monitoring process tags. Please note that a DataChange event also triggers a change of status just as a change of value. The update rate corresponds to the actual "updateRate". The DataChange event only informs of the changed items.</p>
4	Asynchronous read job to the DEVICE	<p>The OPC client starts an asynchronous read job via the OPC group. The OPC server reads the requested process tags from the process image of the S7-CPU and delivers them to the OPC client via the OnReadComplete event. It then writes the read values to its CACHE</p> <p>Note:</p> <p>The OPC client continues being operable for asynchronous calls while the respective job is processed ("responsive").</p>
5	Synchronous read job on the DEVICE	<p>The value of the OPC item is read with the value currently located in the process image of the S7-CPU.</p> <p>Note:</p> <p>This access may take several seconds depending on physics, data traffic and data volume.</p> <p>During this time, the thread of the OPC client, which starts the synchronous read call, cannot perform any other tasks! Single threaded OPC clients are hence blocked during this time and cannot execute further operating and monitoring tasks.</p>

6	Synchronous or asynchronous write job	<p>Write jobs are only executed to the DEVICE, irrespective of whether they are synchronous or asynchronous.</p> <p>Note:</p> <p>This access may take several seconds depending on physics and data volume.</p> <p>If a synchronous write job was triggered, the same behavior as described in point 5 takes place; for an asynchronous call the OPC client behaves as described in point 4.</p>
---	---------------------------------------	---

Access to CACHE and DEVICE

The following table summarizes what operations are possible on "DEVICE" and "CACHE".

Table 3-3

Operation	DEVICE	CACHE	Note
Synchronous reading	✓	✓	Reading from the CACHE requires the group and the respective item to be active.
Asynchronous reading	✓	-	The OPC server reads the tag from the DEVICE. This also updates its CACHE.
Synchronous writing	✓	-	Write jobs are always written to the DEVICE.
Asynchronous writing	✓	-	Write jobs are always written to the DEVICE.
Monitoring (event controlled)	-	✓	Monitoring of tags is only possible using the CACHE.

Note

Apart from read and write jobs, further methods are also available (such as e.g. IOPCAsyncIO2::Refresh) which, however, were not used in this application.

For further information see OPC Specification of the OPC Foundation.

Comparison of application cases

The following table contains a comparison of different application cases and shows the recommended procedure:

Table 3-4

Operation	Cyclic/ Acyclic	Data volume	DataChange mechanism	Sync	Async
Read	Cyclic monitoring	Large	+	*	*
		Small	+	*	*
	Acyclic	Large	-	-	+
		Small	-	+	-
Write	Acyclic	Large	-	-	+
		Small	-	+	-

(*) The onDataChange mechanism is executed asynchronous to the CACHE

Note

Large data volumes in the range of several kilobytes should be read or written with the block services, e.g. "BSEND/BRCV". Here, the SIMATIC works as active communication partner.

3.3 Dividing OPC items into OPC groups

Below, you learn about the criteria to be taken into consideration when dividing the OPC items to OPC groups.

Motivation of dividing OPC items

As already explained in chapter 3.2, certain ways of behavior result from certain call types.

It now makes sense to combine process tags or OPC items with a similar operating or monitoring behavior into groups.

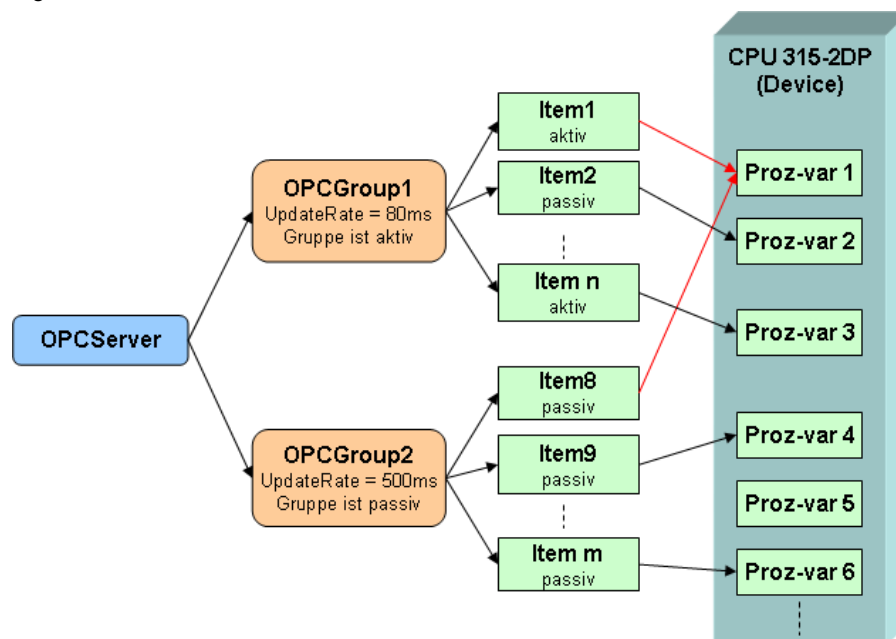
In some cases, however, it is desirable that a process tag is to be monitored at two different locations of the OPC client (such as, for example, in two different pages or dialogs). Another example for this is that a process tag is monitored at one location, but only sporadically (i.e. acyclically) read out at another location.

In these cases it makes sense to create one and the same process tag in several groups.

Dividing process tags to OPC items and OPC groups

The following figure illustrates this option with two different groups. Process tag 1 is mapped to two OPC items, which are stored in two groups. This makes it possible to achieve different ways of behavior for a process tag in an OPC client.

Figure 3-2



3.4 Identifying OPC items created in an OPC client

In order for the process tags mapped to OPC items to be read or written by an OPC client, a unique assignment of OPC items to an OPC client must exist.

The concept of identifiers or handles is used here. It facilitates a more efficient transmission and faster identification of the items for access operations.

Handle types

Two handle types are distinguished. This differentiation ensures that the OPC client as well as the OPC server can identify the respective OPC items.

Table 3-5

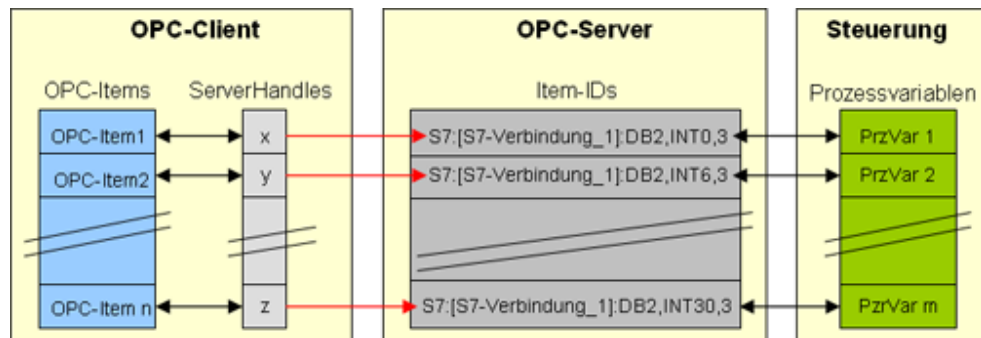
Type	Description
Client handles	These handles are created by the client and are used for identification of an OPC item within the OPC client . When creating the items (AddItems) the client tells the server its client handles.
Server handles	These handles are created by the server and are used for identification of an OPC item within the OPC server . When creating the items (AddItems) the server gives the client back its server handles.

Server handles

If the call direction is from OPC client to OPC server (e.g. write), the OPC client must transfer the respective server handles to the OPC server.

Example: write into the OPC item!

Figure 3-3

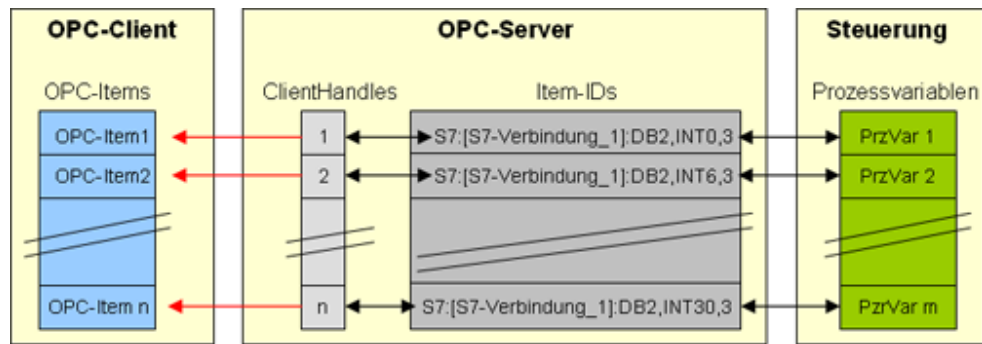


Client handles

If the call direction is from OPC server to OPC client (e.g. onDataChange), the OPC client will receive the respective client handles from the OPC server.

Example: The OPC items in the CACHE have changed. → The onDataChange event is triggered.

Figure 3-4



Effects on the client program

This structure implies that an OPC client must manage the server and/or client handles depending on the application case.

This way, for example, the client handles may, for example, serve as an index of the respective OPC item in an OPC item array.

The server handles should either be encapsulated in a separate server handle array or in an independent OPC item management class.

3.5 Function mechanisms of .NET and inclusion of components of the previous programming world

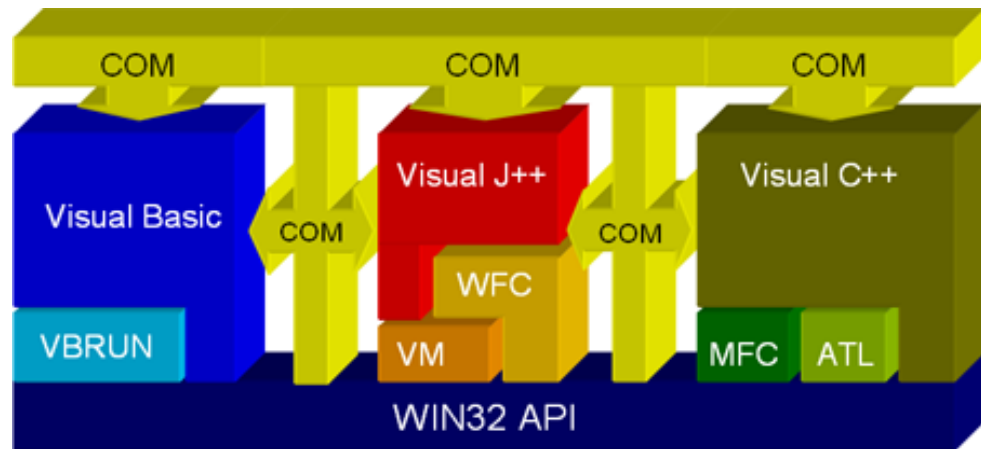
The previous, as well as the latest Windows programming world will be discussed first now. Then the differences to be noted are explained.

It is only a brief introduction to help to understand the "how to program with .NET" as well as "how to combine the new and the old world?". Further information is available in the secondary literature.

3.5.1 Programming model of the old world

Overview

Figure 3-5



API

The previous programming model of the Windows world builds on an **Application Programmers Interface (API)**. The API provides a lot of C functionalities for accessing the operating system resources and functionalities.

Programming languages and libraries

Using various programming languages with partially extensive libraries (**VBRuntime**, **Microsoft Foundation Classes**, **Active Template Library**) it was possible to implement Windows programs. The libraries respectively encapsulated a part of the API functions.

Interacting of the languages with COM

In order for the developed program components to be able to work with each other, **Component Object Model (COM)** by Microsoft was introduced. Using standardized interface definitions made it possible to establish a "connecting glue" between the individual components. The concept of **Distributed Component Object Model (DCOM)** was developed in order to expand the cross-language and cross-component interaction to a cross-computer interaction.

Note

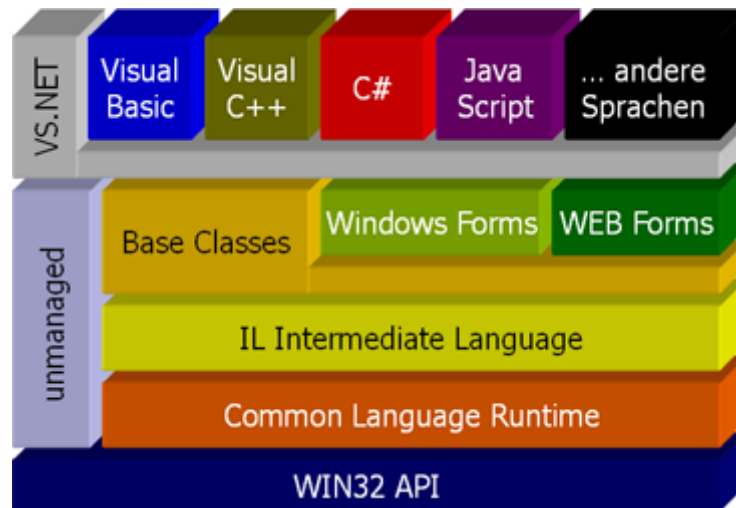
The OPC server works as COM component.

→ OPC clients access the OPC server via COM mechanisms.

3.5.2 Programming model of the .NET world

Overview

Figure 3-6



API

The basis of the model is Win32 API like in the previous Microsoft programming world.

CLR

Based on the API, follows the **Common Language Runtime (CLR)**. It constitutes a runtime environment whose model is comparable with a Java Virtual Machine. It compiles **Just In Time (JIT)** a type of byte code in X86 code (depending on the respective processor architecture); Microsoft speaks of IL code (**Intermediate Language**).

Intermediate Language, Garbage Collector and unmanaged Code

The advantage of the IL code is the fact that it is independent of the platform. It is also possible, for example, to execute this code on Linux (Unix) systems, if a CLR has been implemented there (see \6).

The CLR does not only perform the JIT compiling of the IL code, but also the entire memory management. Similar to Java, a **Garbage Collector (GC)** was developed, who releases un-referenced memory areas within the CLR after an unspecified time. Code that is subjected to the access of the Garbage Collector is also referred to as "managed Code".

In order for it to be possible in certain cases to still control the release of memory areas manually, the developed code can be declared as "unmanaged". This enables protecting certain code areas or entire programs explicitly from the access of the Garbage Collector.

Base classes

Regarding the libraries of partly varying extend of the old world, Microsoft has now implemented a uniform base class library which can be accessed independently from the .NET programming language.

Integrator Visual Studio .NET

Visual Studio .NET (VS .NET) takes on the task of combining the different .NET languages with each other.

This enables realizing .NET components in different languages. Interactions between .NET components are easily possible with Visual Studio .NET.

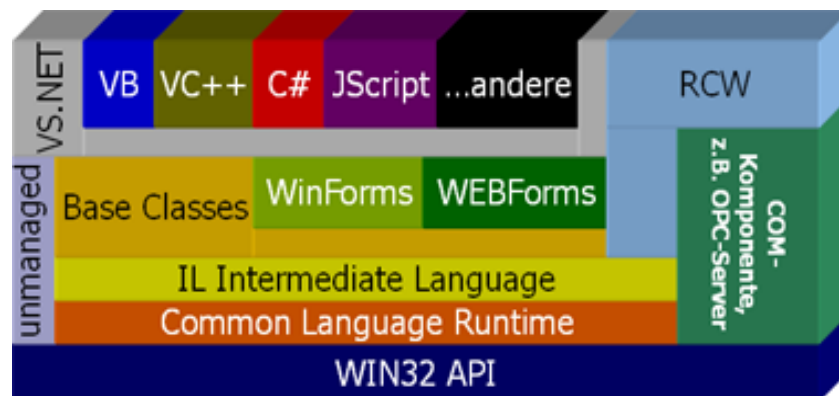
Easy-to-handle integration options therefore make a faster development within the .NET components possible.

3.5.3 Integration of COM components in .NET applications

Overview

Using a COM component in a .NET application/component:

Figure 3-7



Runtime Callable Wrapper (RCW) and the relationship with the COM component

Installing a COM component generated for the previous Windows world within a .Net application, requires a "wrapper", a type of case which encapsulates all interface definitions for the .NET application.

This is necessary as the previous interface definitions located in so-called IDL files (Interface Definition Language, see Glossary) are no longer supported by .NET. This wrapper is also referred to as **R**untime **C**allable **W**rapper (RCW).

For COM components offering the automation interface, Visual Studio can generate these wrappers automatically.

However, for COM components providing the custom interface, such wrappers must be generated manually.

For the Custom Interfaces (e.g. OPC DA 2 / 3) defined by the OPC Foundation, the wrappers were created manually by the OPC Foundation and provided for general use. These .NET OPC RCWs are, for example, installed with the SIMATIC NET CD.

Note The differences between automation and custom interface are not discussed here in greater detail. This requires secondary literature [2].

Memory management when using COM components

The memory management is performed by the GarbageCollector for .NET applications. However, since COM components demand an explicit memory management, the following has to be observed for the data exchange between the two components:

Table 3-6

Direction	Description
.NET to COM	In .NET, all tags are objects. Due to the fact that generally speaking, COM servers do not synchronize with COM clients, transfer values should be protected from the access of the Garbage Collector ("pinning" of objects).
COM to .NET	COM components deliver returned values in form of COM pointers. Due to the fact that in the managed code of .NET clients such pointers do not exist, the returned values must first be saved into .NET objects. This is performed via the .NET data type "IntPtr" and methods / objects of the "System.Marshal" classes.

RCW and OPC

Due to the OPC specification, OPC components offer a custom interface, i.e. an RCW must be generated manually for performant applications.

For the SIMATIC NET OPC server a RCW is part of the delivery for the DataAccess interface V2.05 and for the V3.0 which has to be integrated into the respective Visual Studio project.

If the automation interface of OPC is used with .NET, then the RCW is automatically generated by VS .NET as soon as a reference to the OPC automation interface is included into the .NET project. The resulting double encapsulation of the OPC custom interface however, may cause a reduction in performance.

Note Since a performant connection is the focus of this application, this application deals with the usage of the custom interface.

Note The reverse application case – using a .NET component as COM component – is possible using a **Com Callable Wrapper (CCW)** This however, is not further discussed within the framework of this document, as it is not necessary for the comprehension of this application.

3.6 Basics on S7 communication

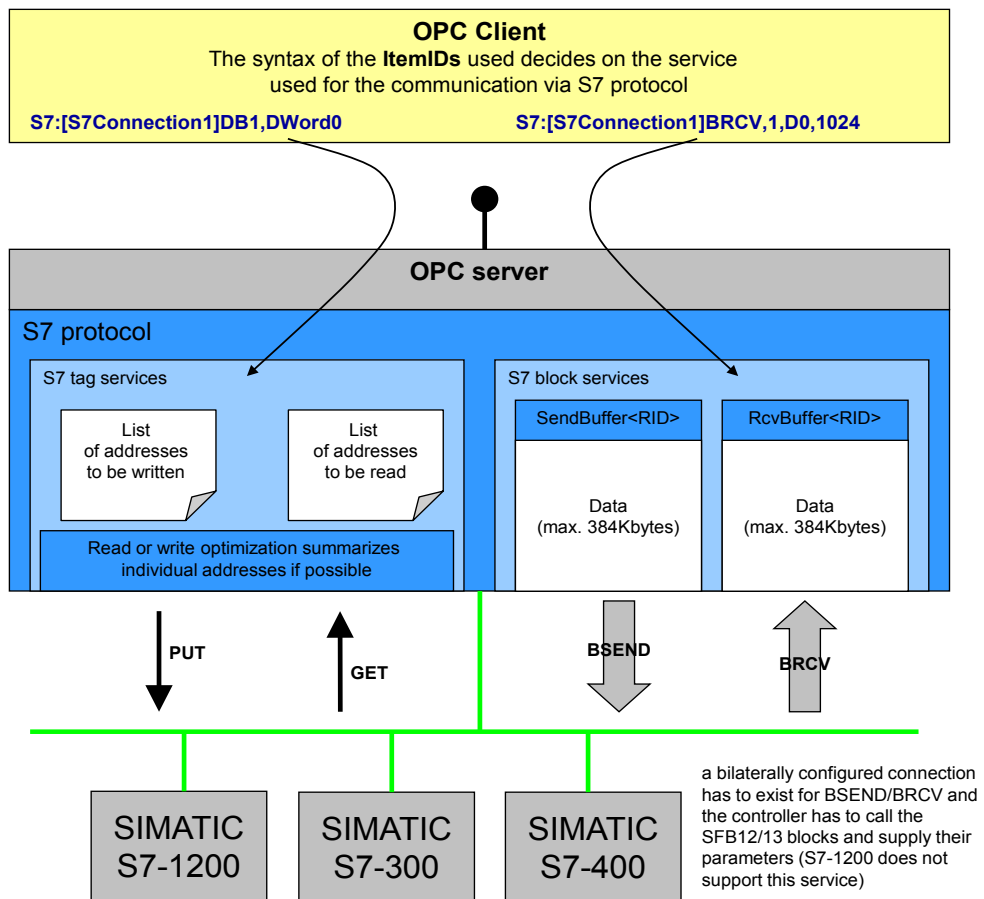
The “classic” OPC server (COM/DCOM) is available for all other protocols such as DP, FDL, SR or SNMP. On the SIMATIC NET CD V8.0 there is additionally an OPC UA server. The OPC UA server is not a part of this description.

Functional chain of the communication

The S7 communication is divided into two very different communication services, into tag services and block services. On the level of the OPC server they are almost completely covered up. Which communication service is used for the controller, can only be detected based on the ItemId. The “S7:” protocol prefix specifies that it is a direct addressing type. The symbolic access uses the “SYM:” prefix.

Internally, the OPC server separates the ItemId into its components and based on its structure, detects via which communication service, communication to the S7 controller is to take place. Here, the connection name identifies the communication partner (inter alia, this name represents e.g. an IP address) and the key word “BRCV” or “BSEND” causes the use of the block services instead of the tag services. The S7 type identifier and the offset address indicate the position of the data within the controller and the data type specifies the interpretation this data.

Figure 3-8



© Siemens AG 2014 All rights reserved

Tag services

A S7 controller replies to requests via tag services, for this purpose only a unilaterally configured connection is necessary. Each S7 controller is a so called "S7 server" and answers PUT/GET requests without the need of any implementation in the control program of the PLC. All data areas of the controller can be directly accessed (I, Q, M, DB, etc.). This communication service is very flexible and, above all, easy to use.

ItemIDs for tag services

S7:[<connectionname>]DB<no>,<type><address>,<quantity>}

Example: S7:[S7 connection_3]DB10,W20

A tag of the word type (16bit no signs), which is located in data block 10 and which starts at the byte offset address 20 (meaning it consists of bytes 20 and 21). This tag is retrieved with Put/Get via the connection called "S7 connection_3", meaning from the S7 controller which is hidden behind this connection.

Symbolic ItemIDs

Apart from the direct addressing, there is the option of symbolic addressing. For this purpose, the address space is generated from STEP7 (TIA Portal). For all symbolic identifiers of the data points in the S7 controllers which are connected with an OPC server via a S7 connection, a symbol export can be triggered. The thus generated symbols file with the ending ATI is introduced to the OPC server via download from STEP7 or via XDB import. The ATI file (Advanced Tag Information) contains an image of the symbolic name for the direct addresses.

Note

All symbols are eventually retrieved via PUT/GET from the controllers. Symbols which represent a BSEND or BRCV tag cannot be generated by STEP7.

Block services

For the exchange of large data volumes the more effective block service is available. On a bilaterally configured connection, large data volumes (up to 64kbytes) can be exchanged. Communication is based on the exchange of data buffers. However, the respective system function blocks (BSEND/BRECV) have to be called in the control program for this purpose. The OPC server provides the respective counterparts to the PC if the corresponding OPC items have been created.

Structure of ItemIDs for block services

S7:[<connectionname>]BRCV,<RID>,<type><address>,<quantity>}

Example: S7:[S7 connection_5]brcv,3

The complete receive buffer for the BSEND/BRECV pair with ID 3 which is connected via the connection named "S7 connection_5" is represented in a byte array for OPC. This byte array always contains the data last sent from the communication partner with BSEND (on the other side of the "S7 connection_5"). On a S7 connection, several BSEND/BRECV pairs belonging together can exist which are connected via their RID. Here, it is the BRECV which belongs to BSEND with ID 3.

S7:[<connectionname>]BSEND<length>,<RID>,<type><address>,<quantity>}

Example: S7:[S7 connection_2]bsend1024,1,W100,20

When writing on this NodeID, an array of words (unsigned integer 16 bit) with 20 elements from the byte offset address 100 is written in the send buffer with a length of 1024. The range of 100 to 140 is overwritten in the 1024 byte size buffer.

The entire block is sent with ID 1 to the communication partner who has to provide a BRCV with ID 1 and a minimum length of 1024 bytes to be able to receive the data.

Note

To be able to use the BSEND/BRCV block services, a bilaterally configured connection has to exist and the controller has to independently call the SFB12/13 blocks and supply their parameters.

Also read the notes in the SIMATIC NET manual regarding the subject of block-oriented services (see \7).

4 Functional Mechanisms of this Application

General overview

Figure 4-1

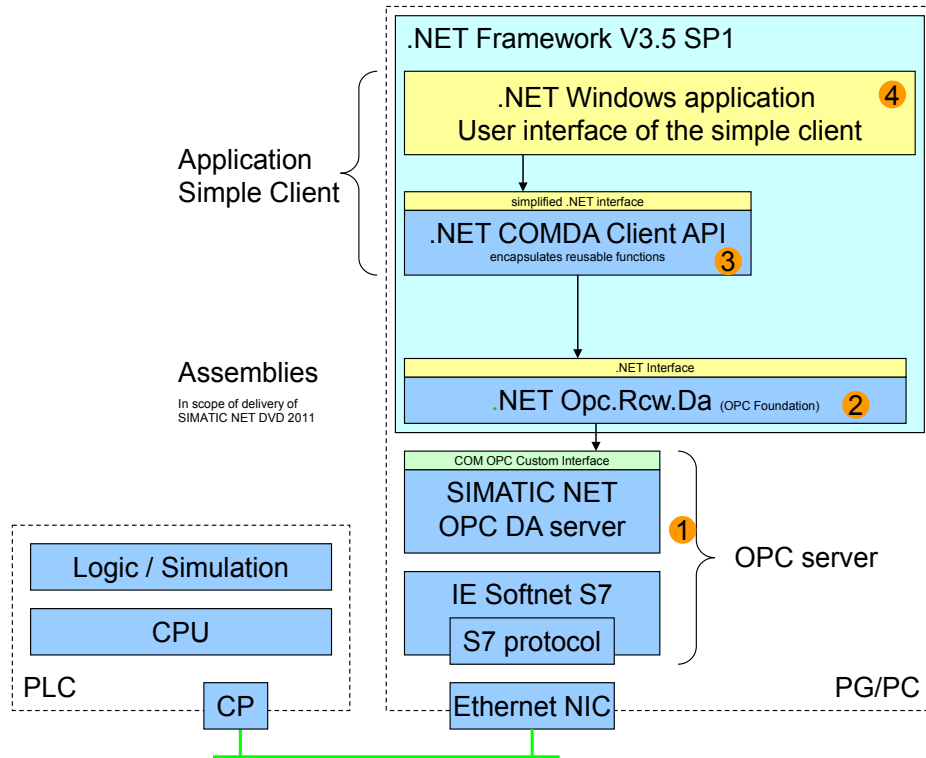


Table 4-1

No.	Module	
1.	OPC DA Server (SimaticNET)	The SIMATIC NET OPC server implements the necessary server logic for groups and items and the data connection to the S7 stations.
2.	OpcRcw.Da	Interface to access the COM components. Encapsulates the OPC interfaces for access to the .NET application.
3.	COMDA Client API	Reusable, simplified and tailored to this .NET Client API task. It offers reusable C# handling classes for finding and connecting to servers and for monitoring values.
4.	Simple Client	Simple user interface for the use of the client API with the functions: connect, disconnect, read, write and data monitoring.

Program overview

The figure below shows the function blocks in the Simple Client and the interaction with the OPC COM DA Server.

Figure 4-2

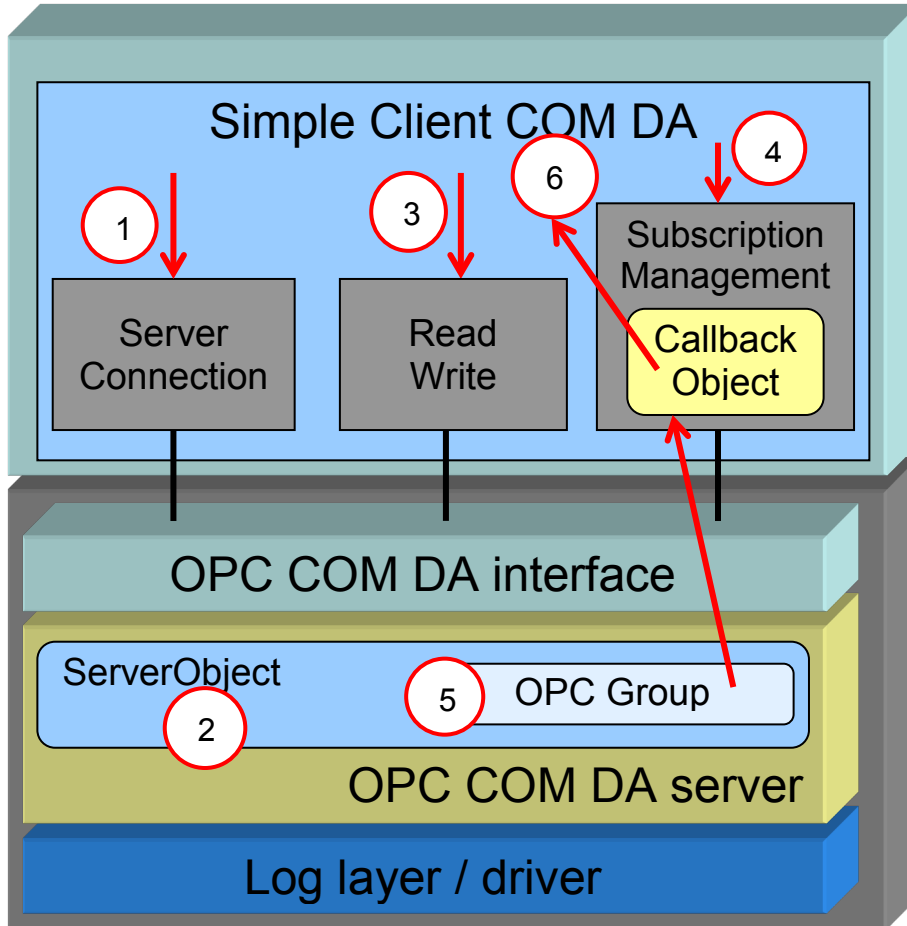


Table 4-2

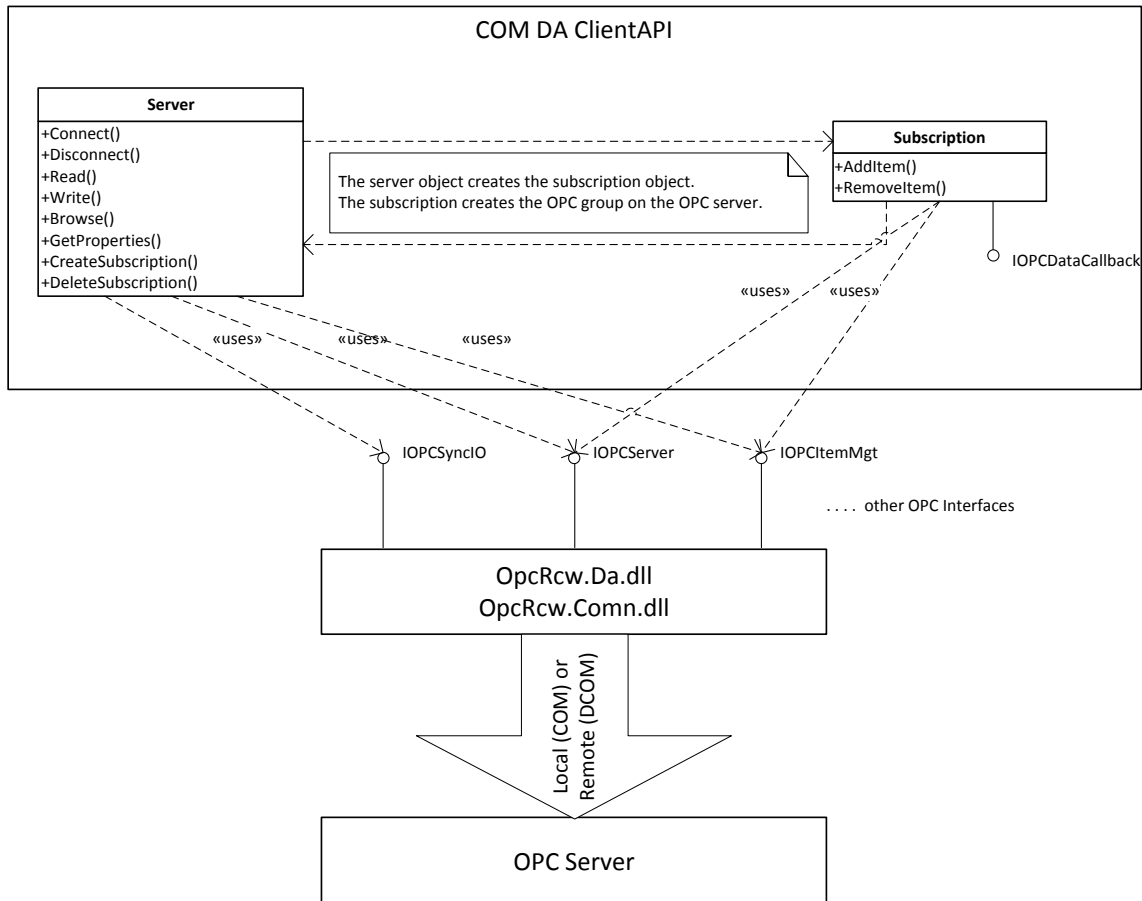
No.	Description
1	When establishing the connection the server object is created in the local or remote OPC server.
2	The server object is created via the CoCreateInstance COM mechanisms.
3	Once the connection has been established, the Client API creates an inactive group that is used for reading and writing of items.
4	When registering tags to monitor value changes, a subscription object is created on the client side. From a logic point of view, this corresponds to an OPC group.
5	An OPC group object is created in the server via the COM DA interface.
6	The received DataChanges are passed on by the ClientAPI to a callback method in the client application. The client application then updates the respective elements in the interface.

4.1 COMDA Client API

The class diagram in Figure 4-3 shows the classes of the COM DA ClientAPI. These classes encapsulate the accesses to the OPC server in a simple and reusable .NET API.

The classes are summarized in the .NET Assembly Siemens.Opc.Da.ClientAPI .dll. It has dependencies to the .NET RCW files of the OPC Foundation Opc.Rcw.Comn.dll and Opc.Rcw.Da.dll.

Figure 4-3



© Siemens AG 2014 All rights reserved

Server class

The **Server** wrapper class described in the table below encapsulates the functionality for the access to the OPC server. Moreover, it simplifies the use of those OPC interfaces which are need by the client application, with the exception of interfaces for the adding and deleting of items in the subscription.

The class is implemented in the OpcDaServer.cs file in the COMDAClientApi project.

Table 4-3

Method	Functionality
Connect	Creates an instance of the IOPCServer COM object. Afterwards a group is created on the server. This group is used for reading and writing of items.
Disconnect	Releases all references to the COM object and deletes all groups.
Browse	Offers a uniform simplified interface for browsing. With this method it can be browsed on DA2.05 and DA30 servers.
Read	Provides the current value of a tag on the server.
Write	Writes the value of a tag on the server.
CreateSubscription	Creates a subscription. The subscription is the container to monitor item values. An OPCGroup is created in the direction of the OPC server. This function is transferred a callback function. This callback function is called as soon as the values for the items of this subscription have changed.
DeleteSubscription	Removes an existing subscription and deletes the group on the OPC server.

Subscription class

The **Subscription** wrapper class described in the table below, encapsulates the use of an OPCGroup for the value exchange between server and client.

The class is implemented in the OpcDaSubscription.cs file in the COMDAClientAPI project.

Table 4-4

Method	Functionality
AddItem	Creates an item in the respective OPC group to monitor value changes and link them with the subscription.
RemoveItem	Removes an item from the subscription.

4.2 Simple Client COM DA

The simple client provides a simple example for the use of the Client API. The most important function such as connect, disconnect, read, write and monitoring of data is displayed in a file or class with a dialog. The code for the example can be found in the SimpleClientCOMDA project in the SimpleClient.cs file.

User interface of the simple example

The user interface is operated via buttons for the individual functions.

Figure 4-4

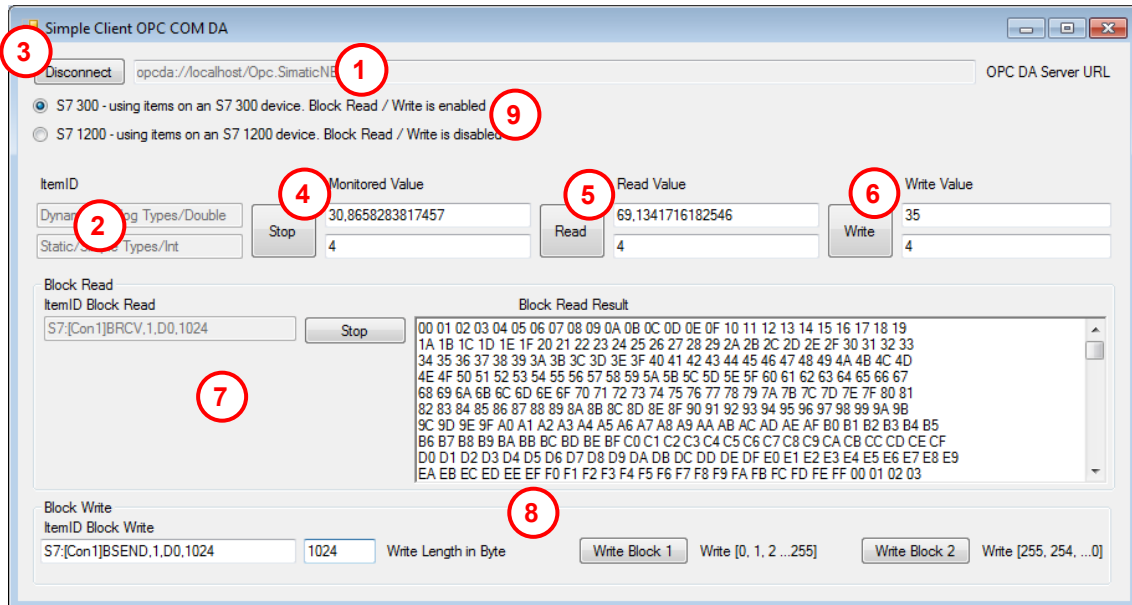


Table 4-5

No.	Description
1.	The server URL can be specified in the text box for the Server URL. The URL is made up of the computer name and the server ProgID: opcda://<computername>/ProgID. For example, for a local SIMATIC NET server://localhost/Opc.SimaticNET.
2.	In the text boxes for the ItemIDs the OPC ItemID is specified. This ID is unique in the address space of the server.
3.	Via the connect and disconnect buttons, the connection to the OPC server can be established or disconnected.
4.	Via the monitor button, the subscription is created and the two items are added to a group. From now on data changes are reported from the server to the client and displayed in the two text boxes next to the button. Errors are each shown instead of values. The button changes the text to "Stop" after creating the subscription. Via this button the subscription can then also be deleted again.
5.	The "Read" button reads the values of the two tags with the indicated ItemIDs and displays them in the text boxes next to the button. Reading in the direction of the server is performed synchronous via the IOPCSyncIO::Read() OPC method.
6.	The "Write" button writes the value from the text box next to the button on the tag which was identified by the ItemID. For writing, the text is sent from the text field to the server as a value. The server automatically converts the values to the suitable data type of the item. Writing in the direction of the server is performed synchronous via the IOPCSyncIO::Write() OPC method.

4 Functional Mechanisms of this Application

No.	Description
7.	In the "Block Read" group, data can be received which is actively sent by the S7 with the BSEND block service. This can be, for example, used for the sending of result data from the S7 to a PC application.
8.	In the "Block Write" group, data blocks can be sent to the S7 which are there received by the BRCV block service. Two blocks with different contents can be sent. This can be used, for example, for the download of recipe data for the S7.
9.	The block services are only available for the S7-300. When switching to S7-1200, the reading and writing of blocks on the interface will be disabled.

Functions of the simple example

The functions can be found in the SimpleClientDA class in the SimpleClient.cs file. Simple error handling is implemented in the functions. If an exception occurs when calling OPC, a dialog with the error message will appear. If an error occurs for one or several tags with the tag related calls, the error is displayed in the respective text boxes.

Table 4-6

Function	Description
btnConnect_Click	In this function the connection to the OPC server is established via the Server::Connect() function of the client API. The URL string from the corresponding text box is transferred.
btnDisconnect_Click	In this function the connection to the OPC server is disconnected via the Server::Disconnect() function of the client API.
btnRead_Click	In this function the values for the two items are read via the Server::Read() function. The result is written into the respective text box. If there is an error when reading or if the quality is bad (i.e. no value can be delivered by the server), an error code is written into the text box.
btnMonitor_Click	First of all a subscription is created with the Server::CreateSubscription(). Afterwards the two items are created with Subscription::AddItem(). The ClientHandles are permanently set in this example, since only 3 items are used. The ClientHandle is transferred from the server to the client if there is a message of value changes. Thus, the client can uniquely assign the items. If a subscription has already been created it is deleted by Server::DeleteSubscription() and the client does no longer receive value changes.
OnDataChange	The function is indicated as callback function at Server::CreateSubscription(). In the function it is first of all checked whether the call arrives in the main thread of the dialog. If this is not the case, the call is transferred to the main thread of the dialog via BeginInvoke. Otherwise access to the dialog is not possible. Afterwards the ClientHandles are checked and based on the handles, the respective text field is updated. If the value is a normal tag, the value is simply output as text. However, if it is the value of a block tag, the byte array is extracted and displayed as a sequence of HEX values for the individual elements of the byte arrays.
btnWrite_Click	The function calls the Server::Write() and writes the current text as value to the item.

4 Functional Mechanisms of this Application

Function	Description
btnMonitorBlock_Click	<p>First of all a subscription is created with the <code>Server::CreateSubscription()</code>. Afterwards the "BlockWrite" item is created with <code>Subscription::AddItem()</code>.</p> <p>If a subscription has already been created, it is deleted by <code>Server::DeleteSubscription()</code> and the client no longer receives value changes for the "BlockWrite" item.</p>
btnWriteBlock1_Click	<p>The function writes generated values to the "BlockWrite" item. The simulation creates a byte array with the length indicated in the user interface and fills the values with a tag value which starts at 0 and is incremented after each assignment.</p>
btnWriteBlock2_Click	<p>The function is identical to <code>btnWriteBlock1_Click</code>; however, the tag value assigned here starts with 255 and is decremented after each assignment.</p>

Note Further details are contained in the source code of this application as comment.

4.3 S7 program

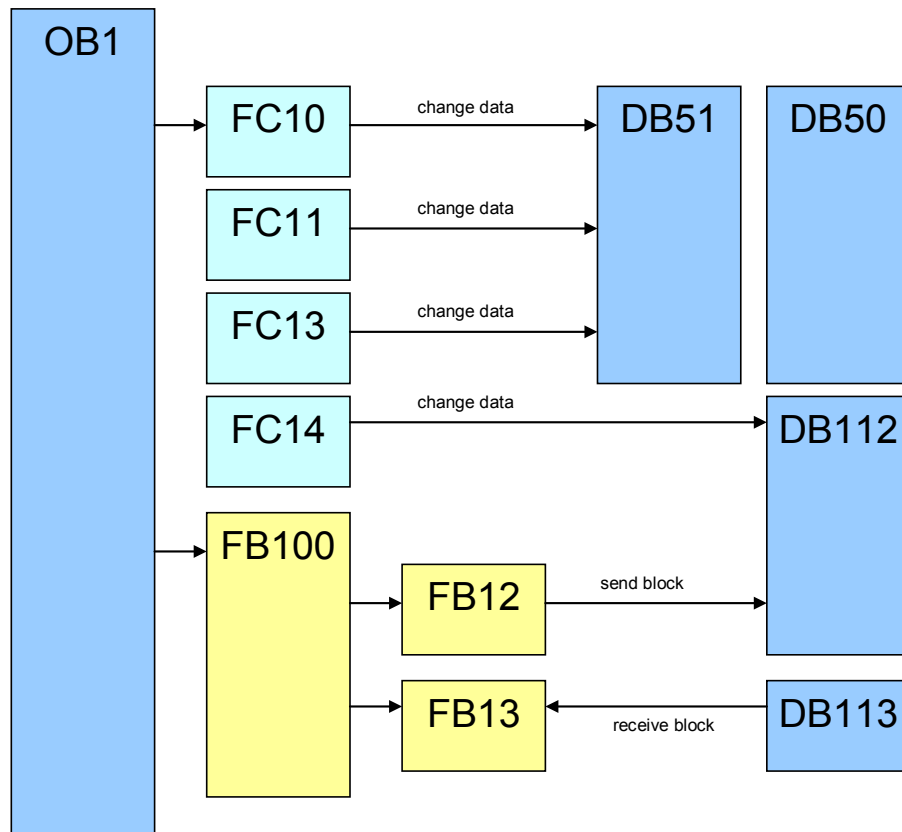
Overview

The S7 program is essentially divided in two parts. First of all, the dynamic data for the tag services is simulated, afterwards the send data is simulated and the block services are called in FB 100.

Note

The block services are only available in the S7-300; the S7-1200 solely communicates via S7 tag services.

Figure 4-5



© Siemens AG 2014 All rights reserved

Simulation of dynamic data

The table below gives a brief overview of program parts and their function for data simulation. Details were deliberately excluded here.

4 Functional Mechanisms of this Application

Table 4-7

Block	Remarks
OB1	Cyclic Main initially, a variable timer is set here, whose interval is used to call the other program functions. The rate of change of data can be set via DB10 byte 0.
FC10	ChangeDateAndTime increments a date value as well as the time in DB51.
FC11	ChangeSimpleTypes increments the data of DB51. 8bit types are incremented with +1, 16bit types with +100 and 32bit types with +1000.
FC13	ChangeString increments a string of length 10 in DB51.
DB10	SimulationConfiguration contains global tags for the configuration of data simulation.
DB50	StaticDataTypes contains simple data types which were given symbolic names. The values are pre-initialized with maximum end value range.
DB51	DynamicDataTypes contains simple data types which were given symbolic names. The values are incremented with the functions FC10 to FC12 according to their value ranges.
SFC21	FILL auxiliary function to fill data areas with values, storage initialization.

Block-oriented data

The S7 communication for the S7-300 is realized via FBs (loadable function blocks) and not via SFBs (integrated system function blocks) as is the case for the S7-400. However, if you call a SFB instead of a FB in the S7 program, the block delivers an "ERROR" and displays "STATUS = 27". This status indicates that the function block for the S7 communication is not present on the S7-300. The communication FBs for the S7-300 are located in the "SIMATIC_NET_CP > CP300 > blocks" library.

The table below gives a brief overview of the program parts and their function regarding BSEND/BRCV. Details were deliberately excluded here.

Table 4-8

Block	Remarks
OB1	Cyclic Main Call of the send block (SFB12) for BSEND and of the receive block (SFB13) for BRCV via function block 100. For S7-300, FB12 and FB13 from the library are used since S7 300 communicates via loadable FBs and not via system functions.
FC14	ChangeSendData Increments a byte which will then be copied to the entire send buffer
FB100 + DB100 (Instance DB)	InvokeBSENDandBRCV Calls the real communication blocks and supplies its parameters.
DB112	SendData Data block with 4096 bytes length which will be transferred to the send block.
DB113	ReceiveData data block with 4096 bytes length which will be transferred to the receive block.
SFB12 + DB12 (Instance DB)	BSEND The send block transfers the send buffer to the communication processor (CP), which will then send it according to RID and connectionID to the communication partner.
FB12	BSEND (only S7-300)
SFB13 + DB13 (Instance DB)	BRCV The receive block picks up the data package last received from the communication processor (CP) according to RID and connectionID and files it in the receive buffer.
FB13	BRCV (only S7-300)
SFC21	FILL Auxiliary function to fill data areas with values, storage initialization.

The program logic sends or receives data blocks and supplies the respective parameters via the FB100. When larger data packages are sent, a multiple call of the BSEND or BRCV block is necessary. This multiple call is performed by FB100.

Note

If the send and receive block is called cyclically, a high communication load may be generated and there is furthermore the danger that the data buffers are overwritten before they are processed by the counterpart. This is why the program logic should contain a flow control to ensure data consistency. For this purpose the parameters DONE (ready) and NDR (new data received) are to be used.

To be able to exchange data between two S7-300 stations via a S7 connection configured in TIA portal, communication functions have to be called in the S7 program. The FB12 "BSEND" block is used for sending data and the FB13 "BRCV" block for receiving data.

Here, the S7 connection has to be bilaterally configured since the S7 communication via FB12 "BSEND" and FB13 "BRCV" is based on the client-client principle.

Note

When the S7 connection is configured via the integrated IE interface of the S7-300 controller of the CPU31x-2PN/DP or the CPU319-3PN/DP, the FB12 "BSEND" and FB13 "BRCV" from the "Standard Library -> Communication Blocks -> Blocks" library has to be used with the family="CPU_300". These FBs can be used for the S7 communication via the integrated IE interface of the CPU as well as for the S7 communication via the S7-300 IE-CPs.

5 Configuration and Setting of the OPC Server

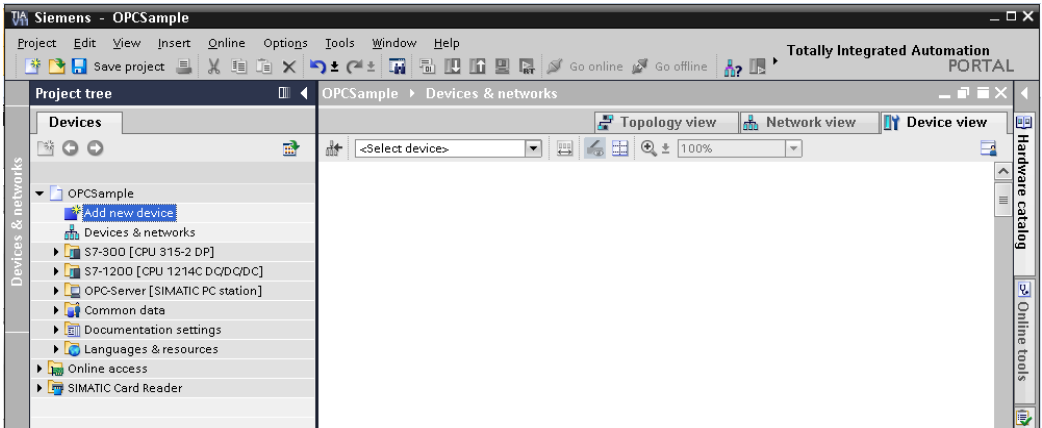
Introduction

In the sections below, we show you the steps to configure the PC station and the OPC server in STEP7 V11. You only need to read this chapter if you are interested in the details. The configuration has already been completed in the delivered STEP7 project.

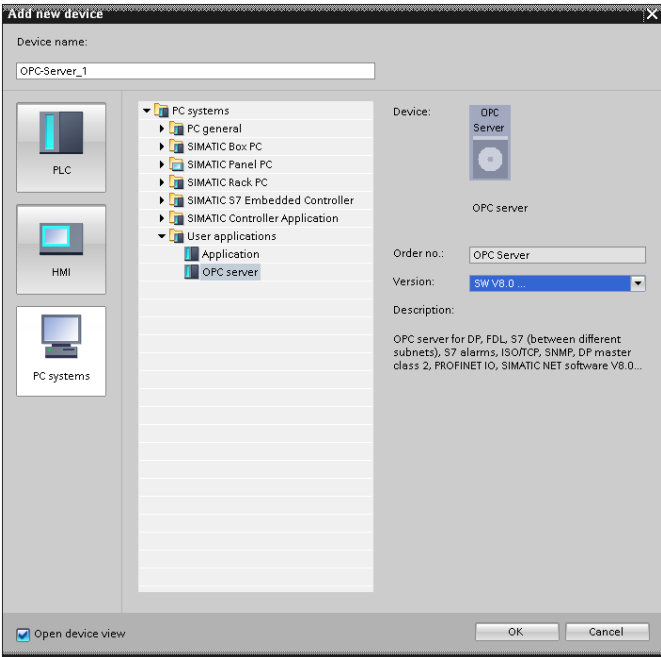
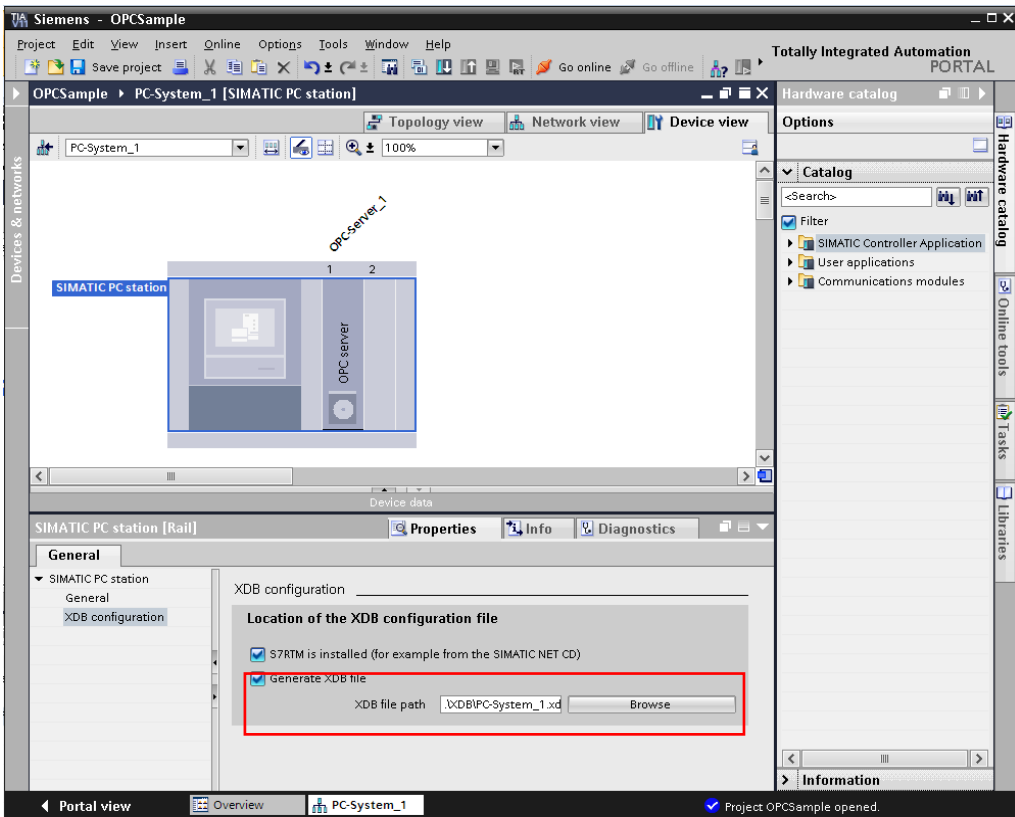
5.1 Configuration of the OPC server in STEP7 V11

Execute the following steps, for example, in order to add an OPC server to an existing STEP7 V11 project (or higher) with already configured SIMATIC station.

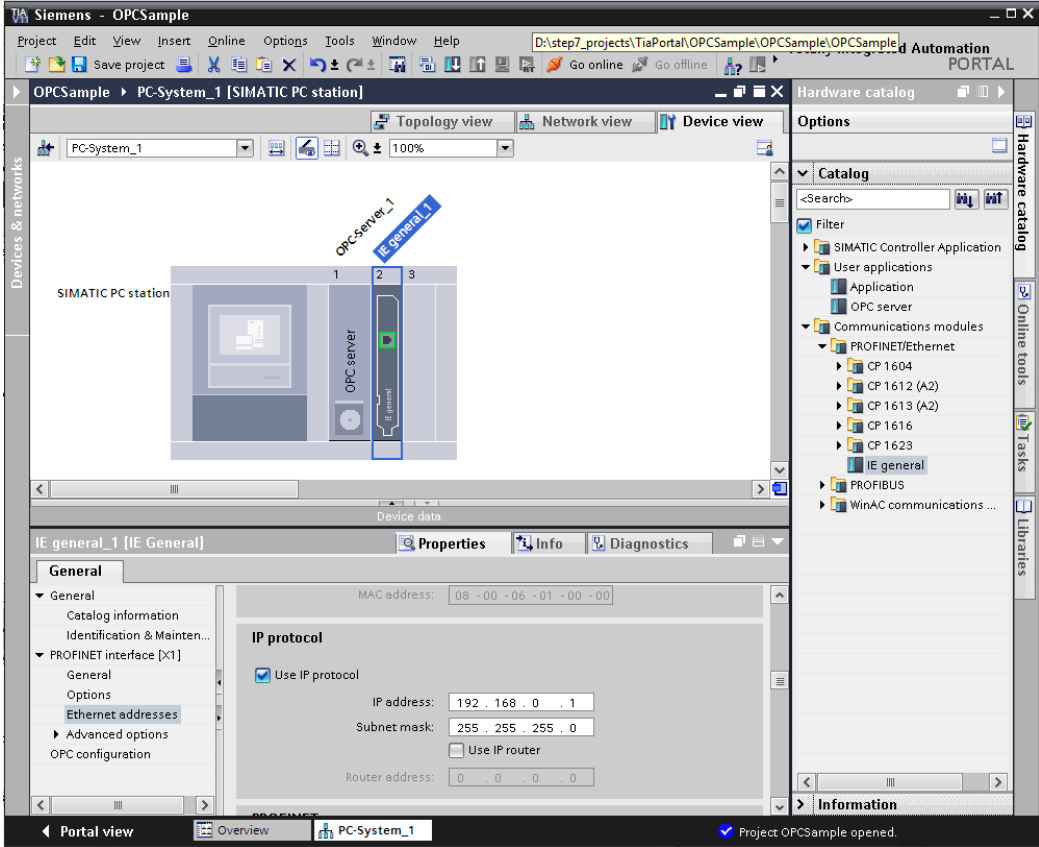
Table 5-1

No.	Action
1	Open your STEP7 V11 project and go to the project view.
2	Double click "Add new device" 

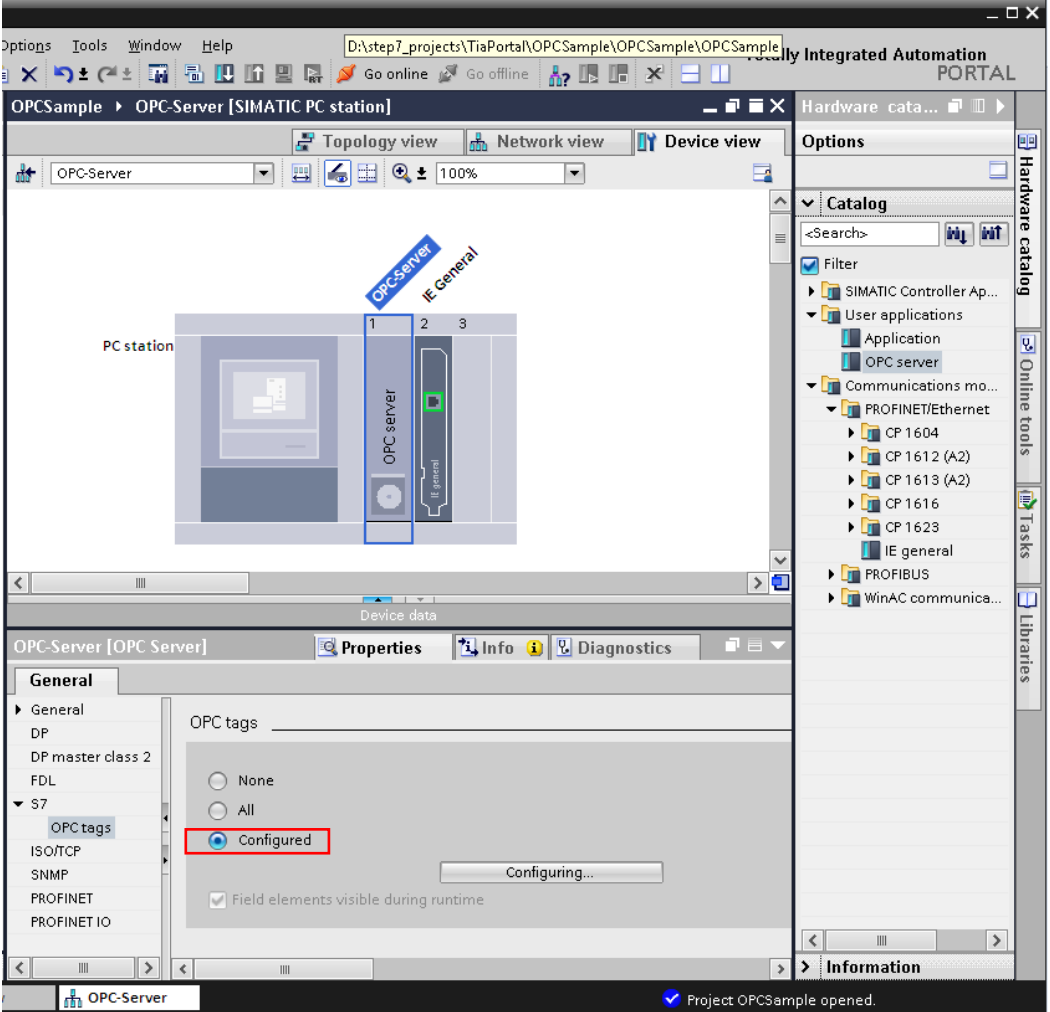
5 Configuration and Setting of the OPC Server

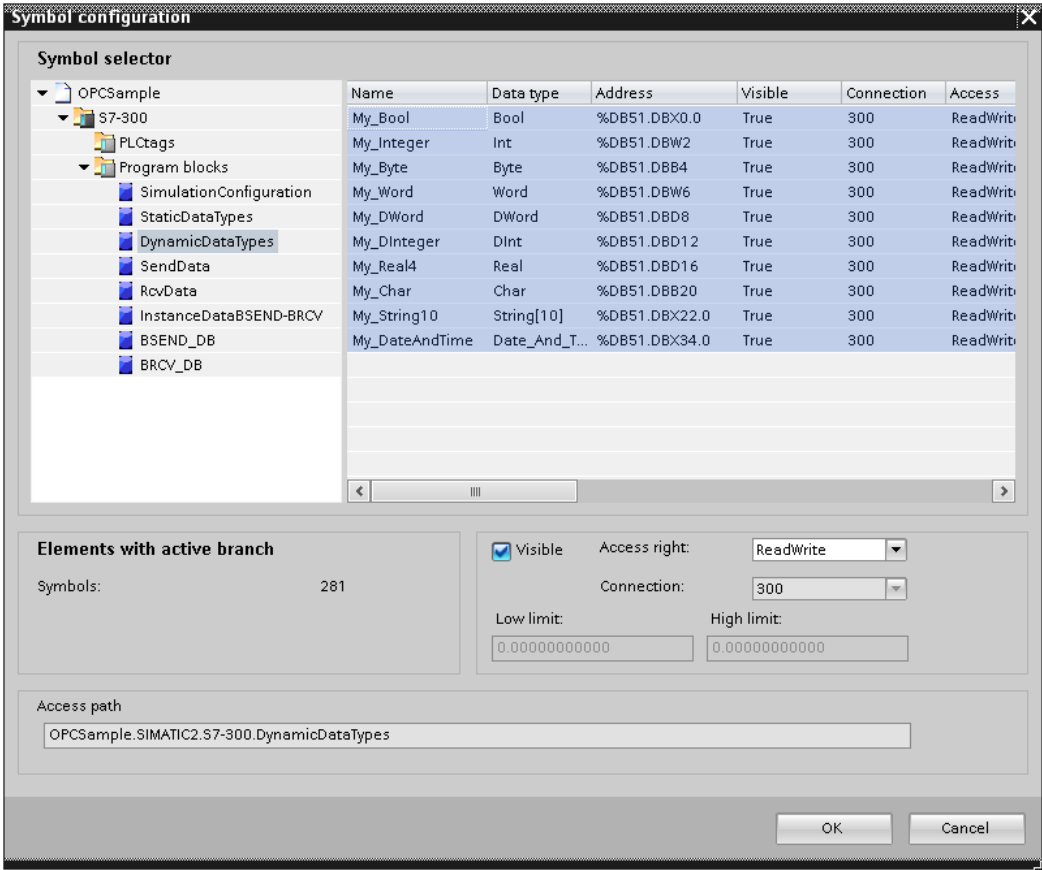
No.	Action
3	<p>Select "PC systems" and in the "User applications" the "OPC server".</p> <p>If possible, the version of the SIMATIC NET OPC server should correspond to the really installed version.</p> <p>In this application the SIMATIC NET OPC Server V8.0 (or higher) was used.</p> 
4	<p>A SIMATIC PC station is created where a SIMATIC NET OPC server is configured on index 1.</p> <p>Enable the creation of the XDB file (Generate XDB file). The storage of the XDB file by the system can be set under "XDB file path".</p> 

5 Configuration and Setting of the OPC Server

No.	Action
5	<p>Add an "IE general" network card and assign the IP address. Note: The IP address has to match the IP address of the target system.</p> 

5 Configuration and Setting of the OPC Server

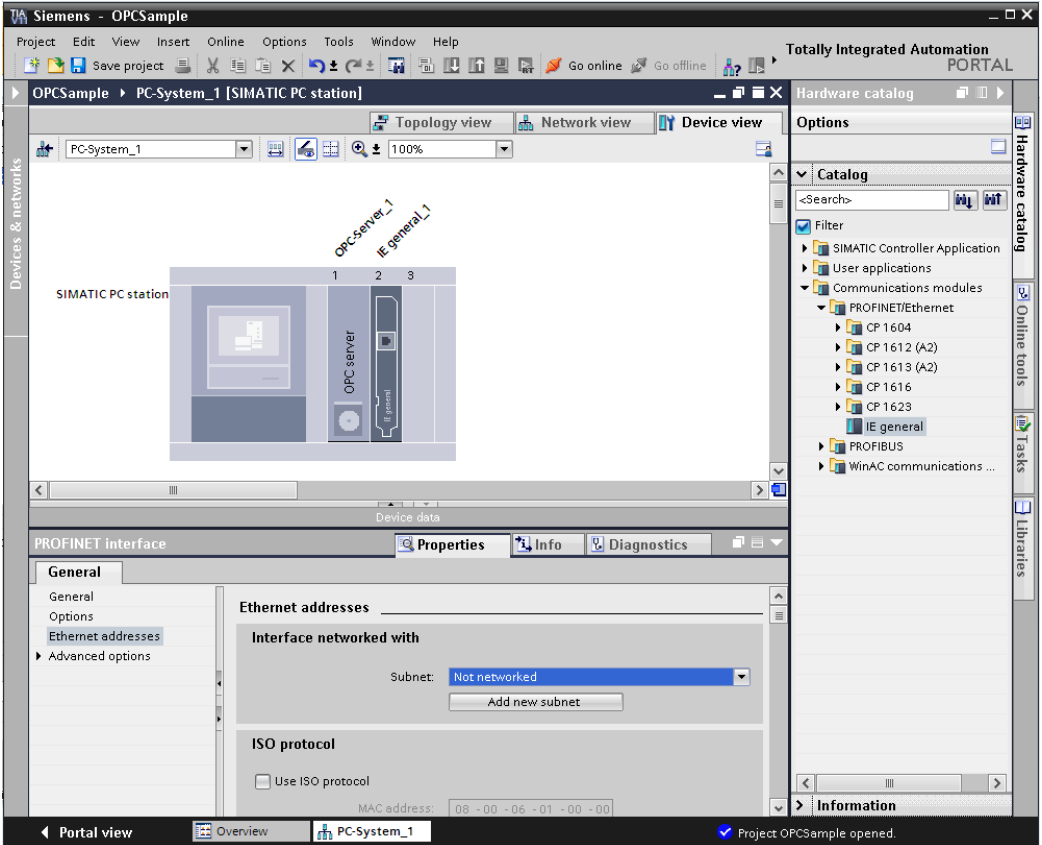
No.	Action
6	<p>Enable the symbolic display of the OPC tags. When using symbols, you can configure whether you want to use "All" or "Configured" symbols. Often you do not want to make all symbols available via the OPC server (e.g. internal tags or contents of instance data blocks).</p>
	

No.	Action
7	<p>If you only want to access a symbolic selection of process tags, select the “Configured” option. Select the symbolically addressable tags via the “Configuring...” button.</p>  <p>Note: If no symbols are displayed, the S7 connection configuration is missing. Only when a S7 connection was configured for the respective SIMATIC by the OPC server, will symbols be displayed. This will be shown in the next chapter.</p>

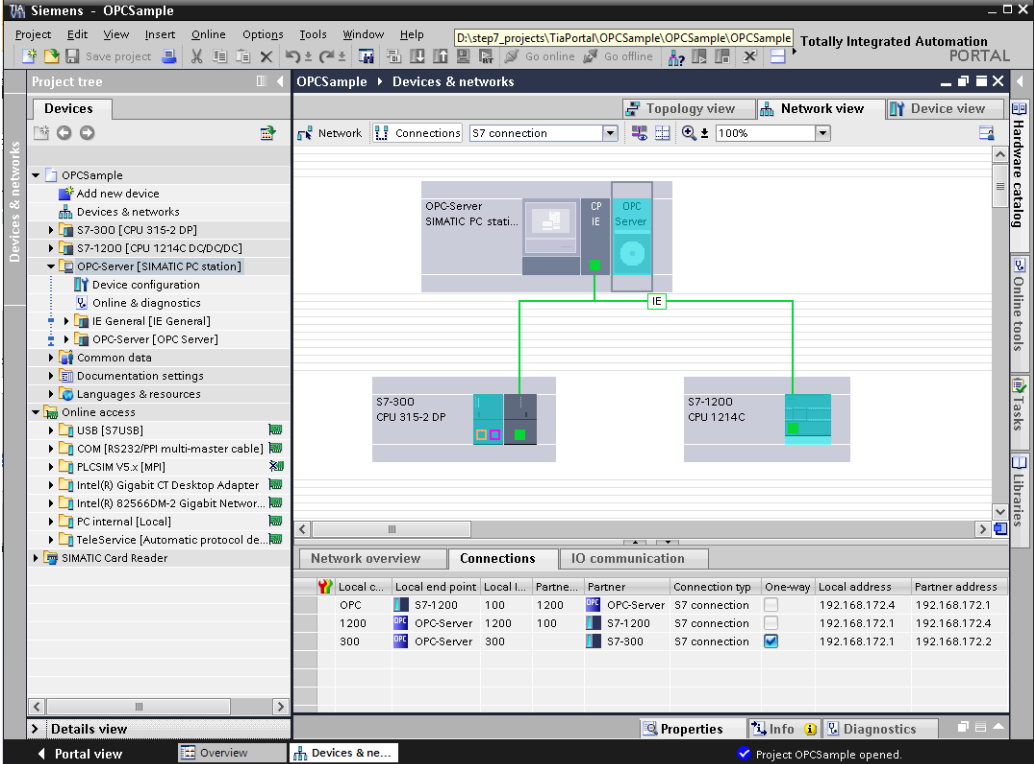
5.2 Configuring the S7 connections

The PC station and the OPC server require configured S7 connections in order to be able to use the symbolic display of the OPC items.

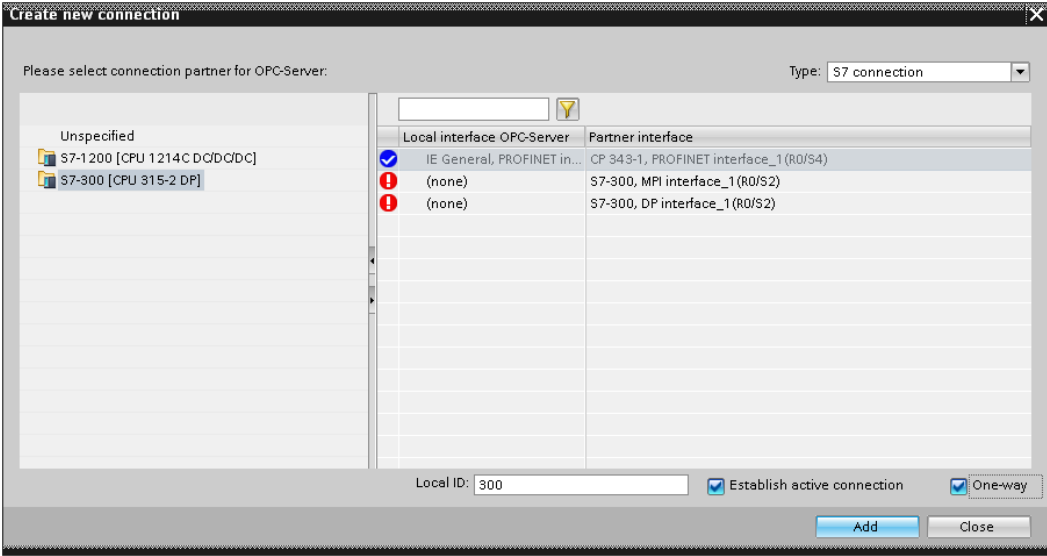
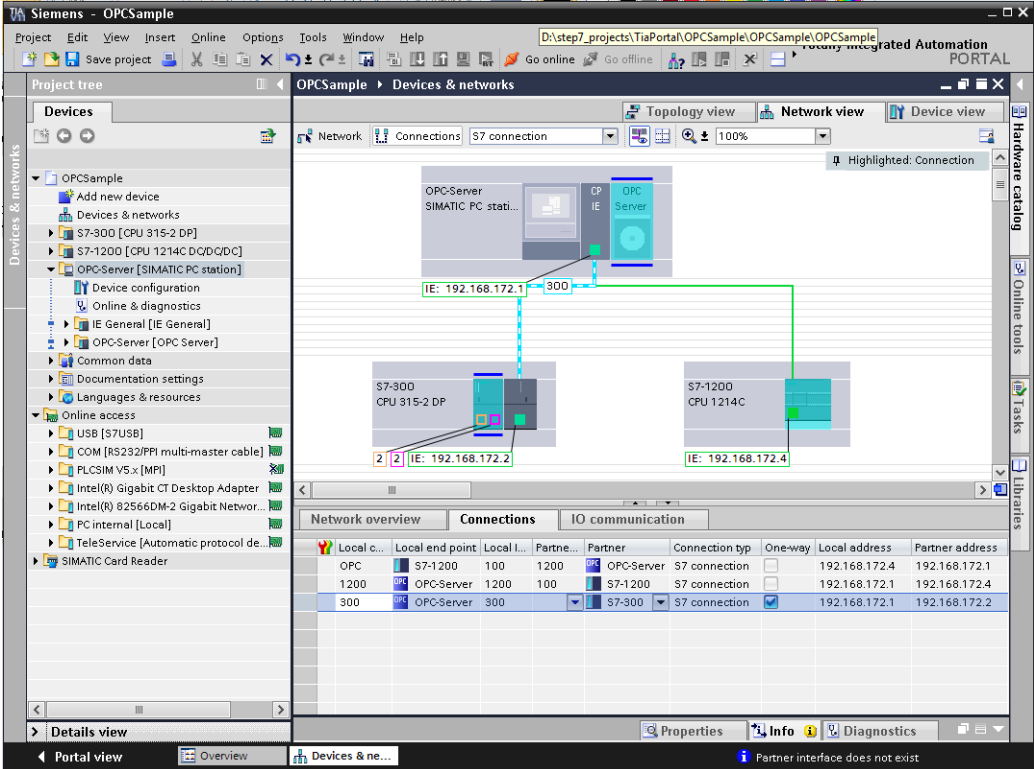
Table 5-2

No.	Action
1	Open the object properties of the network card. If no IE network has been configured yet, generate a new one with “Add new subnet” and apply the standard parameters.
	
Note:	<p>All network nodes should be in the same IE network.</p> <p>Go to the network view, network the PC station with the SIMATIC station/s and configure a S7 connection each.</p>

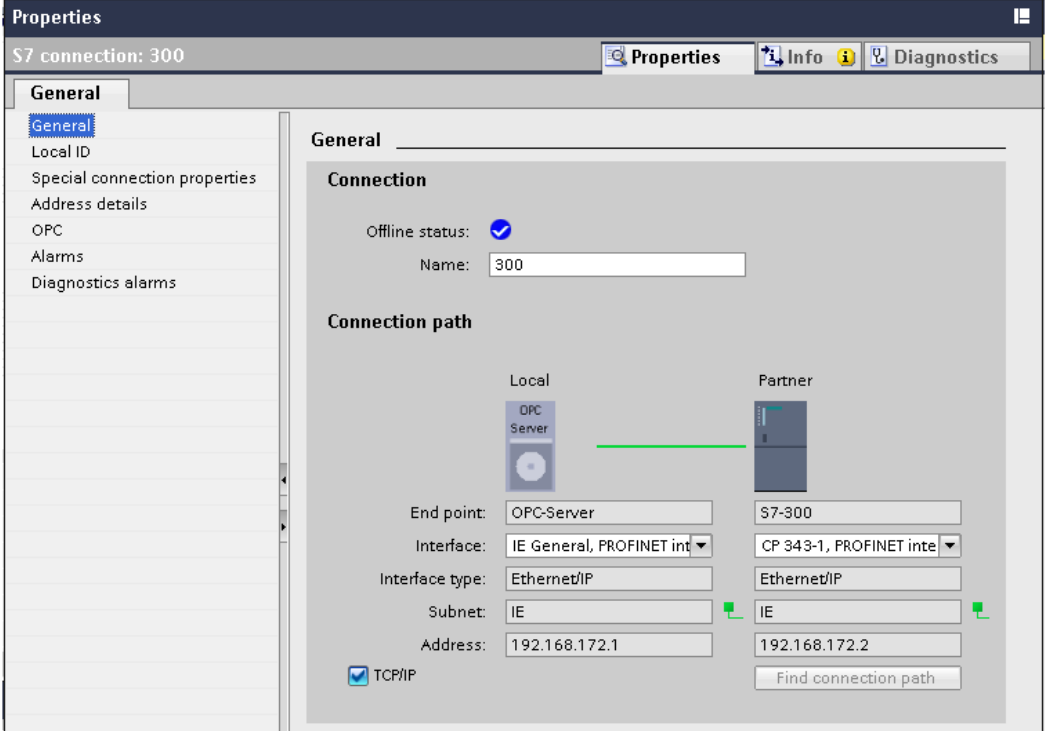
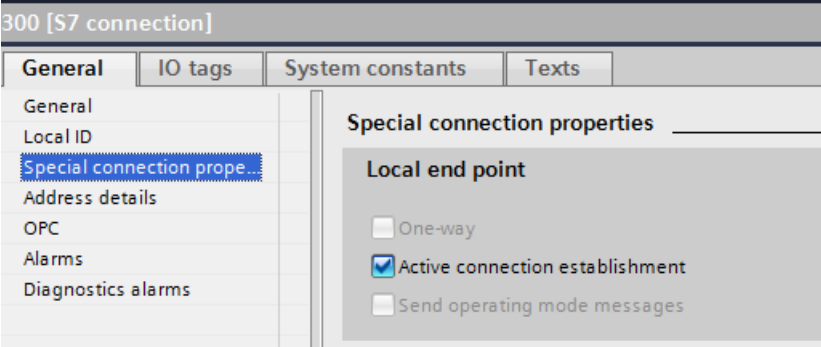
5 Configuration and Setting of the OPC Server

No.	Action																																												
2	<p>Open the network view. Connect all nodes with the same IE network.</p>  <p>The screenshot displays the Siemens TIA Portal interface. The 'Network view' is active, showing a network topology with three main components: an OPC-Server SIMATIC PC station, an S7-300 CPU 315-2 DP, and an S7-1200 CPU 1214C. All three are connected to a common IE network. Below the network diagram, a table provides details on connections and IO communication.</p> <table border="1" data-bbox="598 853 1310 1021"> <thead> <tr> <th colspan="2">Network overview</th> <th colspan="2">Connections</th> <th colspan="4">IO communication</th> </tr> <tr> <th>Local c...</th> <th>Local end point</th> <th>Local I...</th> <th>Partne...</th> <th>Partner</th> <th>Connection typ</th> <th>One-way</th> <th>Local address</th> <th>Partner address</th> </tr> </thead> <tbody> <tr> <td>OPC</td> <td>S7-1200</td> <td>100</td> <td>1200</td> <td>OPC-Server</td> <td>S7 connection</td> <td><input type="checkbox"/></td> <td>192.168.172.4</td> <td>192.168.172.1</td> </tr> <tr> <td>1200</td> <td>OPC-Server</td> <td>1200</td> <td>100</td> <td>S7-1200</td> <td>S7 connection</td> <td><input type="checkbox"/></td> <td>192.168.172.1</td> <td>192.168.172.4</td> </tr> <tr> <td>300</td> <td>OPC-Server</td> <td>300</td> <td></td> <td>S7-300</td> <td>S7 connection</td> <td><input checked="" type="checkbox"/></td> <td>192.168.172.1</td> <td>192.168.172.2</td> </tr> </tbody> </table>	Network overview		Connections		IO communication				Local c...	Local end point	Local I...	Partne...	Partner	Connection typ	One-way	Local address	Partner address	OPC	S7-1200	100	1200	OPC-Server	S7 connection	<input type="checkbox"/>	192.168.172.4	192.168.172.1	1200	OPC-Server	1200	100	S7-1200	S7 connection	<input type="checkbox"/>	192.168.172.1	192.168.172.4	300	OPC-Server	300		S7-300	S7 connection	<input checked="" type="checkbox"/>	192.168.172.1	192.168.172.2
Network overview		Connections		IO communication																																									
Local c...	Local end point	Local I...	Partne...	Partner	Connection typ	One-way	Local address	Partner address																																					
OPC	S7-1200	100	1200	OPC-Server	S7 connection	<input type="checkbox"/>	192.168.172.4	192.168.172.1																																					
1200	OPC-Server	1200	100	S7-1200	S7 connection	<input type="checkbox"/>	192.168.172.1	192.168.172.4																																					
300	OPC-Server	300		S7-300	S7 connection	<input checked="" type="checkbox"/>	192.168.172.1	192.168.172.2																																					
3	<p>Select the OPC server. "Add new connection" in the context menu.</p>																																												

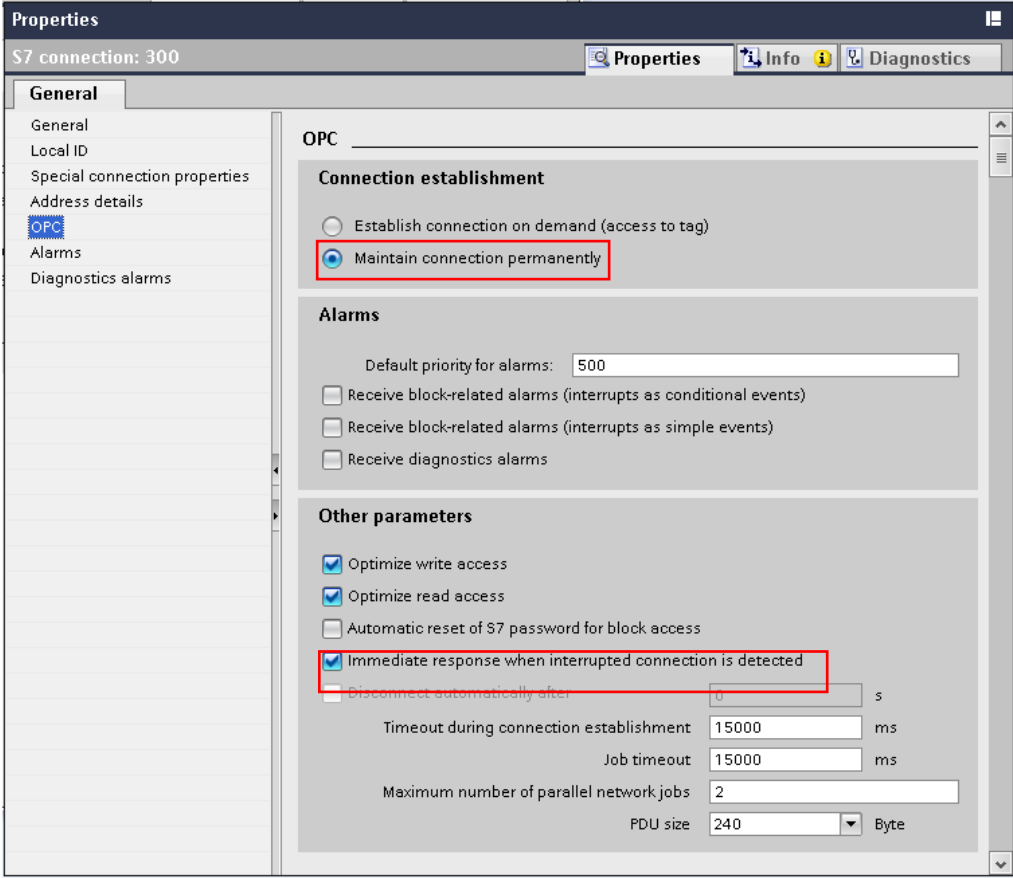
5 Configuration and Setting of the OPC Server

No.	Action																																				
4	<p>Select the type of connection (S7 connection) and the connection partner as well as the physical access point (blue) that is supported by both partners.</p> <p>Assign a name for the S7 connection.</p> 																																				
5	<p>The connection now appears in the connection list.</p>  <table border="1" data-bbox="592 1480 1310 1592"> <thead> <tr> <th>Local c...</th> <th>Local end point</th> <th>Local I...</th> <th>Partne...</th> <th>Partner</th> <th>Connection type</th> <th>One-way</th> <th>Local address</th> <th>Partner address</th> </tr> </thead> <tbody> <tr> <td>OPC</td> <td>S7-1200</td> <td>100</td> <td>1200</td> <td>OPC-Server</td> <td>S7 connection</td> <td><input type="checkbox"/></td> <td>192.168.172.4</td> <td>192.168.172.1</td> </tr> <tr> <td>1200</td> <td>OPC-Server</td> <td>1200</td> <td>100</td> <td>S7-1200</td> <td>S7 connection</td> <td><input type="checkbox"/></td> <td>192.168.172.1</td> <td>192.168.172.4</td> </tr> <tr> <td>300</td> <td>OPC-Server</td> <td>300</td> <td></td> <td>S7-300</td> <td>S7 connection</td> <td><input checked="" type="checkbox"/></td> <td>192.168.172.1</td> <td>192.168.172.2</td> </tr> </tbody> </table>	Local c...	Local end point	Local I...	Partne...	Partner	Connection type	One-way	Local address	Partner address	OPC	S7-1200	100	1200	OPC-Server	S7 connection	<input type="checkbox"/>	192.168.172.4	192.168.172.1	1200	OPC-Server	1200	100	S7-1200	S7 connection	<input type="checkbox"/>	192.168.172.1	192.168.172.4	300	OPC-Server	300		S7-300	S7 connection	<input checked="" type="checkbox"/>	192.168.172.1	192.168.172.2
Local c...	Local end point	Local I...	Partne...	Partner	Connection type	One-way	Local address	Partner address																													
OPC	S7-1200	100	1200	OPC-Server	S7 connection	<input type="checkbox"/>	192.168.172.4	192.168.172.1																													
1200	OPC-Server	1200	100	S7-1200	S7 connection	<input type="checkbox"/>	192.168.172.1	192.168.172.4																													
300	OPC-Server	300		S7-300	S7 connection	<input checked="" type="checkbox"/>	192.168.172.1	192.168.172.2																													

5 Configuration and Setting of the OPC Server

No.	Action
6	<p>Configure the connection by editing the properties of the connection.</p> 
7	<p>The connection with CPU 315-2 PN/DP was configured both ways here.</p> 

5 Configuration and Setting of the OPC Server

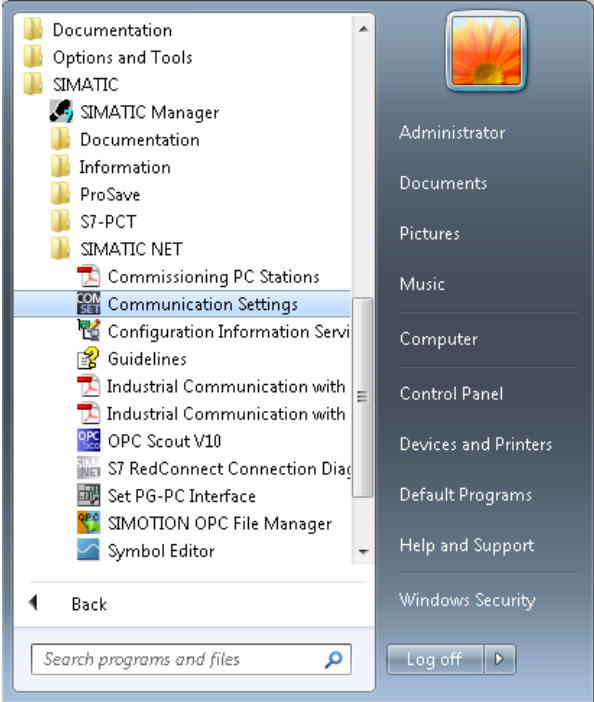
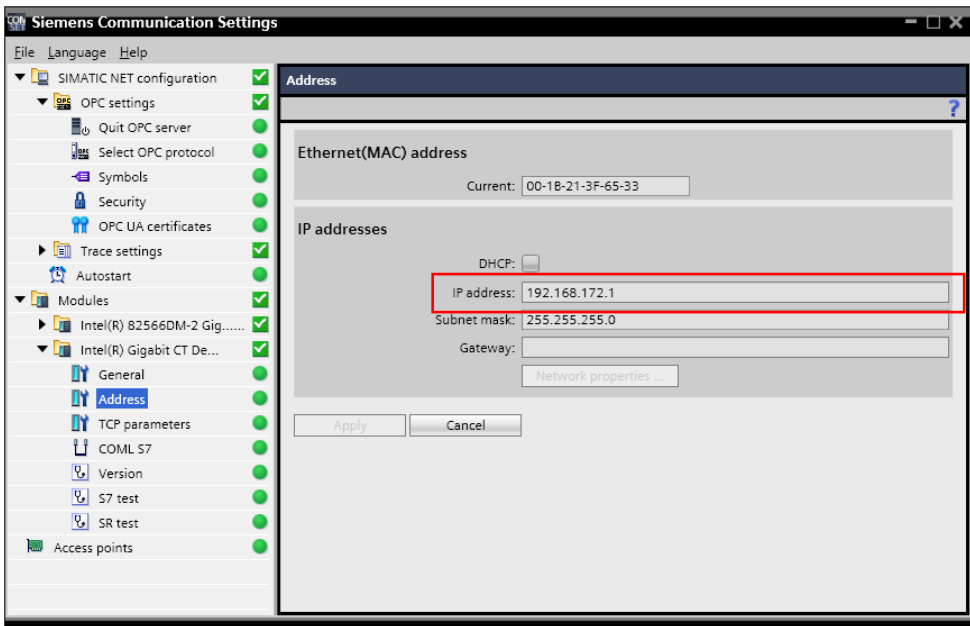
No.	Action
9	<p>The OPC server should always establish the connection actively and maintain it permanently. The OPC server should furthermore respond instantly if a connection failure is detected.</p> 
10	Perform the same steps in the same way for the S7 connection for S7-1200.
11	Save your project.

5.3 Check the settings

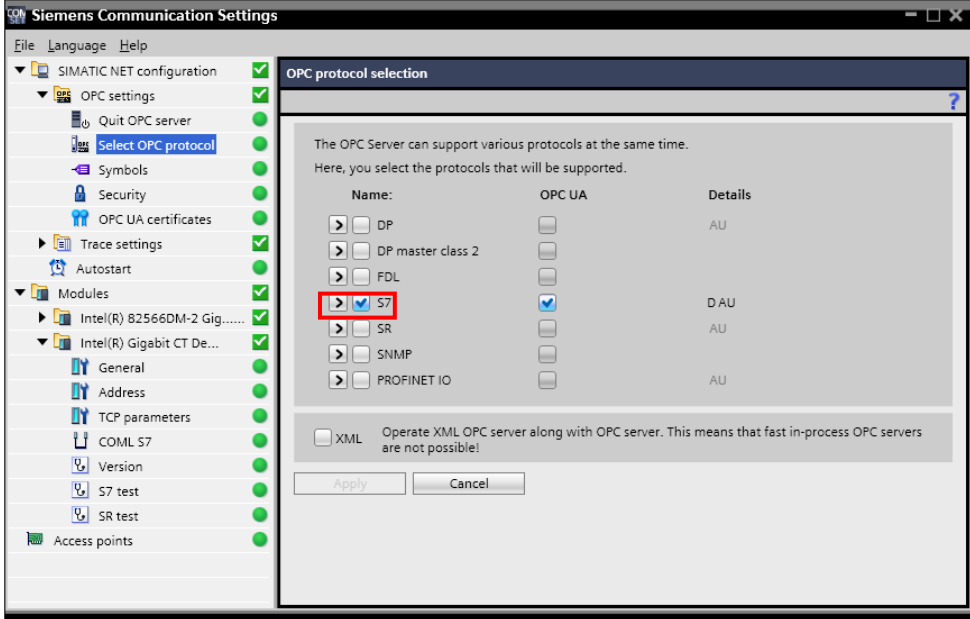
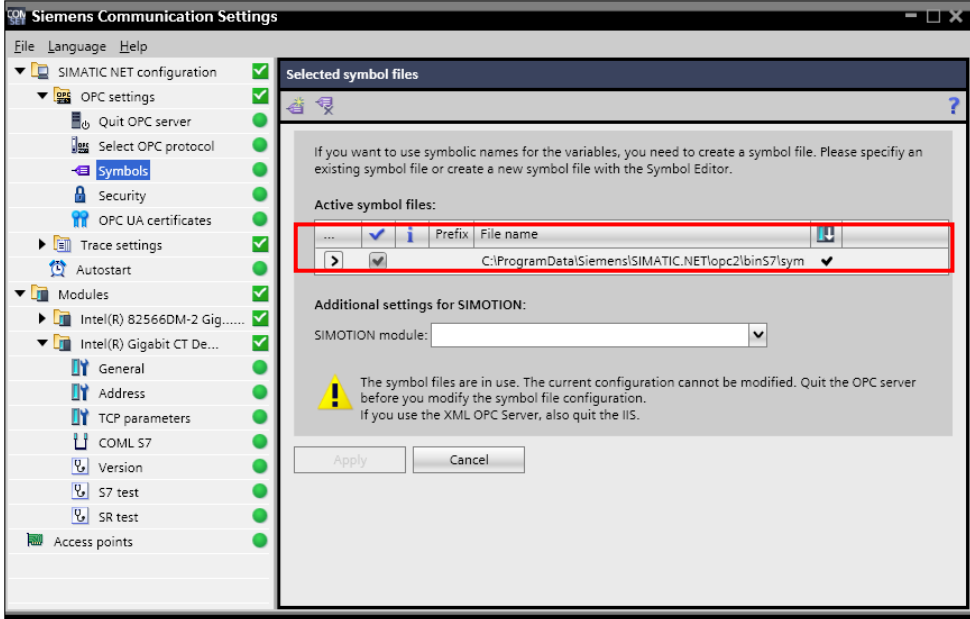
The settings can be checked with the Communication Settings configuration console.

The following settings should be considered:

Table 5-3

No.	Action
1	<p>Open the configuration dialog</p>  <p>The screenshot shows a Windows Start menu search interface. The search bar contains the text 'Communication Settings'. The results list various folders and applications, with 'Communication Settings' highlighted in blue. Other items include 'SIMATIC Manager', 'Documentation', 'Information', 'ProSave', 'S7-PCT', 'SIMATIC NET', 'Commissioning PC Stations', 'Configuration Information Servi', 'Guidelines', 'Industrial Communication with', 'Industrial Communication with', 'OPC Scout V10', 'S7 RedConnect Connection Diag', 'Set PG-PC Interface', 'SIMOTION OPC File Manager', and 'Symbol Editor'. A 'Log off' button is visible at the bottom right of the search results.</p>
2	<p>Check the set IP address.</p>  <p>The screenshot shows the 'Siemens Communication Settings' dialog box. The 'Address' tab is selected. Under 'Ethernet(MAC) address', the current address is '00-1B-21-3F-65-33'. Under 'IP addresses', the 'DHCP' checkbox is unchecked. The 'IP address' field is highlighted with a red box and contains the value '192.168.172.1'. The 'Subnet mask' is '255.255.255.0' and the 'Gateway' field is empty. There are 'Apply' and 'Cancel' buttons at the bottom.</p>

5 Configuration and Setting of the OPC Server

No.	Action																								
4	<p>Check the set protocols Note: Enabling the S7 protocol is sufficient for this application.</p>  <p>The screenshot shows the 'Siemens Communication Settings' dialog box with the 'OPC protocol selection' tab active. The left sidebar shows a tree view with 'OPC settings' expanded. The main area contains a table of protocols:</p> <table border="1"> <thead> <tr> <th>Name:</th> <th>OPC UA</th> <th>Details</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/> DP</td> <td><input type="checkbox"/></td> <td>AU</td> </tr> <tr> <td><input type="checkbox"/> DP master class 2</td> <td><input type="checkbox"/></td> <td></td> </tr> <tr> <td><input type="checkbox"/> FDL</td> <td><input type="checkbox"/></td> <td></td> </tr> <tr> <td><input checked="" type="checkbox"/> S7</td> <td><input checked="" type="checkbox"/></td> <td>DAU</td> </tr> <tr> <td><input type="checkbox"/> SR</td> <td><input type="checkbox"/></td> <td>AU</td> </tr> <tr> <td><input type="checkbox"/> SNMP</td> <td><input type="checkbox"/></td> <td></td> </tr> <tr> <td><input type="checkbox"/> PROFINET IO</td> <td><input type="checkbox"/></td> <td>AU</td> </tr> </tbody> </table> <p>Below the table, there is a checkbox for 'XML Operate XML OPC server along with OPC server. This means that fast in-process OPC servers are not possible!' and 'Apply' and 'Cancel' buttons.</p>	Name:	OPC UA	Details	<input type="checkbox"/> DP	<input type="checkbox"/>	AU	<input type="checkbox"/> DP master class 2	<input type="checkbox"/>		<input type="checkbox"/> FDL	<input type="checkbox"/>		<input checked="" type="checkbox"/> S7	<input checked="" type="checkbox"/>	DAU	<input type="checkbox"/> SR	<input type="checkbox"/>	AU	<input type="checkbox"/> SNMP	<input type="checkbox"/>		<input type="checkbox"/> PROFINET IO	<input type="checkbox"/>	AU
Name:	OPC UA	Details																							
<input type="checkbox"/> DP	<input type="checkbox"/>	AU																							
<input type="checkbox"/> DP master class 2	<input type="checkbox"/>																								
<input type="checkbox"/> FDL	<input type="checkbox"/>																								
<input checked="" type="checkbox"/> S7	<input checked="" type="checkbox"/>	DAU																							
<input type="checkbox"/> SR	<input type="checkbox"/>	AU																							
<input type="checkbox"/> SNMP	<input type="checkbox"/>																								
<input type="checkbox"/> PROFINET IO	<input type="checkbox"/>	AU																							
5	<p>Check whether the symbols have been loaded.</p>  <p>The screenshot shows the 'Siemens Communication Settings' dialog box with the 'Selected symbol files' tab active. The left sidebar shows 'Symbols' selected. The main area contains a table of active symbol files:</p> <table border="1"> <thead> <tr> <th>Prefix</th> <th>File name</th> </tr> </thead> <tbody> <tr> <td><input checked="" type="checkbox"/></td> <td>C:\ProgramData\Siemens\SIMATIC.NET\opc2\bin\S7sym</td> </tr> </tbody> </table> <p>Below the table, there is a dropdown for 'SIMOTION module:' and a warning message: 'The symbol files are in use. The current configuration cannot be modified. Quit the OPC server before you modify the symbol file configuration. If you use the XML OPC Server, also quit the IIS.' and 'Apply' and 'Cancel' buttons.</p>	Prefix	File name	<input checked="" type="checkbox"/>	C:\ProgramData\Siemens\SIMATIC.NET\opc2\bin\S7sym																				
Prefix	File name																								
<input checked="" type="checkbox"/>	C:\ProgramData\Siemens\SIMATIC.NET\opc2\bin\S7sym																								
6	<p>If one of the settings does not correspond with the displayed images, perform the preceding configurations again.</p> <p>Close the configuration dialog box.</p>																								

6 Installation and Commissioning

6.1 Hardware and software installation

In this chapter we describe which hardware and software component you have to install in order to operate the example with the existing configuration data. Also note the manuals as well as delivery information that are delivered with the respective products.

Installing the hardware

For details on the hardware components, please refer to chapter 2.3. For setting up the hardware, please proceed according to the following table:

NOTICE	Only switch on the voltage supply after the last step.
---------------	--

Table 6-1

No.	Focus	Action
1	Control S7-300 station	Install the station in compliance with the diagram shown in chapter 2.
2	Control S7-1200 station	Install the station in compliance with the diagram shown in chapter 2.
3	PG/PC station	Install the station in compliance with the diagram shown in chapter 2.
4	Industrial Ethernet	Connect the controller with the PG as shown in the illustration in chap. 2.

Note Instead of using a hub or a switch you can also use a cross cable for a direct connection.

Always observe the installation guidelines for SIMATIC S7.

Installation of the standard software

STEP7 V11 and SIMATIC NET have to be installed on the PG/PC. STEP7 V11 is already preinstalled on the current SIMATIC PGs.

A description of the installation procedure for STEP7 and SIMATIC NET is not part of this documentation. The installation takes place in the usual Windows environment and is self-explanatory or described in the respective manuals.

Address overview of the involved modules

If you want to operate the project on an existing Industrial Ethernet, you have to note the following address specification:

Table 6-2

Focus	Module	IP address
PG/PC	NDIS network card	192.168.172.1
Control	CP 343-1	192.168.172.2
Control	CPU 1214 C	192.168.172.4

Notes

- Note the correct subnet mask 255.255.255.0.
- Alternatively, it is also possible to change the assigned IP addresses in the STEP7 project.

Setting the IP address

The Ethernet network card has to be switched to the configured operation. For this purpose the PC station has to be configured.

Note

Please ensure, that the network card has the fixed IP address 192.168.172.1 (it can be set via the network settings and the TCP/IP properties) if you want to use the project included in delivery.

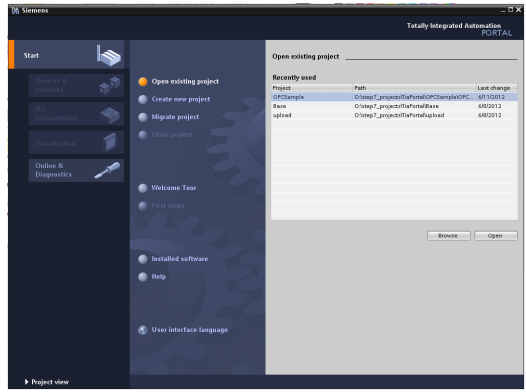
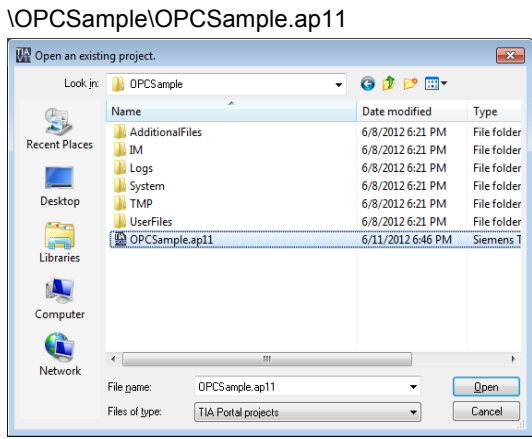
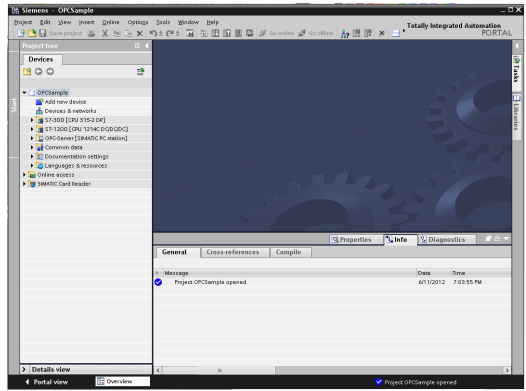
6.2 Loading the PC station via STEP 7 V1x

Installation of the STEP7 project via TIA portal

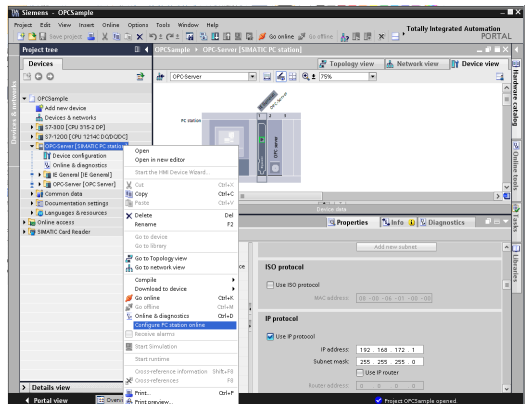
The PC station can be loaded directly from the TIA portal. Alternatively, a PC station can also be configured via the station configuration editor and the XDB file (see chap. 2).

The description in **Fehler! Verweisquelle konnte nicht gefunden werden.** was created with TIA V11. The same procedure applies for TIA V13.

Table 6-3

No.	Action	Remarks
1	Extract the TIA project: STEP7_TIA11.zip	Unzip the project in a path in which you have the read and write permissions.
2	Open the TIA portal and navigate to the project via the browser function	
3	Confirm by opening.	
4	Go to the project view once you opened	

6 Installation and Commissioning

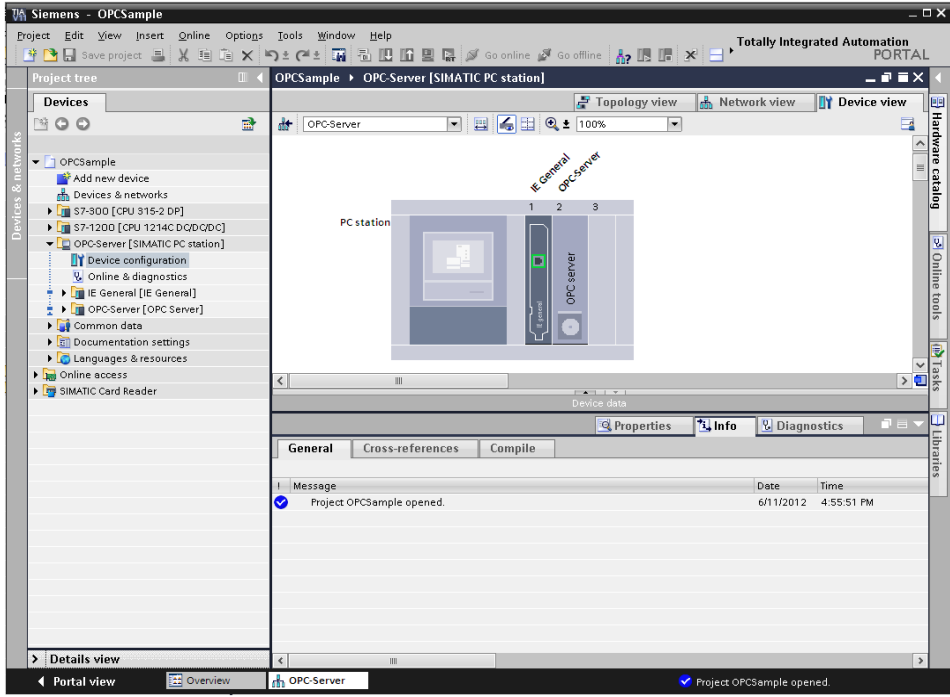
No.	Action	Remarks
5	<p>Load the PC station</p> <p>Alternative: You can also configure the PC station via the import of the included XDB file (see 6.3).</p>	

Changing the IP address of the PC station in STEP7 V1x

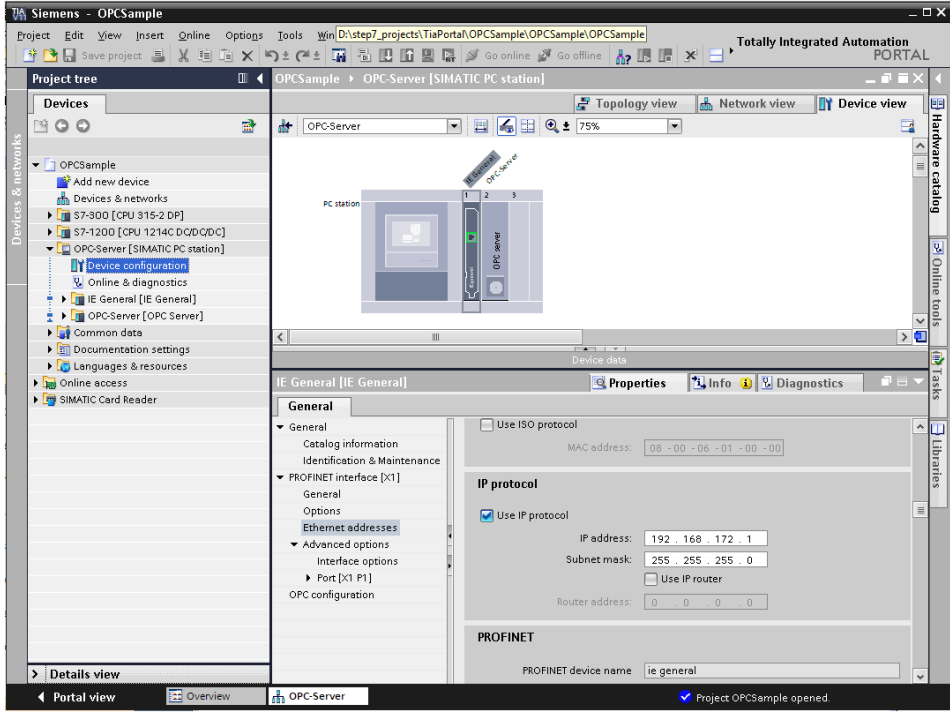
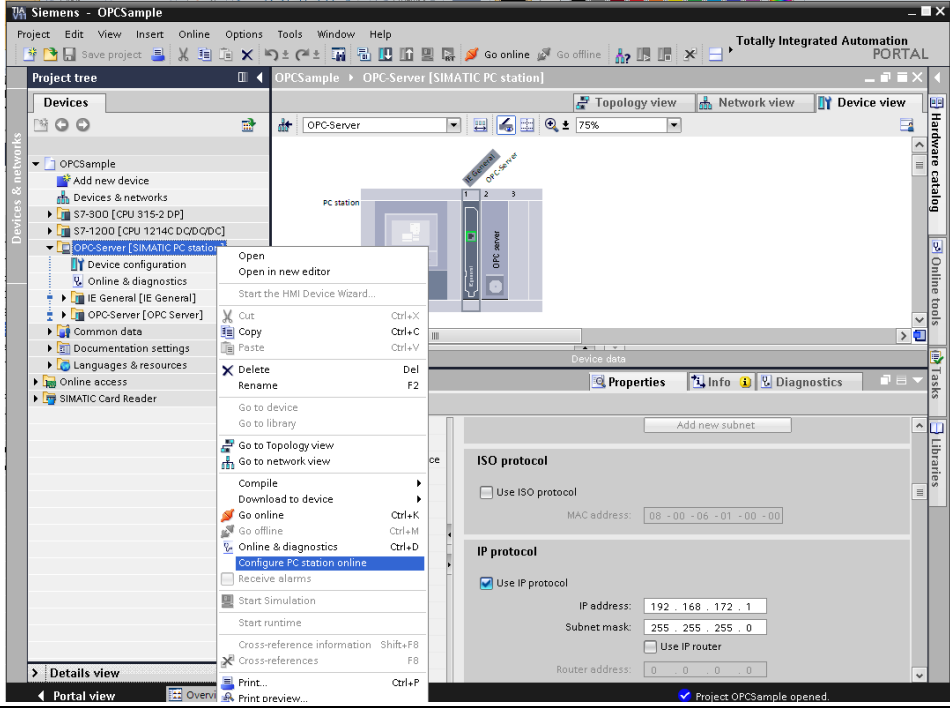
Note Only execute these steps if you want to change the IP address of your PC station.

Table 6-4

No.	Action/Remark
1.	Start the TIA Portal V1x
2.	Go to the project view, select the SIMATIC PC station in the project view and open the "Device view".



6 Installation and Commissioning

No.	Action/Remark
3.	<p>Select your network card in the PC station and select the “Properties” tab in the bottom window. You can change the IP address under “Ethernet addresses” -> “IP protocol”.</p> <p>Note: In this case an IE General network card was used.</p> 
4.	<p>In the context menu of the SIMATIC PC station you can find “Configure PC station online”. Changes on the configuration of the PC station require the station to be reloaded.</p> 
5.	<p>After these configuration steps the station has to be reloaded.</p>

6.3 Importing the XDB file into the Station Configuration Editor

Introduction

Alternatively to loading a PC station via the TIA portal (see chap Fehler! Verweisquelle konnte nicht gefunden werden.), it can also be configured via the station configuration editor and the XDB file. The XDB file already exists in the TIA project included.

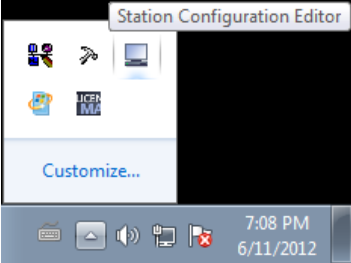
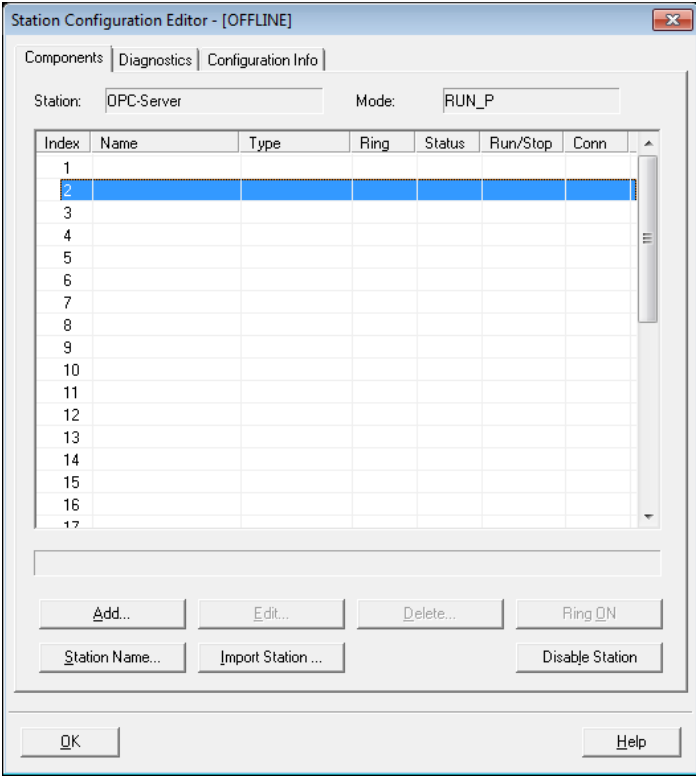
Setting the IP address

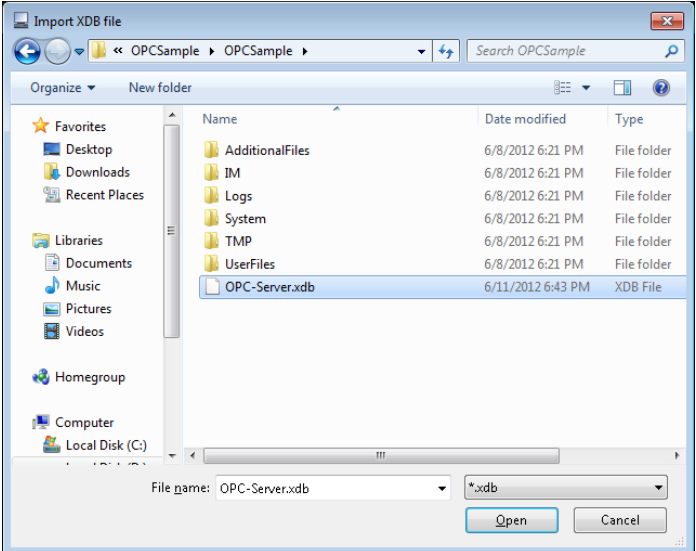
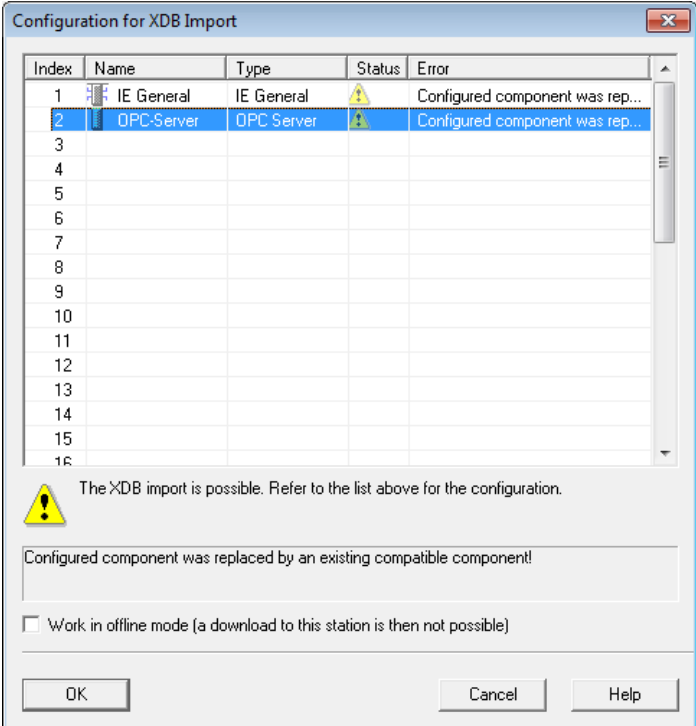
The Ethernet network card has to be switched to the configured operation. For this purpose the PC station has to be configured.

Note

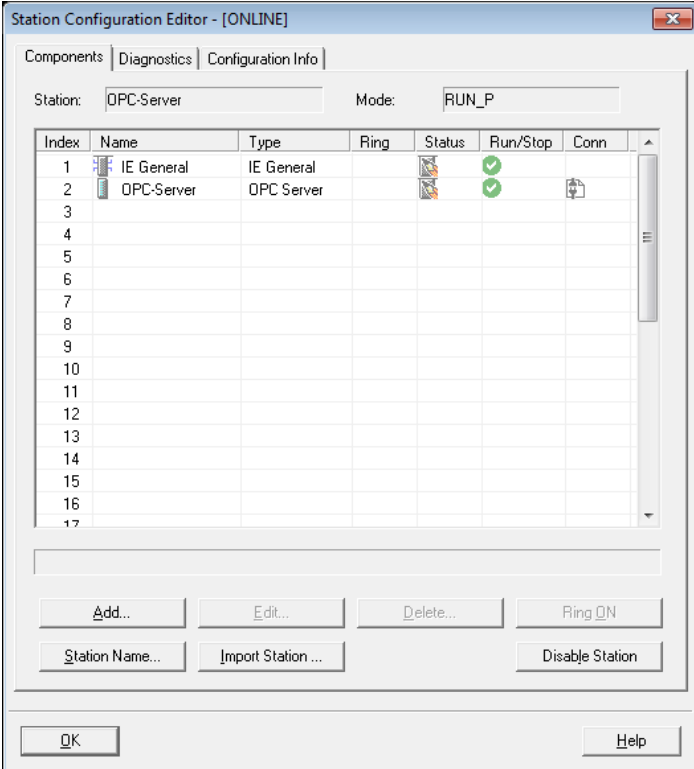
Please ensure, that the network card has the fixed IP address 192.168.172.1 (it can be set via the network settings and the TCP/IP properties) if you want to use the project included in delivery.

Table 6-5

No.	Action	Remarks
1	<p>Open the Station Configuration Editor by double-clicking</p>  <p>(icon in the task bar, next to the time)</p>	
2	Click on the "Import Station" button.	Confirm the task with "Yes".

No.	Action	Remarks
3	<p>Navigate to the project folder of your STEP7 V11 project. Select the XDB file. Click the dialog.</p>	
4	<p>The import wizard confirms that import is possible. Confirm with OK. Note: If components have been configured in a different version, they will be exchanged by existing compatible versions</p>	

6 Installation and Commissioning

No.	Action	Remarks																																																																																																																														
5	After the successful import of the XDB file, your PC station is ONLINE.	 <p>The screenshot shows the 'Station Configuration Editor - [ONLINE]' window. It has three tabs: 'Components', 'Diagnostics', and 'Configuration Info'. The 'Configuration Info' tab is active. The window displays the following information:</p> <ul style="list-style-type: none"> Station: OPC-Server Mode: RUN_P <table border="1" data-bbox="703 398 1337 801"> <thead> <tr> <th>Index</th> <th>Name</th> <th>Type</th> <th>Ring</th> <th>Status</th> <th>Run/Stop</th> <th>Conn</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>IE General</td> <td>IE General</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>2</td> <td>OPC-Server</td> <td>OPC Server</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>3</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>4</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>5</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>6</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>7</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>8</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>9</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>10</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>11</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>12</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>13</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>14</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>15</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>16</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>17</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table> <p>Below the table, there are several buttons: 'Add...', 'Edit...', 'Delete...', 'Ring QN', 'Station Name...', 'Import Station ...', and 'Disable Station'. At the bottom of the window are 'OK' and 'Help' buttons.</p>	Index	Name	Type	Ring	Status	Run/Stop	Conn	1	IE General	IE General					2	OPC-Server	OPC Server					3							4							5							6							7							8							9							10							11							12							13							14							15							16							17						
Index	Name	Type	Ring	Status	Run/Stop	Conn																																																																																																																										
1	IE General	IE General																																																																																																																														
2	OPC-Server	OPC Server																																																																																																																														
3																																																																																																																																
4																																																																																																																																
5																																																																																																																																
6																																																																																																																																
7																																																																																																																																
8																																																																																																																																
9																																																																																																																																
10																																																																																																																																
11																																																																																																																																
12																																																																																																																																
13																																																																																																																																
14																																																																																																																																
15																																																																																																																																
16																																																																																																																																
17																																																																																																																																

6.4 Installation of the OPC client on the PC/PG

The application software is delivered with a setup program.

NOTICE	<p>If you are using an older operating system then Windows 7 SP1 and if SimaticNET PC Software V8.x was not installed on it, you have to install .NET-Framework 3.5 +SP1 first.</p> <p>Information on this matter can be found on the Microsoft Internet pages (see \5).</p>
---------------	---

For installing the operator user interface proceed as follows:

Table 6-6

No.	Action	Remarks
1.	Extract the 21043779_OPCCClient_RCW_CODE.zip file	This zip file contains the STEP7 V11 and the STEP7 V13, project as well as the OPC client with C# Source code.
2.	Unzip the Csharp_OPCCClient_RCW_CODE.zip file	
3.	You find the DAClient.exe file in the \OpcClientDA_V2\bin directory	EXE can only be executed if the respective assemblies are located in the same directory

Files included

The archive file contains the MS Visual Studio Solution file and the source code as well as pre-compiled binary files for x86 systems. In the subfolder is the executable file (EXE) as well as the required assemblies.

Directory: \OpcClientDA_V2\bin

Table 6-7

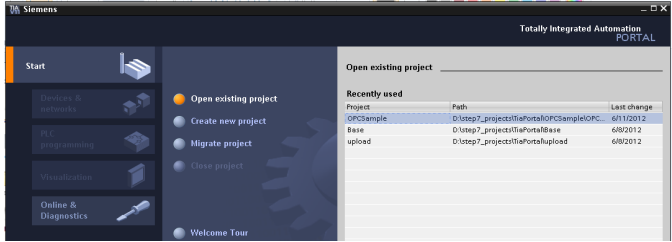
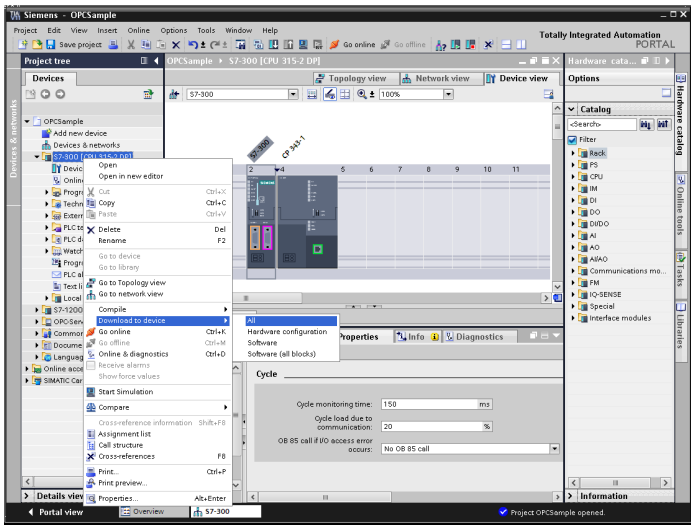
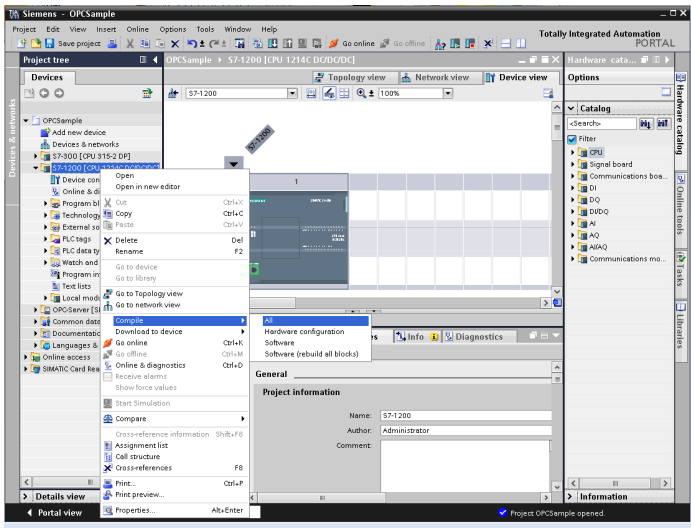
File	Belongs to...
DAClient.exe	Main application
OpcCRcw.Comn.dll	OPC RC Wrapper
OpcRcw.Da.dll	OPC RC Wrapper
ClientAPI.dll	Reusable OPC DA functions

6.5 Loading the simulation to the S7 stations

Loading the TIA project

To do this, proceed as follows:

Table 6-8

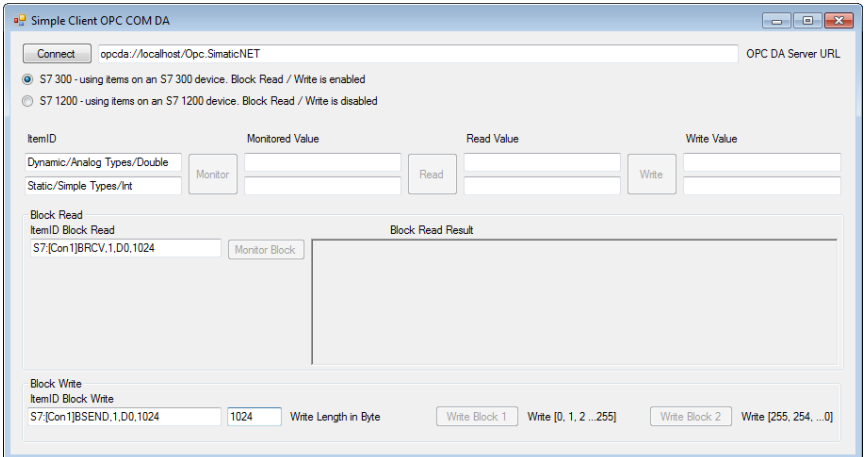
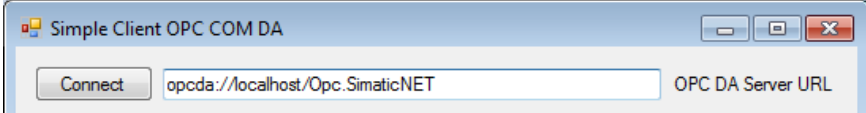
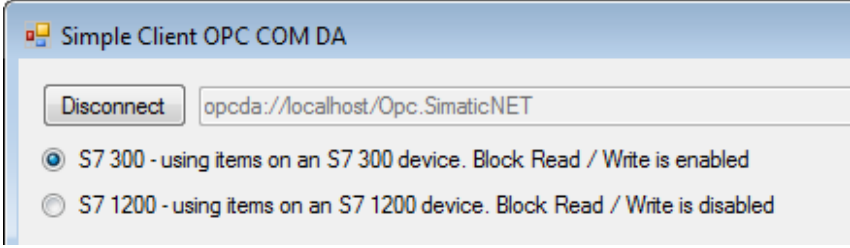
No.	Action	Remarks
1	Extract the TIA project: STEP7_TIA13.zip	Unzip the project in a path in which you have the read and write permissions.
2	Open the TIA portal and navigate to the project via the browser function	
3	Compile and load the S7-300 station	
4	Compile and load the S7-1200 station.	

7 Operating the Application

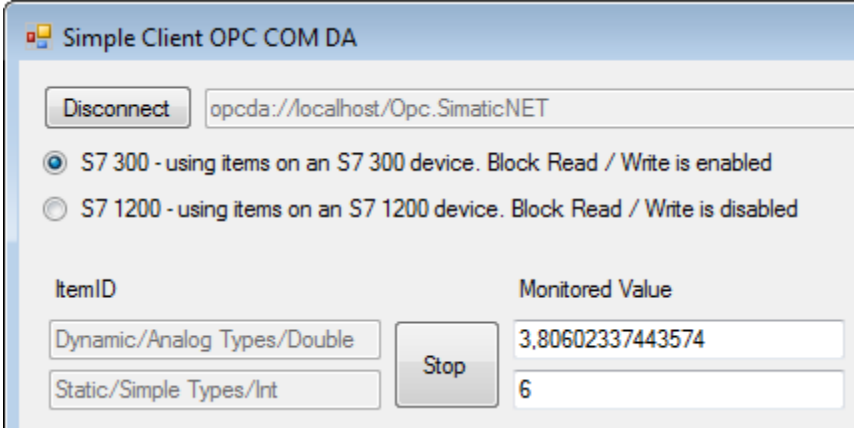
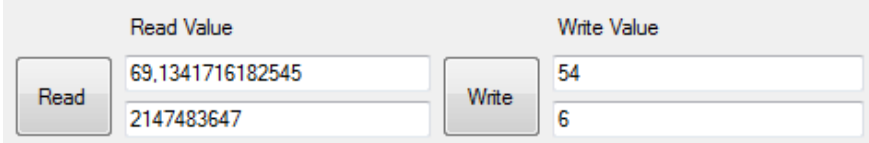
7.1 Overview

The first part for operating the application shows how data of S7 tags are monitored, read and written via direct or symbolic addressing.

Table 7-1

No.	Action	Remarks
1	Start the simple client	<p>After starting the application the user interface will appear.</p> 
2	Establish connection to server	<p>The server URL has to be entered manually. The Connect button can be used to establish the connection to the server.</p> 
3	Select type of controller.	<p>The use of block services in this application is only possible with controllers of the S7-300 family. When selecting S7-1200 the interface elements for block services are disabled.</p> 

7 Operating the Application

No.	Action	Remarks
4	Monitoring of data	<p>For the monitoring of data changes, reading, and writing, two tags can be indicated. As the Item ID, the symbolic name is either specified in the OPC server or addressed directly (e.g. S7:[con1]MB100).</p> <p>Via the Monitor button, monitoring can be switched on. The reported data changes are displayed in the text boxes next to the button. If monitoring is enabled, the button changes the text to Stop.</p>  <p>The screenshot shows a window titled "Simple Client OPC COM DA". It features a "Disconnect" button and a text field containing "opcda://localhost/Opc.SimaticNET". There are two radio buttons: the first is selected and labeled "S7 300 - using items on an S7 300 device. Block Read / Write is enabled", and the second is unselected and labeled "S7 1200 - using items on an S7 1200 device. Block Read / Write is disabled". Below this, there are two "ItemID" input fields: "Dynamic/Analog Types/Double" and "Static/Simple Types/Int". To the right of these fields is a "Monitored Value" column with two text boxes containing the values "3,80602337443574" and "6". A "Stop" button is positioned between the two ItemID fields.</p>
5	Reading and writing	<p>For reading and writing, the same ItemIDs are used as for monitoring. By pressing the "Read" button the two tags are read. Via the two "Write" buttons the tags can be written individually.</p>  <p>The screenshot shows a section with two columns: "Read Value" and "Write Value". Under "Read Value", there is a "Read" button and two text boxes containing the values "69,1341716182545" and "2147483647". Under "Write Value", there is a "Write" button and two text boxes containing the values "54" and "6".</p>

7.2 Using the block services

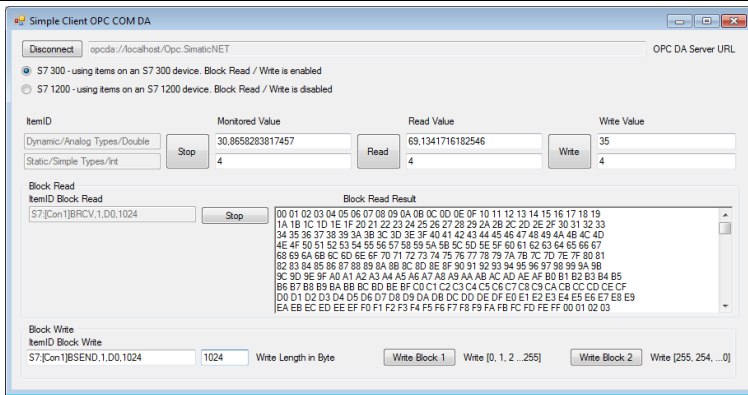
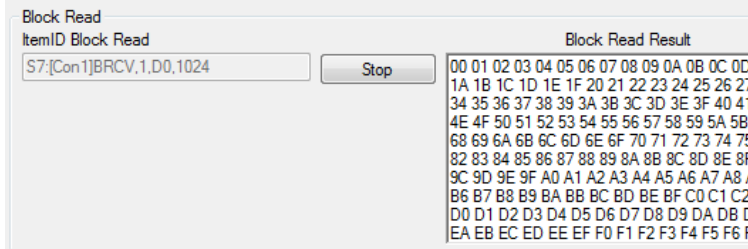
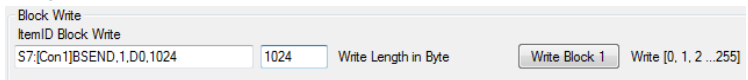
Using the block services

A typical application for block-oriented services is shown here as a clear, simple example: the sending of recipe data to the S7 or the receiving of result data from the S7, whilst using the BSEND and BRCV block services.

The server decides via which service communication takes place with the S7 based on the syntax of the ItemIDs, for example, S7:[S7connection1]BRCV,1,D0,1024].

For the use of the block services, the respective ItemID therefore has to be entered in the fields “ItemID Block Read” or “ItemID Block Write”

Table 7-2

No.	Action	Remarks
1.	Start the Simple Client and connect with the S7 server	 <p>The screenshot shows the 'Simple Client OPC COM DA' window. It is currently 'Disconnected' from 'opcda://localhost/Opc.SimaticNET'. Two connection options are listed: 'S7 300 - using items on an S7 300 device. Block Read / Write is enabled' (selected) and 'S7 1200 - using items on an S7 1200 device. Block Read / Write is disabled'. Below, there are fields for 'ItemID', 'Monitored Value' (30.9656283817457), 'Read Value' (69.1341716182546), and 'Write Value' (35). There are 'Stop', 'Read', and 'Write' buttons. A 'Block Read' section shows 'ItemID Block Read' as 'S7:[Con1]BRCV,1,D0,1024' and a 'Block Read Result' field displaying a long string of hexadecimal data.</p>
2.	Receiving from the S7	<p>For receiving from the S7, “Monitor Block” has to be enabled. As soon as monitoring is enabled, the button changes the text to “Stop”. In the “Block Read Result” text field, the data of the read block is displayed as HEX code.</p>  <p>This close-up shows the 'Block Read' section. The 'ItemID Block Read' is 'S7:[Con1]BRCV,1,D0,1024'. The 'Stop' button is visible. The 'Block Read Result' field contains a long string of hexadecimal characters: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF 00 01 02 03.</p>
3.	Sending of data to S7	<p>Select the recipe by either pressing “Write Block 1” or “Write Block 2”. The first button writes into the data block by incrementing a value for each array entry from 0 onwards. The second button decrements the array entries from 255.</p>  <p>This close-up shows the 'Block Write' section. The 'ItemID Block Write' is 'S7:[Con1]BSEND,1,D0,1024'. The 'Write Length in Byte' is set to 1024. There are two buttons: 'Write Block 1' and 'Write [0, 1, 2 ...255]'. The 'Write Block 2' button is not visible in this view.</p>

8 Glossary

COM / DCOM

COM (Component Object Model): Software model for communication between components, based on a standardized interface.

DCOM: Software model for communicating beyond computer boundaries based on COM.

Event handler

An event handler processes occurring events or Windows messages.

Exception

An exception is referred to as an exceptional situation.

It can be generated either by the operating system (e.g. division by zero) or by the user program.

Exception handler

An exception handler processes occurring exceptional situations. This is usually a secured error behavior and/or a message to the user.

HRESULT

Return data type of COM objects.

IDL

Interface Definition Language: A Microsoft standard language for the definition of function and parameter interfaces.

Polling

Term referring to the (mostly cyclical) polling of certain values or states.

Sink interface

Using the sink interface, messages can be sent between components. The sink interface is based on COM mechanisms.

Thread

Within an application or a process, threads make it possible to execute several code fragments virtually in parallel, meaning at the same time.

If an application uses several threads, this application also has the property "multi-threaded".

If an application only has one thread, it is called "single-threaded". All code fragments for these applications are always processed sequentially.

Windows message

The standard Microsoft Windows operating systems exchange messages to notify of events, e.g. the paint event.

Wrapper

The term “wrapper” normally refers to a class group which encapsulates other class groups for data conversion or easier handling. It can be considered an “envelope” enclosing the “wrapped” classes, covering them from the outside.

9 Related Literature**9.1 Bibliographic references**

This list is not complete and only represents a selection of relevant literature.

Table 9-1

	Topic	Title
/1/	STEP7 SIMATIC S7-300/400	Automating with STEP 7 in STL and SCL Author: Hans Berger Publicis Corporate Publishing ISBN: 978-3-89578-397-5
/2/	STEP7 SIMATIC S7-300/400	Automating with STEP7 in LAD and FBD Author: Hans Berger Publicis Corporate Publishing ISBN: 978-3-89578-296-1
/3/	STEP7 SIMATIC S7-300	Automating with SIMATIC S7-300 inside TIA Portal Author: Hans Berger Publicis Corporate Publishing ISBN: 978-3-89578-357-9
/4/	STEP7 SIMATIC S7-400	Automating with SIMATIC S7-400 inside TIA Portal Author: Hans Berger Publicis Corporate Publishing ISBN: 978-3-89578-372-2
/5/	STEP7 SIMATIC S7-1200	Automating with SIMATIC S7-1200 Author: Hans Berger Publicis Corporate Publishing ISBN: 978-3-89578-355-5

9.2 Internet link specifications

This list is not complete and only represents a selection of relevant information

Table 9-2

	Topic	Title
\1\	OPC	<ul style="list-style-type: none"> OPC DA 2.05 Specification at http://www.opcfoundation.org/
\2\	.NET	<ul style="list-style-type: none"> Inside C#, Tom Archer .NET Crashkurs, Clemens Vasters, Oellers, Javidi, Jung, Freiberger, DePetrillo Microsoft .NET Framework Programming, Jeffrey Richter
\3\	Siemens Industry Online Support	http://support.automation.siemens.com
\4\	Link to this document	http://support.automation.siemens.com/WW/view/en/21043779
\5\	Visual Studio	http://www.microsoft.com
\6\	Cross-platform ILs	http://www.mono-project.com/
\7\	SIMATIC NET Industrial communication Volume 2	http://support.automation.siemens.com/WW/view/en/61630140

10 History

Table 10-1

Version	Date	Modifications
V1.0	05/2005	First version
V2.0	12/2012	Complete revision of STEP7 V11
V2.1	06/2014	Migration to <ul style="list-style-type: none"> - STEP7 V13 - Visual Studio 2010