SIEMENS

Application Description • 08/2014

Data Acquisition and Storage with OPC Client in C# and SIMATIC NET .NET OPC Client API

SIMATIC NET OPC Server

http://support.automation.siemens.com/WW/view/de/21447513

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.

© Siemens AG 2014 All rights reserved

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 5

1.1 Overview 5
1.2 Requirements 7

2 Solution 8

2.1 Overview of the overall solution 8
2.2 Description of the core functionality 10
2.2.1 Software components involved 10
2.2.2 Application procedure 11
2.3 Required hardware and software components 13
2.4 Alternative solutions 14

3 Basics on OPC Application 17

3.1 Overview of the OPC interfaces 17
3.2 Important components of OPC 18
3.3 The programming interface SIMATIC NET .NET OPC Client API 19
3.3.1 Function and structure 19
3.3.2 Overview of the functionalities 20
3.4 Data base interface ADO.NET 22
3.5 Basics on S7 Communication 24
3.5.1 Communication between OPC client and S7 controller 24
3.5.2 Tag services 25
3.5.3 Block services 26
3.5.4 Optimized S7 communication 27

4 Functional Mechanisms of this Application 29

4.1 Logic data interface between controller and OPC Client 29
4.2 Programming of the client whilst using the .NET OPC Client API 35
4.2.1 Integrating the assemblies 35
4.2.2 Setting the compiler 36
4.2.3 Sequence of the example program 37
4.3 Explanations on the simulation program in the controller 39
4.3.1 Process of simulation 39
4.3.2 Function of the blocks 40

5 Configuration and Settings 41

5.1 Configuration of the PC station 41
5.1.1 Importing the XDB file to the Components Configurator 42
5.1.2 Changing the IP address of the PC station in STEP 7 V1x 44
5.2 Configuring an OPC server 46
5.2.1 Configuring the OPC server in STEP 7 V1x 46
5.2.2 Configuring the S7 connections 50
5.2.3 Final check of the settings 54

6 Installation and Commissioning 56

6.1 Hardware installation 56
6.2 Software installation 57
6.2.1 De-archiving the application software 57
6.2.2 Installing of the standard software 57
6.2.3 Installing the OPC client on the PC/PG 57
6.3 Commissioning the application 59

7 Operating the Application 62

© Siemens AG 2014 All rights reserved

7.1 General operational steps at the user interface 62
7.2 Operation of the “Data acquisition” scenario 63
7.3 Operation of the “Download Recipe” scenario 64

8 Links & Literature 66

9 History 66

© Siemens AG 2014 All rights reserved

1 Task

1.1 Overview

Introduction

In the automation world, there are various, often vendor-dependent, communication interfaces. Nevertheless, field devices from different manufacturers must exchange their data with each other.

The application on hand illustrates a possible procedure for generating individual OPC clients in C#, suitable for mass data recording in the automation world.

The following scenarios are standard in automation technology:

- The control program collects data from the process and provides the data to be stored in a database on the PC side (data acquisition).
  Example: a production step has been completed and the parameters from this step are to be documented for quality assurance purposes.
- The control program needs production data in order to continue the process (recipe download).

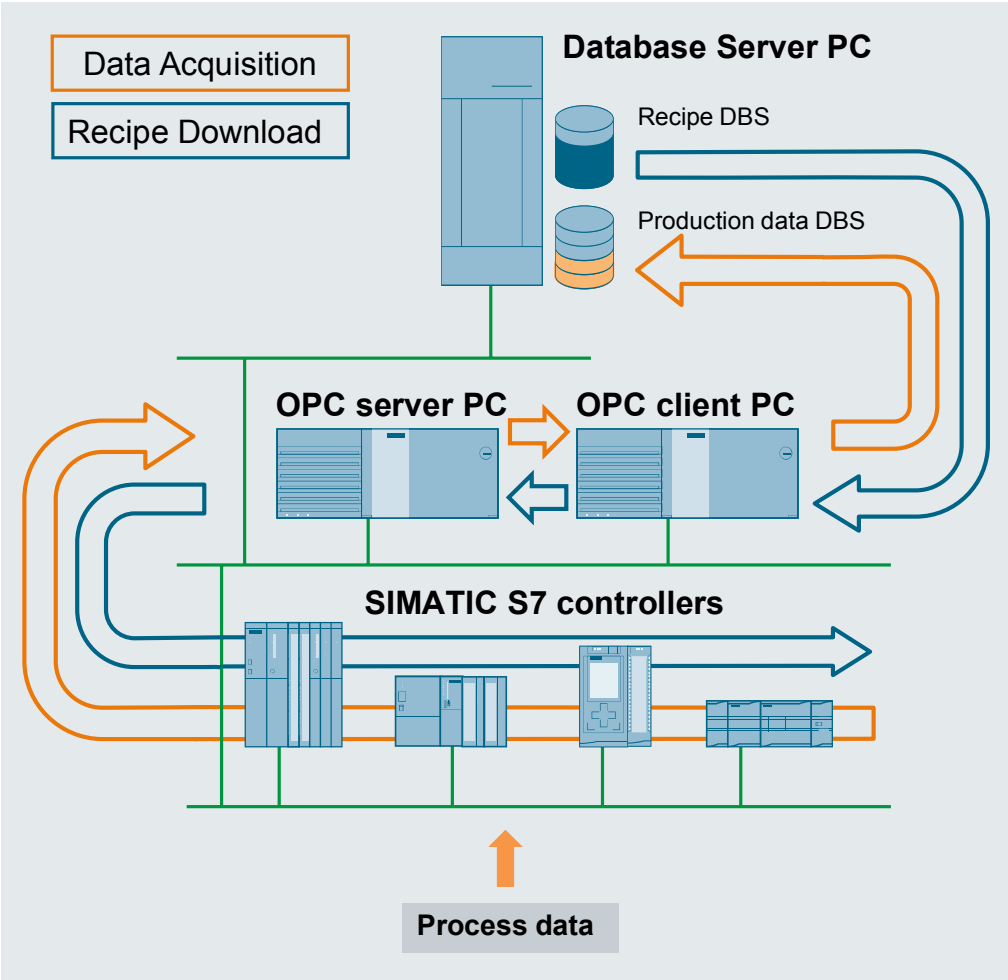
  Example: the production step for the next workpiece is to be started. Relevant parameters are stored in a database. The data must be read from the database by the PC and must be written onto the PLC.

Copyright © Siemens AG 2014 All rights reserved

Overview of the automation task

The following figure gives an overview of the general automation task:

Figure 1-1



Description of the automation task

Process data available in various PLCs must be stored in one database on a central server. Recipes from a recipe database, on the other hand, are stored into the respective storage sections of the PLC. For connecting the data base to the process, the industrial standard OPC is used. Logic and controlling are performed on the OPC client.

Copyright © Siemens AG 2014 All rights reserved

1.2 Requirements

There are different requirements to the individual components.

Requirements for operating and monitoring software for the visualization

The software is to enable the fast and simple creation 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.

Requirement for the data interface between PC program and controller

Process data are exchanged via the standardized OPC DA interface:

- Connection to the process data via Industrial Ethernet as well as the SIMATIC NET OPC Server V8.1 (or higher).
- Using the OPC DataAccess interface
- Universal programming of the client for the COM/DCOM as well as for the OPC UA server from SIMATIC NET
- Symbolic addressing of the process data
- Reading and writing of process data
- Monitoring of process data

Requirement for data storage

The OPC client stores data from the SIMATIC S7 via the standardized methods of ADO.NET. Two possibilities of saving data are illustrated here:

- Saving the data to a Microsoft Access data base
- Saving the data to an Excel csv file.

Requirements for the development environment

The current Windows development environment is to be used:

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

Copyright © Siemens AG 2014 All rights reserved

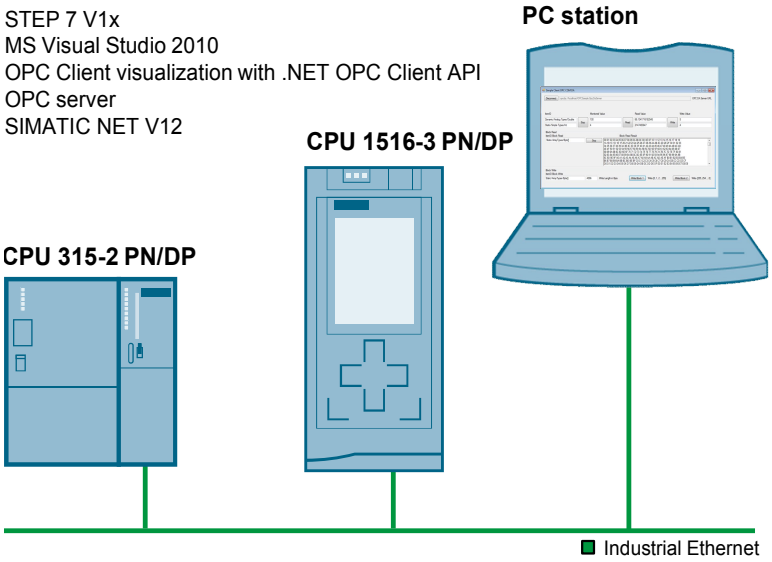
2 Solution

2.1 Overview of the overall solution

Schematic layout

The figure below shows the most important components of the solution:

Figure 2-1



Structure

On the controller side, there is a SIMATIC S7-300 station with a CPU 315-2 PN/DP as well as a S7-1500 station with a CP 1516-3 PN/DP. The programs of both stations are used for illustration and are identical and not optimized for symbolic programming.

At the PC station, the SIMATIC NET OPC server as well as the “OpcDataTransaction” OPC client developed for this application run with a very simple user interface for demonstrating the basic function.

Copyright © Siemens AG 2014 All rights reserved

Content of the example

This application example shows the coupling of a production process to a Windows-based PC with a very simple data exchange via OPC that can be realized very quickly.

In particular, the procedure for the implementation of a simple but universal handshake between the OPC client application and the S7 controller is demonstrated. With the handshake principle two scenarios can be illustrated:

- A coordinated data acquisition by the controller
- A coordinated recipe download to the controller

The respective state machine is simply controlled on the SIMATIC S7 side by a watch table.

Advantages

The solution demonstrated here uses the .NET OPC Client API as the interface for the programming of the OPC client. It allows the programmer to create an OPC communication under .NET rather simply. In addition, this API can be used with the SIMATIC NET OPC server either for the OPC DA server (COM/DCOM) or for the more modern OPC UA server.

Delimitation

This application does not contain a complete description

- of the .NET framework,
- of C# or VB.NET,
- of the ADO.NET interface
- of the OPC interface

Assumed knowledge

Basic knowledge

- in the area of object-oriented programming
- in programming language C#
- in OPC technologies

is assumed.

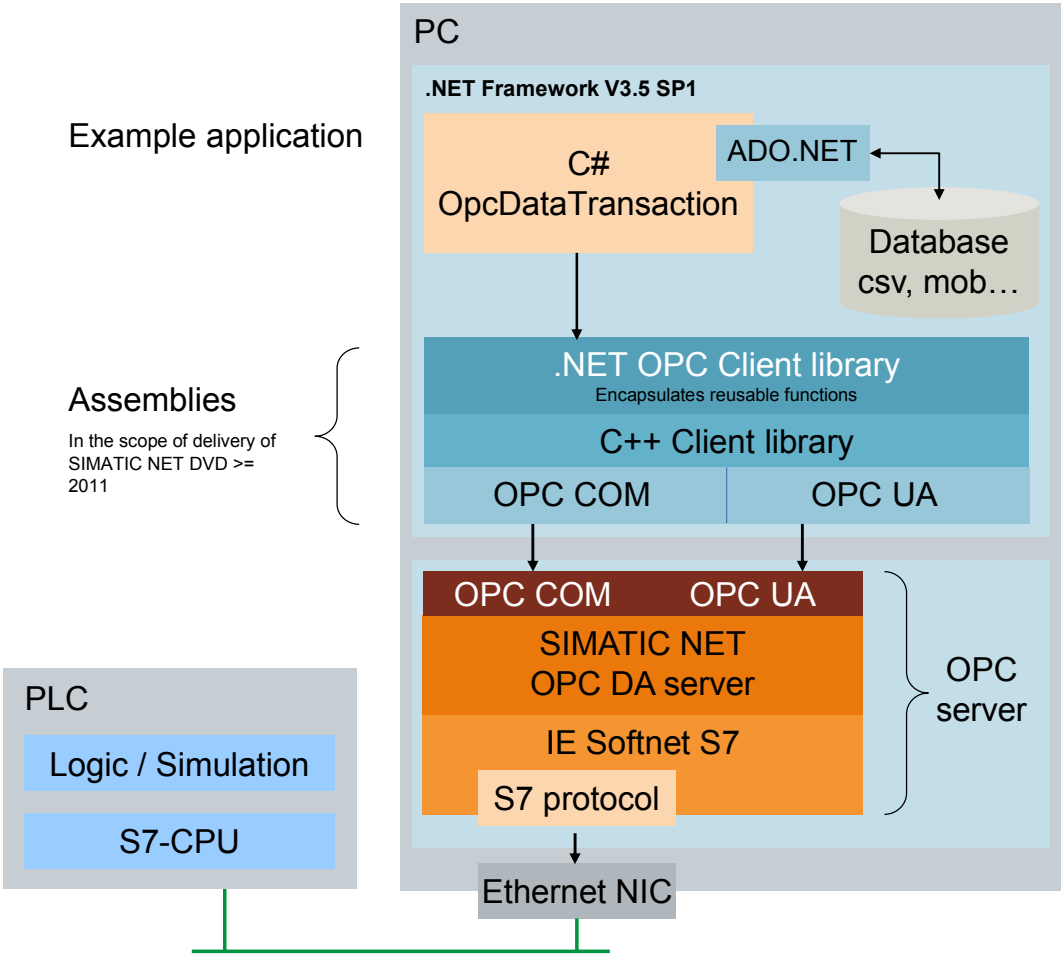
Copyright © Siemens AG 2014 All rights reserved

2.2 Description of the core functionality

2.2.1 Software components involved

The following figure shows the involved software components. Please find more detailed descriptions in chapter 3.

Figure 2-2



PG/PC

On the PG/PC, the application "OpcDataTransaction" for controlling transactions between the PLC and the C#-OPC client has been realized.

For coupling to the process, the OPC client uses the .NET OPC Client API that is automatically installed with SIMATIC NET OPC Server V8.x. The two assemblies (SimaticNET.OPC.Common and SimaticNET.OPC.OpcClient) used in this example encapsulate the OPC DA interface and offer the client a simple usage option from .NET applications. The data acquired by OPC are stored via the ADO.NET interface in a database or in a CSV file.
The SIMATIC NET OPC server establishes the connection to the controller via the SIMATIC NET SOFTNET IE S7 connection.

Note For further information on .NET OPC Client API, the ADO .NET interface and the S7 communication, please see chapter 3 Basics.

Copyright © Siemens AG 2014 All rights reserved

Control

The controller provides the data to be stored. For this purpose, a simple S7 program for simulating various data types is implemented. The transaction logic is only controlled via a watch table in this example.

Created software components

- OPC client (developed in C#)
- STEP 7 simulation program

2.2.2 Application procedure

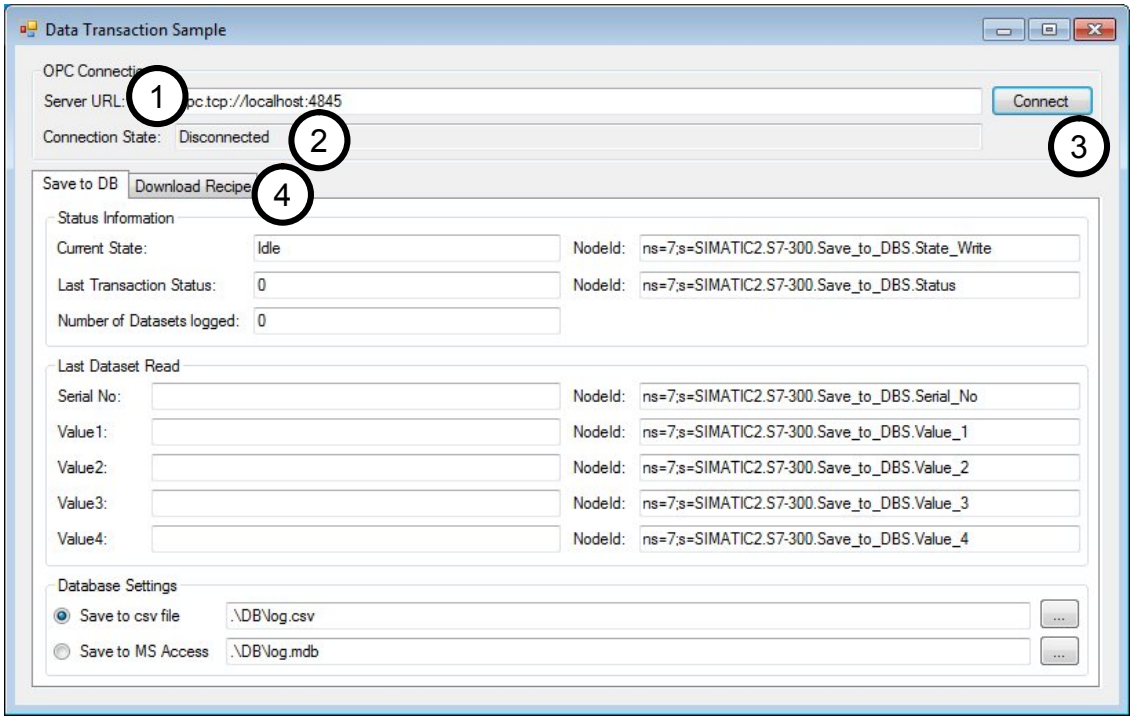
Overview

The client “OpcDataTransaction” shows how to read data from an OPC server and how to save them by means of the .NET OPC Client API data. The C# sample code shows how the data are written asynchronously in an extra thread into a csv-file or an MS Access database via the ADO.NET interface.

Description of the HMI

A short overview of the user interface is given below. For the exact description of the individual controls and functions, please see chapter 7.

Figure 2-3



Copyright © Siemens AG 2014 All rights reserved

The marks have the following meaning:

Table 2-1

No.	Description
1	The Server URL states to which server the client is to connect. This string also defines whether the connection is established via COM DA or OPC UA. (See note below)
2	Connection State indicates the current status of the OPC connection to the server. Possible states are: • Undefined • Disconnected • ErrorShutdown • ErrorWatchdog • Connected
3	The Connect button serves for establishing or terminating OPC connections to the server.
4	Tabs to switch between the two examples “Data acquisition” and “Download Recipe” (see also chapter 4.1 and chapter 7)

Note The SIMATIC NET OPC Data Control used here for .NET can only be used for the coupling with SIMATIC NET OPC servers.

Copyright © Siemens AG 2014 All rights reserved

2.3 Required hardware and software components

Hardware components for the controller

Table 2-2

Component	No.	Article number	Note
CPU 315-2 PN/DP	1	6ES7315-2EH14-0AB0	or a comparable S7-CPU
CPU 1516-3 PN/DP	1	6ES7516-3AN00-0AB0	or a comparable S7-CPU

For simulating the functions of the OPC client, it is sufficient to use one of the two CPUs.

Hardware components for the PC

Table 2-3

Component	No.	Article number	Note
Field PG M3	1	6ES7715-…	or a standard PC
NDIS-capable network card	1	Depending on product	Integrated in the field PG

Standard software components

Table 2-4

Component	No.	Article number	Note
STEP 7 Professional V11, SP2	1	6ES7822-1A.01-	optional; no S7-1500 CPU configured
STEP 7 Professional V12, SP1	1	6ES7822-1A.02-..	
STEP 7 Professional V13, SP1	1	6ES7822-4AA03-0YA5	With configured S7-1500 CPU
SIMATIC NET DVD V8.2 SOFTNET IE S7	1	6GK1704-1LW08-2AA0 6GK1704-1CW08-2AA0	LW=8 S7 connections (Lean), CW=64 S7 connections
Visual Studio 2010 with C# components	1	Depending on product Can be ordered via your administrator or http://www.microsoft.com/	
.NET Framework 3.5	1	Free download at http://www.microsoft.com/	Installed by SIMATIC NET. Also to be used in Visual Studio

Copyright © Siemens AG 2014 All rights reserved

Sample files and projects

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

Table 2-5

Component	Note
21447513_Csharp_OpcDataTransaction_CODE.zip	• Source code of the user Interface • OPC Client • STEP 7 projects V11/V12 (TIA Portal) STEP 7 V13 with S7-1500 CPU
21447513_Csharp_OPCDataTransaction_DOKU_V2_1_en.pdf	This document

2.4 Alternative solutions

Here, you learn which alternatives for programming an OPC client under .NET exist and which characteristics they have.

Copyright © Siemens AG 2014 All rights reserved

Decision criteria for using OPC interfaces

The following table shows you the most important decision criteria for selecting the OPC interface to be used:

Table 2-6

Deciding criteria	Description
COM DA Interface	Is the use of OPC DA interface with reading/writing/ monitoring sufficient for the application?
COM A&E interface	Is the processing of events and process alarms pursued (for the future)?
OPC UA Interface	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.

The variant realized in this example is highlighted in blue in this table.

Table 2-7

Criterion	OPC Connector	OPCClient.API	OPCDa.RCW	OPCUa.RCW
COM DA Interface	✓	✓	✓	-
COM A&E interface	-	-	-	-
OPC UA Interface	✓ (*)	✓ (*)	-	✓
Security	- (**)	- (**)	- (**)	++
Platforms	-	-	-	✓
Functionality, flexibility	-	+	++	+++
Combination with other manufacturers	- (***)	- (***)	✓	✓

(*) only DA functions

(**) security only via DCOM

(***) classic OPC DA and OPC UA parallel, limited to SIMATIC NET OPC servers

Using the Standard OPC Foundation .NET Wrapper (Raw Interface)

Principally, the standard OPC .NET API (.RCW) library can be used (see chapter 3.3)

This has the following advantages and disadvantages:

Table 2-8

Advantages	Disadvantages
• Connection to OPC servers of other manufacturers possible. • Full function of the API can be used (full flexibility)	• Complex API complicates its use. • Basic functionality of a client (e.g. reconnect in the event of a disconnection) has to be individually implemented. • Administrative functions have to be implemented individually. • No support of OPC UA.

Note The SIMATIC NET OPC Client API controls can only connect with SIMATIC NET OPC servers but not with OPC servers of other manufacturers.

Copyright © Siemens AG 2014 All rights reserved

Examples for the alternative .NET OPC libraries

The following table lists some other interesting OPC examples under .NET.

Table 2-9

Example	OPC library used	
Programming of OPC DA .NET Clients with C# for the SIMATIC NET OPC Server (COM/DCOM)	OPCDa.RCW	http://support.automation.siemens.com/WW/view/en/21043779
	OPC DataConnector	
Programming an OPC UA .NET Client with C# for the SIMATIC NET OPC UA Server.	OPCUa.RCW	http://support.automation.siemens.com/WW/view/en/42014088
Individually Programmed Signaling System based on the SIMATIC NET OPC Alarm and Event/OPC UA Server.	OPCAE.RCW + individual Client API	http://support.automation.siemens.com/WW/view/en/26548467
	OPCAUa.RCW + individual Client API	

Copyright © Siemens AG 2014 All rights reserved

3 Basics on OPC Application

In recent years, the OPC Foundation (an interest grouping of well-known manufacturers for the definition of standard interfaces) has defined a large number of software interfaces to standardize the information flow from the process level to the management level.

Note For further information about the OPC and concrete applications, please refer to the Siemens Industry Online Support (http://support.automation.siemens.com/WW/view/en/20229805/136000), and the respective OPC specification of the OPC foundation (http://www.opcfoundation.org/).

3.1 Overview of the OPC interfaces

Classic OPC interfaces

According to the different requirements within an industrial application, different OPC specifications were developed in the past:

- Data Access (DA): access to process data
- Alarm & Events (A&E): interface for event-based information including acknowledgment
- Historical Data Access (HDA): function for archived data
- Data eXchange (DX): server for cross-communication

COM / DCOM

These software interfaces have been defined according to the rules of Microsoft COM (Component Object Model) and can therefore be easily integrated into Microsoft operating systems. COM or DCOM (Distributed COM) provides the functionality of inter process communication and organizes the information exchange between applications, even across network boundaries (DCOM).
The use of COM or DCOM causes OPC servers and clients to run only on a Windows PC or in the local network and that the communication to the respective automation system has to be realized mainly via proprietary protocols.
Additional tunneling tools have to be used for the network communication between client and server in order to get through firewalls or to avoid the complicated DCOM configuration. The interface can furthermore only be accessed natively with C++ applications; .NET or JAVA applications can only gain access via a wrapper layer. In practice, these restrictions lead to additional communication and software layers which increase the configuration workload and complexity.

OPC unified architecture

To solve the mentioned restrictions in real-life situations and to fulfill the additional requirements, the OPC Foundation has defined a new platform in the last five years, called OPC Unified Architecture, which offers a uniform basis for the exchange of information between components and systems.
OPC UA will also be available as an IEC 62541 standard and therefore forms the basis for other international standards.

Copyright © Siemens AG 2014 All rights reserved

The aim of this new standard is the generic description and uniform access to all information which is to be exchanged between systems or applications. This includes the functionality of all previous OPC interfaces. Furthermore, it is to generate the possibility of natively integrating the interface in the respective system, irrespective of which operating system the system is operated on and irrespective of the programming language in which the system was created.

This example discusses the OPC Unified Architecture interface.

Features of OPC UA

OPC UA offers the following features:

- Summary of all previous OPC features and information such as DA, A&E and HDA in a generic interface.
- Use of open and platform-independent protocols for inter-process or network communication.
- Internet access and communication by means of firewalls.
- Integrated access control and security mechanisms on protocol and application level.
- Extensive representation options for object-oriented models.
- Objects can have tags and methods and can trigger events.
- Expandable type system for objects and complex data types.
- Transport mechanisms and modelling rules form the basis for other standards.
- Scalability of small embedded systems up to business applications and from simple DA address spaces up to complex, object-oriented models.

Copyright © Siemens AG 2014 All rights reserved

3.2 Important components of OPC

OPC Client

By the active calling of functions in the server, the OPC client informs the OPC server that it desires information about or access to certain tags.

The following access methods are possible:

- Reading
- Writing
- Monitoring tags (reporting value changes).

OPC server

The OPC server is the central communication unit between the OPC client and the respective data source (here the S7 controller).

The SIMATIC NET OPC server generally provides DA and US interfaces. The library used in this example can serve both interfaces.

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. In this example, the S7 protocol is used which contains two principle transmission mechanisms (see also chapter 3.5):

- the tag services and
- the block services.

In this example only the tag services are used. The connections are configured in TIA Portal (see chapter 5.2).

API

API refers to an Application Programming Interface, which contains a number of software functions. These functions are used by an application to get access to certain functions of another application or an operating system.
This example uses the SIMATIC NET .NET OPC Client API for creating the OPC client (see also chapter 3.3).

3.3 The programming interface SIMATIC NET .NET OPC Client API

3.3.1 Function and structure

Function

The SIMATIC NET .NET OPC Client API offers a programming interface for developers in the languages C# and Visual Basic .NET. The simple and intuitive interface allows for programming OPC Client applications which can process both data from COM DA and from UA servers.

So the application can access to DA items or UA nodes transparently via the same functions.

In case of machine access or network limits the UA interface should be chosen, since a secure connection between the client and the server can be built up much easier than with the COM/DCOM-based “classic” OPC DA.

Structure

The SIMATIC NET .NET OPC Client API contained in the SimaticNET.OPC.OpcClient.DLL library on the SIMATIC NET DVD mainly offers two categories of functions:

- the namespace of the actual access to OPC process data (DA or UA interface) - SimaticNET.OPCDaClient.
- a namespace with a rather general functionality for finding information about OPC servers (DA and UA) necessary for building up a connection - SimaticNET.OpcCmn (isn't used in this example).

Copyright © Siemens AG 2014 All rights reserved

3.3.2 Overview of the functionalities

Namespace SimaticNET.OPCDaClient

The following functionality provides the namespace **SimaticNet.OpcDaClient** of the .NET component via the “DaServerMgt” object:

Table 3-1

No.	Function	Explanation
1.	Connecting to the OPC server	With the method “Connect” the connection to the OPC server can be built up and can be disconnected with the “Disconnect” method. The connection is supervised by the .NET component. Should there be connection errors, the status changes are reported through the “ServerStateChanged” event.
2.	Reading and writing of OPC Data Access items	With the methods “Read” and “Write” (or “ReadAsync” and “WriteAsync”) the values from the OPC items can be read and written synchronously and asynchronously.
3.	Reporting data changes	The .NET component offers a mechanism for reporting value changes. No more cyclic reading. With the “Subscribe” method, items can be registered for supervision and with “SubscriptionCancel” they can be withdrawn again. Changed values are reported via the “DataChanged” event.
4.	Finding information on the address space.	The “Browse” method is for searching the address space of an OPC Data Access server for OPC items. With the “GetProperties” method, features of the OPC items can be determined.

Namespace Simatic NET.OpcCmn

In the namespace SimaticNet.OpcCmn general OPC functions are combined. This applies especially to searching for OPC servers and finding the information necessary for establishing the connection.

In the present example the functions of this namespace are not changed!

Table 3-2

No.	Function	Explanation
1.	Searching for OPC COM servers	With the method “EnumComServers” OPC servers on a computer can be found. It is possible to differentiate between the different OPC specifications such as “OPC Data Access”, “OPC Historical Data Access” or “OPC Alarm & Events”.
2.	Finding COM OPC server names	The “ClsidFromPorgId” method determines the corresponding “CLSID” of the OPC server, depending on the transmitted parameters computer name and “ProgID”.
3.	Searching for OPC UA servers	The “getCertificateForEndpoint” method combines to form the transmitted endpoint URL and loads the server certificate which is returned as Out parameter. You will get the In parameters before by browsing on the Discovery server.
4.	Finding the UA endpoint.	The “EndpointIdentifier” class supplies all the data additionally necessary for building up a connection with an UA server.
5.	Creating the UA certificate	The class “PkiCertificate” encapsulates an X509 certificate and allows for a simple creation of such a certificate. Furthermore, methods for an access on the "WindowsCertificateStore" for loading and saving certificates are provided. With Properties, the fields of the certificate can only be read.

Copyright © Siemens AG 2014 All rights reserved

Note With this API only connections with OPC servers of the SIMATIC NET product family can be built up. Connections to other OPC servers or to OPC servers of other manufacturers are not possible.

Note A detailed description of the interface can be found in the Siemens Industry Online Support in the “SIMATIC NET Industrial Communication with PG/PC Volume 2 - Interfaces Programming Manual”(http://support.automation.siemens.com/WW/view/en/61630891)

Copyright © Siemens AG 2014 All rights reserved

3.4 Data base interface ADO.NET

Function

ADO.NET (ActiveX data object .NET) is a set of classes which make data access services available for a great number of database systems for .NET framework programmers. The database interface is object-oriented.

The object model provides a great number of components for creating distributed applications with data release. The most important ones are:

- DataSet: a central element for receiving and combining several tables.
- DataTable: a table in DataSet
- DataReader: for reading data
- DataAdapter:
  - creates the connection to the actual databases,
  - fills the DataSet with information,
  - writes the changed data sets back to DataSet.

Notes

The following points must be noted with regards to the ADO.NET application model:

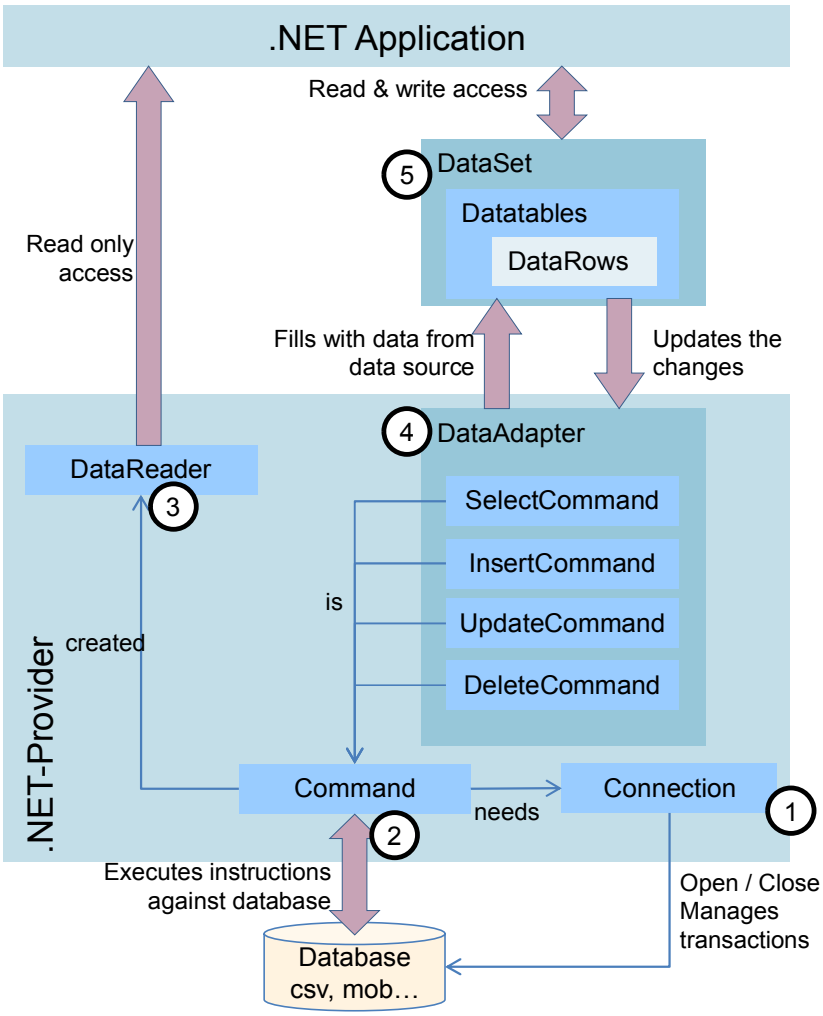
- The data volume held in the memory must be observed.
- If a client stores data to the data base, the other clients will not necessarily be aware of this (due to transient data management).
- Changes at the database structure must be adjusted in DataSet.
- The DataSet should be resynchronized, during the update in order for the data of the data base to correspond with the data in DataSet. This can be initiated by a simple parameter specification.

For further information, please refer to the secondary literature.

Copyright © Siemens AG 2014 All rights reserved

Application model of ADO.NET

Figure 3-1



Explanation

The numbers have the following meaning:

Table 3-3

No.	Object / Action	Explanation
1.	A Connection is used to open a connection to the database.	• Connections are now only opened when they are necessary. • Connection pools are created as soon as “Open” is called. • A Close call does not delete the Connection from the Connection pool.
2.	Command objects mainly execute SQL statements against the database.	
3.	DataReader enables performant reading of a forward directed data stream from the data base.	The DataReader can only be used for reading data from the data base.

Copyright © Siemens AG 2014 All rights reserved

No.	Object / Action	Explanation
4.	The DataAdapter provides an interface to a DataSet (5) using the respective command.	• DataAdapter corresponding with a DataSet or a DataTable. • In order to execute an Update command with a DataAdapter, it must only contain one DataTable!
5.	A DataSet contains the data of the database.	The DataSet is a representation of the database structure and its data in the memory of the PC. • This enables a transient saving of selected data of the data base. Transfer of database data via XML and WebServices is possible without greater problems. • The DataSet object can also be used for creating or modifying relations between DataTables. It is therefore possible to adjust the data model in the transient memory.

3.5 Basics on S7 Communication

3.5.1 Communication between OPC client and S7 controller

The S7 communication contains two different methods of data transmission:

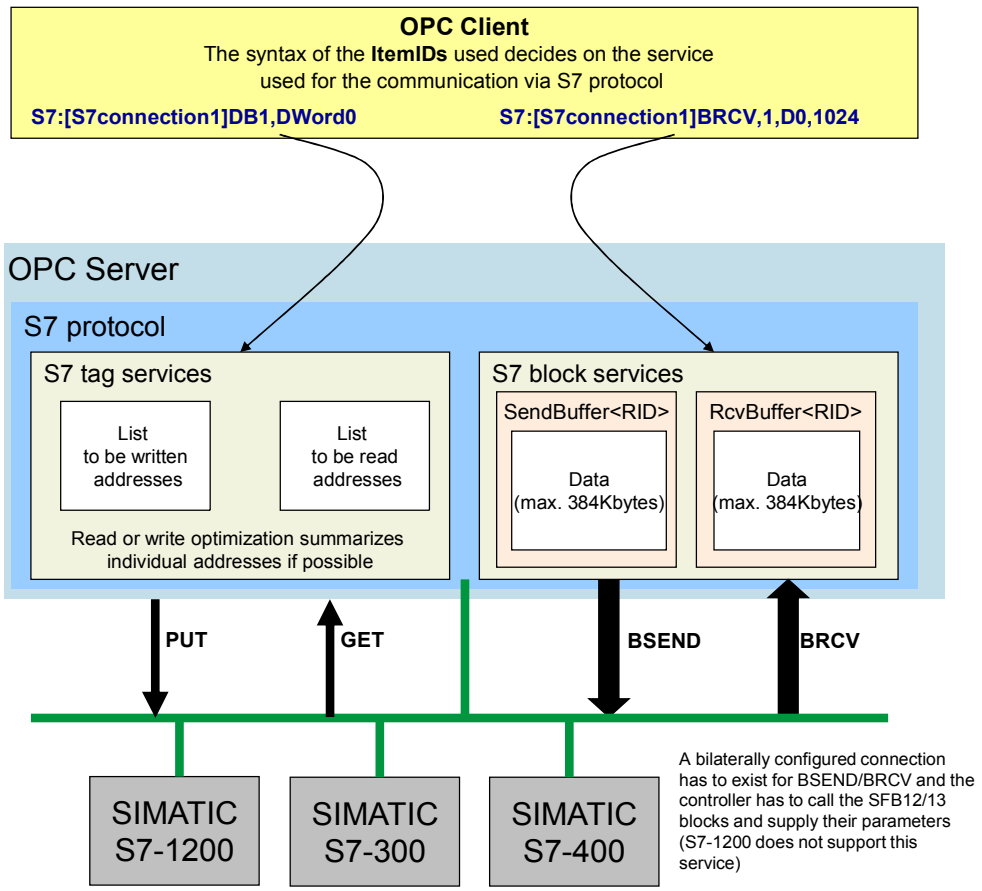
- Tag services
- Block services

On the level of the OPC server they are almost completely covered up. The communication service used for the controller can only be detected on the basis of 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 the communication to the S7 controller is to take place. Here, the connection name identifies the communication partner (this name represents an IP address for example) 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 of this data.

Copyright © Siemens AG 2014 All rights reserved

Figure 3-2 Services and items for COM OPC DA server



Copyright © Siemens AG 2014 All rights reserved

3.5.2 Tag services

An S7 controller replies to requests via tag services; for this purpose only a unilaterally configured connection is necessary.
Every S7 controller is a so-called “S7 server” and answers PUT/GET requests in the control program without any further implementation. 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.

Note For the SIMATIC S7-1500 controls, the tag services need to be enabled separately in Settings under “Security”.

Note In this example, only the tag services are used.

ItemIds for tag services

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

Example: S7:[S7-connection_3]DB10,W20

This is a “word”-type tag (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 by 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 STEP 7. 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 STEP 7 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 from the controllers via PUT/GET. Symbols which represent a BSEND or BRCV tag cannot be generated by STEP 7.

Copyright © Siemens AG 2014 All rights reserved

3.5.3 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 “S7connection_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 “S7 connection_5”). On a S7 connection, several BSEND/BRECV pairs belonging together and connected via their RID can exist. 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 to the send buffer of 1024 byte length. The range of 100 to 140 is overwritten in the 1024 byte size buffer. The entire block with ID 1 is sent to the communication partner who has to provide a BRECV 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.

Note The use of the BSEND/BRCV block services is only possible with SIMATIC S7-300/S7-400/ S7-1500, as well as S7-1200 from FW V3.0 .

Further information

In the following document in the Industry Online Portal, we show you all the OPC services based on the possibilities of the S7 communication.

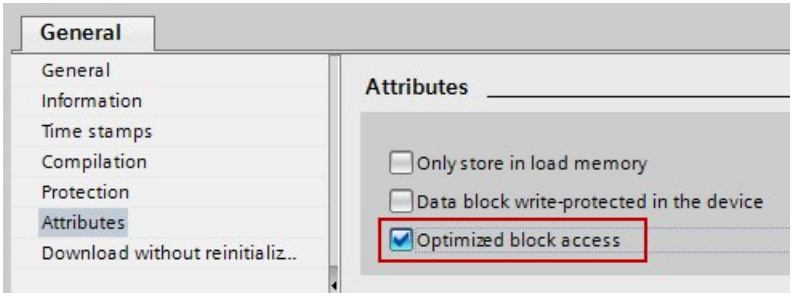
"Data communication between the S7 Station and the PC Station, using the SIMATIC NET OPC Server" - http://support.automation.siemens.com/WW/view/en/67295801

3.5.4 Optimized S7 communication

Background

With an S7 connection, access to optimized data blocks in the S7-1200/S7-1500 CPUs is no longer possible. In order to fetch data from an S7-1200/S7-1500 via an S7 connection using an OPC server, the data blocks to be read from must not be optimized.

Figure 3-3 Properties of a data block



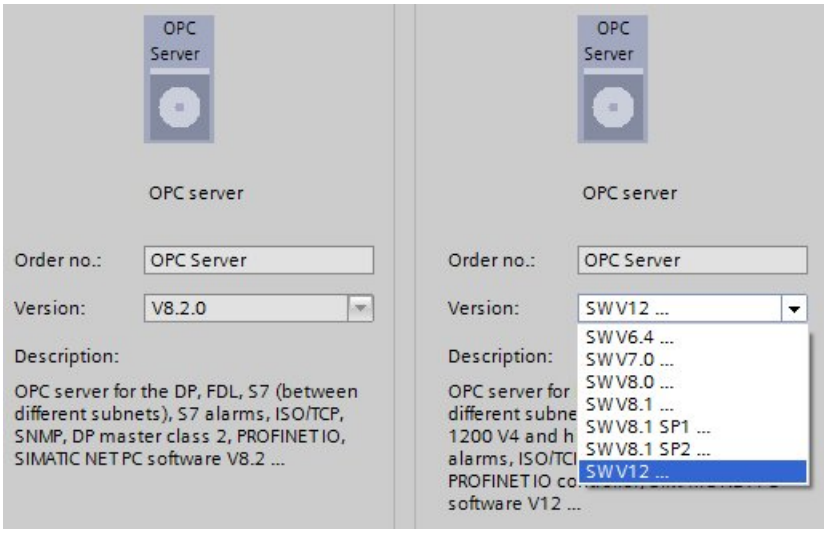
Disadvantage: the performance of the innovated controllers S7-1200/S7-1500 is affected by using non-access-optimized data blocks.

Remedy

In order to avoid performance loss in the S7-1200/S7-1500 controllers due to non-access-optimized data blocks, the so-called "optimized S7 communication" applies as of SIMATIC NET OPC V12.

Copyright © Siemens AG 2014 All rights reserved

Figure 3-4



This connection type is created automatically when using SIMATIC NET OPC Server V12 and an innovated controller. Access to access-optimized data blocks is then also possible using the optimized S7 connection.

Access with OPC Client

The point of access to the optimized S7 connections is realized via the SimaticNET.S7OPT server on port 4850.

Copyright © Siemens AG 2014 All rights reserved

4 Functional Mechanisms of this Application

In the following section we will explain the internal procedures in the OPC client. For details on C# programming, please refer to the well documented source code also contained in this example.

4.1 Logic data interface between controller and OPC Client

Overview

This example shows the following transfer mechanisms:

- Data acquisition and storage
- Download recipe.

Description of the “Data acquisition” function

The control program provides a set of data to be saved to the database. To guarantee the reliability it must be made sure that the control program only overwrites the data when the OPC client has consistently read out and saved the complete set. The OPC client and the control program must therefore be synchronized via a status tag. This **State_Write** status tag is monitored by the OPC client throughout the entire runtime.

The following table gives an overview of all the tags used for the acquisition of data in the SIMATIC S7-CPUs.

Table 4-1

Symbolic name	Description
"Save_to_DBS".State_Write	Status tag for the synchronization of the OPC client and the control program. The following status values are defined: • Idle = 0 • TransferToDataBase = 1 • TransferReady = 2 • Error = 3
"Save_to_DBS".Serial_No	Example data for saving to the database. Data type: Integer
"Save_to_DBS".Value_1	Example data for saving to the database. Data type: String
"Save_to_DBS".Value_2	Example data for saving to the database. Data type: Float.
"Save_to_DBS".Value_3	Example data for saving to the database. Data type: Word
"Save_to_DBS".Value_4	Example data for saving to the database. Data type: DateTime:
"Save_to_DBS".Status	Tag indicating the status of the last database operation. The OPC client can signal to the controller that an error occurred while saving to the database. The control program must then decide whether it continues anyway or whether it stops until the error is corrected.

Copyright © Siemens AG 2014 All rights reserved

Process of data acquisition and storage

The figure below shows the data flow and the process of data acquisition, as it has been implemented in the example. Every step is then described in the subsequent table.

Figure 4-1

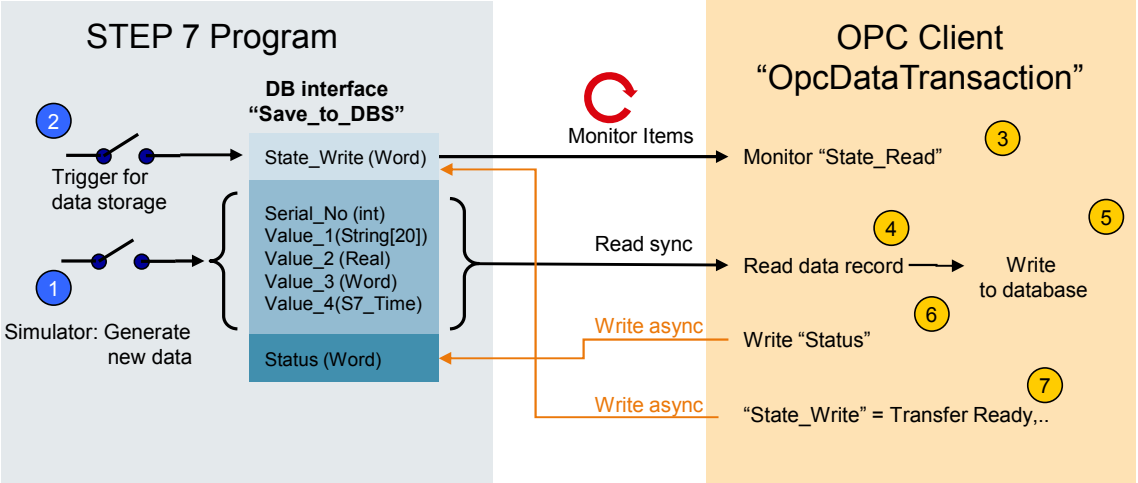


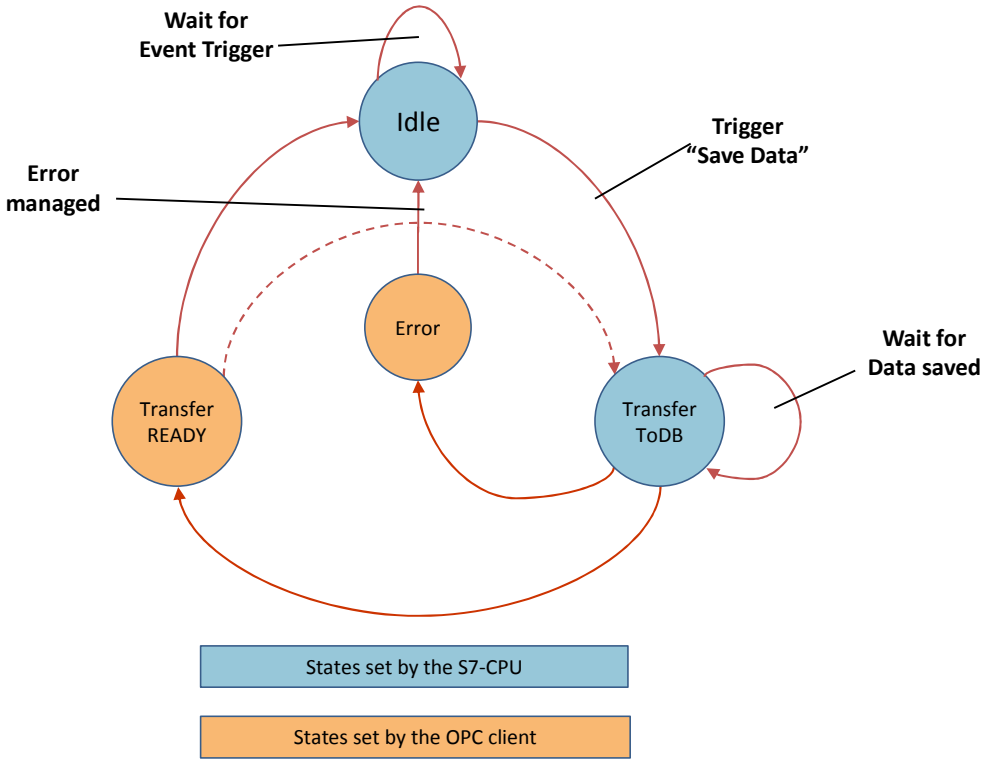
Table 4-2 Logic steps of a sequence of data acquisition and storage

No.	Description
1	With the tag table, the user writes the value "true" to the **SimulateNewRecordSet** tag, which triggers the simulation of new data in the controller. The data is transferred to the DB Save_to_DBS.
2	With the tag table, the user changes the status from "Idle" (0) for example, to "TransferToDatabase" (1) into the status tag **State_Write** in the data block **Save_to_DBS**. This triggers a data transfer.
3	The OPC client (cyclically) monitors the tag and detects any changes in the value of the **State_Write** tag and displays the new client status ("TransferToDatabase"). The OPC client now knows that now data is available at the controller and that the controller waits for this data to be collected.
4	The OPC client reads the complete data set from the controller via the OPC server.
5	The data set is written by the OPC client into the database or .csv file via ADO.NET methods.
6	The OPC client writes the result (success / failed) of the saving operation to the tag status (Ok, ReadError, WriteError, DBError) In case of an error, the logic in the controller can now decide how to proceed. If the operation was successful, the controller can continue with the next command.
7	As the last step in the chain, the client sets the current status in the controller to "TransferReady" (2). The controller now knows that the OPC client has completed the operation and can set the status to "Idle" (0) again, or trigger a new transfer.

Copyright © Siemens AG 2014 All rights reserved

The figure below shows the status transitions in the “data transfer and storage” mode. The status values are always indicated by the “State_Write” tag. A further continuing logic in the program of the controller should follow these transitions.

Figure 4-2 Status transitions “Data acquisition and storage”



Copyright © Siemens AG 2014 All rights reserved

Description of the “Download” function recipe

For the further processing, the control program needs a production data set (typically from a SCADA system or MES). The OPC client can access this data and make it available for the control program. To guarantee the reliability, it must be ensured that the control program only continues when the OPC client has consistently read out and saved the complete set. This means that the OPC client and the control program must by synchronized via a status tag. This “State_Read” status tag is monitored by the OPC client throughout the entire runtime.

The table below gives an overview of all the tags used for the recipe download.

Table 4-3

Symbolic name	Description
"Req_from_DBS".State_Read	Status tag for the synchronization of the OPC client and the control program. The following status values are defined: • Idle = 0 • TransferRecipe = 1 • TransferReady = 2 • Error = 3
"Req_from_DBS".Recipe_No	Identifier for the data set which the controller needs. Data type: Integer
"Req_from_DBS".Value_1	Example data for writing on the controller. Data type: String
"Req_from_DBS".Value_2	Example data for writing on the controller. Data type: Float.
"Req_from_DBS".Value_3	Example data for writing on the controller. Data type: Word
"Req_from_DBS".Value_4	Example data for writing on the controller. Data type: DateTime:
"Req_from_DBS".Status	Tag indicating the status of the last database operation. The OPC client can use it to signal to the controller that an error occurred while reading the requested data. In this application example, the OPC client does not read from a real database, but uses the values entered on the surface.

Copyright © Siemens AG 2014 All rights reserved

Process of a recipe request

The figure below shows the data flow and the process of a recipe request, as it has been implemented in the example. Every step is then described in the subsequent table.

Figure 4-3

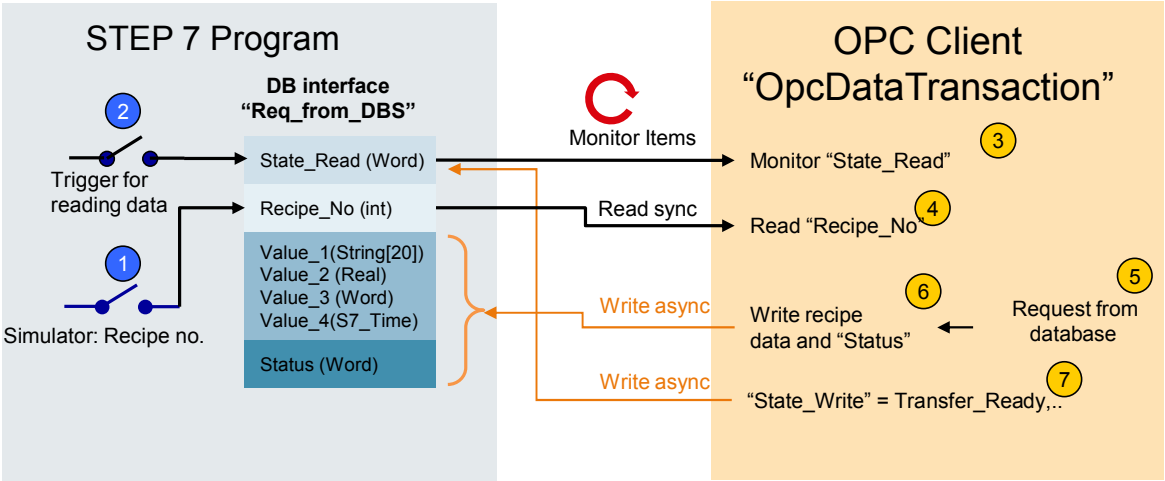


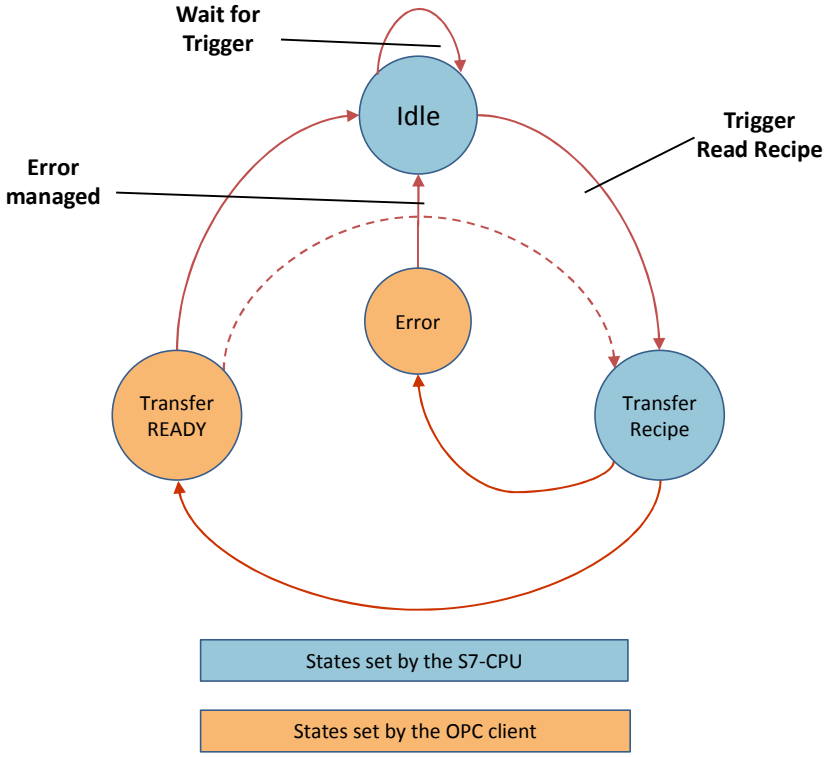
Table 4-4 Logic steps of a recipe request

No.	Description
1	With the tag table, the user writes a recipe number to the **Recipe_No.** tag This number represents the data set the requested by the control program.
2	With the tag table, the user writes the status "TransferRecipe" (1) to the status tag **State_Read** in data block **Req_from_DBS**.
3	The OPC client detects the changes of the **State_Read** tag and shows the new status "TransferRecipe". The client now knows that the controller has requested a recipe data set and is waiting for the transfer.
4	The OPC client reads the requested recipe number from the **Recipe_No** tag.
5	The OPC client reads the requested data set from the "database". For reasons of simplification, it is not actually read from a database here. However, the OPC client instead uses values from the respective controls in the user interface.
6	The OPC client writes the dataset to the tags in the data block **Req_from_DBS** and updates the status word **Status** (Ok, ReadError, WriteError, DBError). If reading the data from the database was not successful, the data set will not be written but the status word updated.
7	The OPC client changes the status of the State_Read tag to "TransferReady" (2). Now, the controller knows that the OPC client has completed the operation. The controller has received the requested recipe data and can continue processing the order and reset the status to "Idle" (0), for example, or trigger another transfer.

Copyright © Siemens AG 2014 All rights reserved

The figure below shows the status transitions in the “data transfer and storage” mode. The status values are always indicated by the “State_Write” tag. A further continuing logic in the program of the controller must follow these transitions.

Figure 4-4 Status transitions “Recipe Download”



Copyright © Siemens AG 2014 All rights reserved

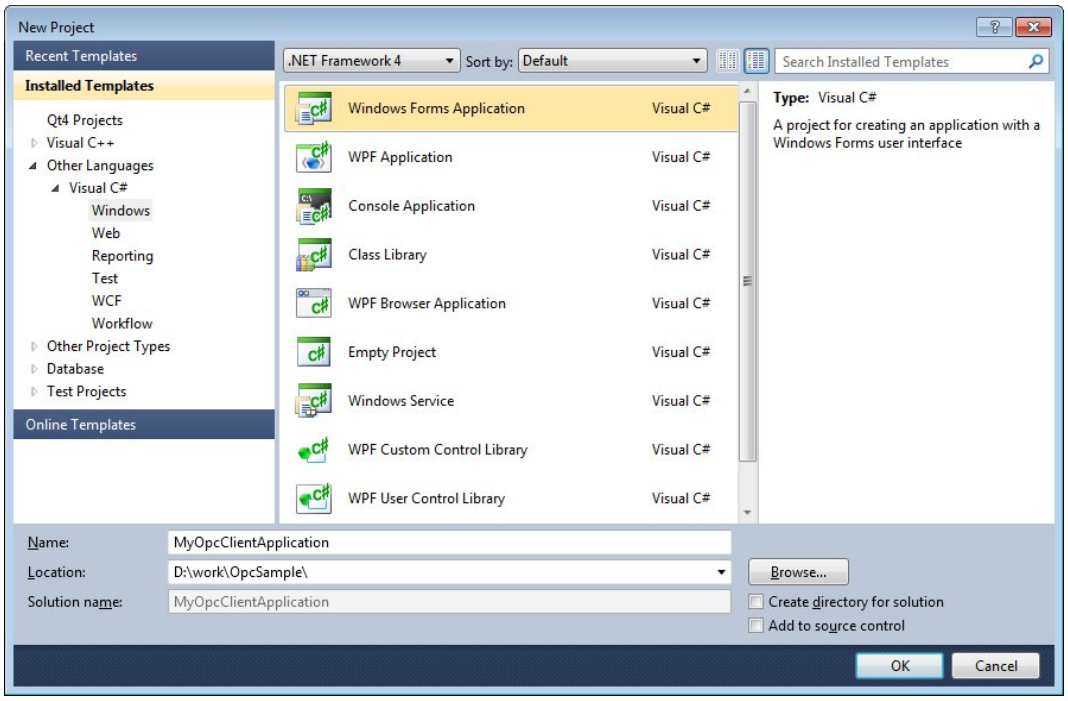
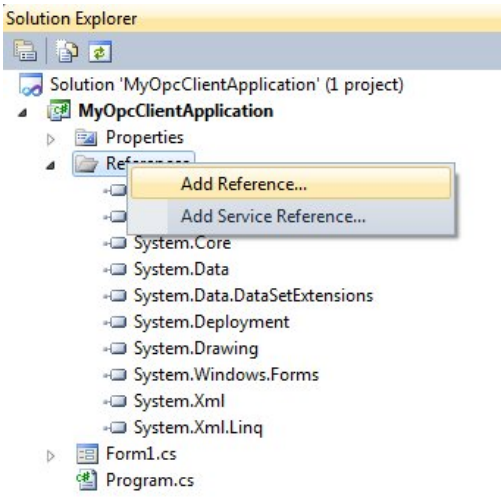
4.2 Programming of the client whilst using the .NET OPC Client API

4.2.1 Integrating the assemblies

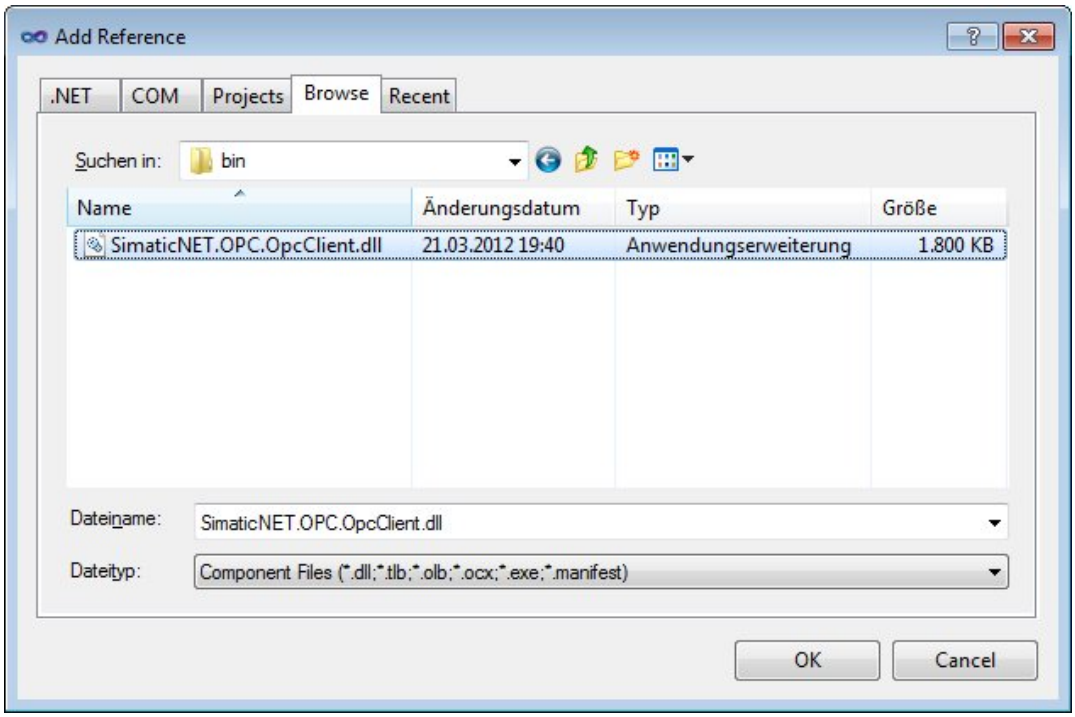
General information

In order to use the functionality of the .NET OPC Client API , the assemblies first have to be referenced accordingly in MS Visual Studio.

Table 4-5

No.	Action
1.	Open MSVS 2010 and create a new project, for example, for a Windows Forms application. 
2.	For adding the references in the project settings, first open the “Add Reference” dialog with the context menu. 

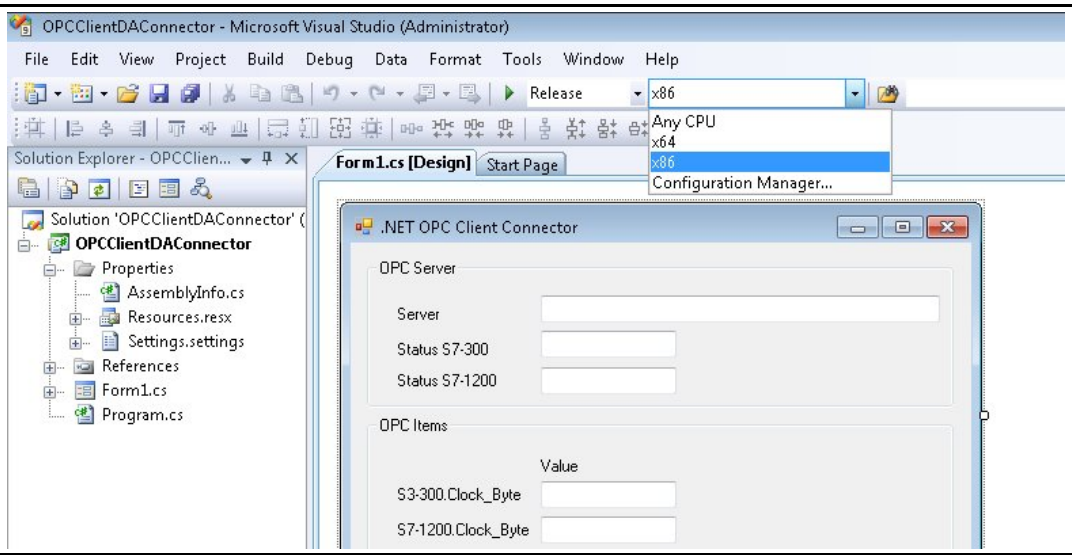
Copyright © Siemens AG 2014 All rights reserved

No.	Action
3.	Select the **SimaticNET.OPC.OpcClient** in the dialog and add it. 

4.2.2 Setting the compiler

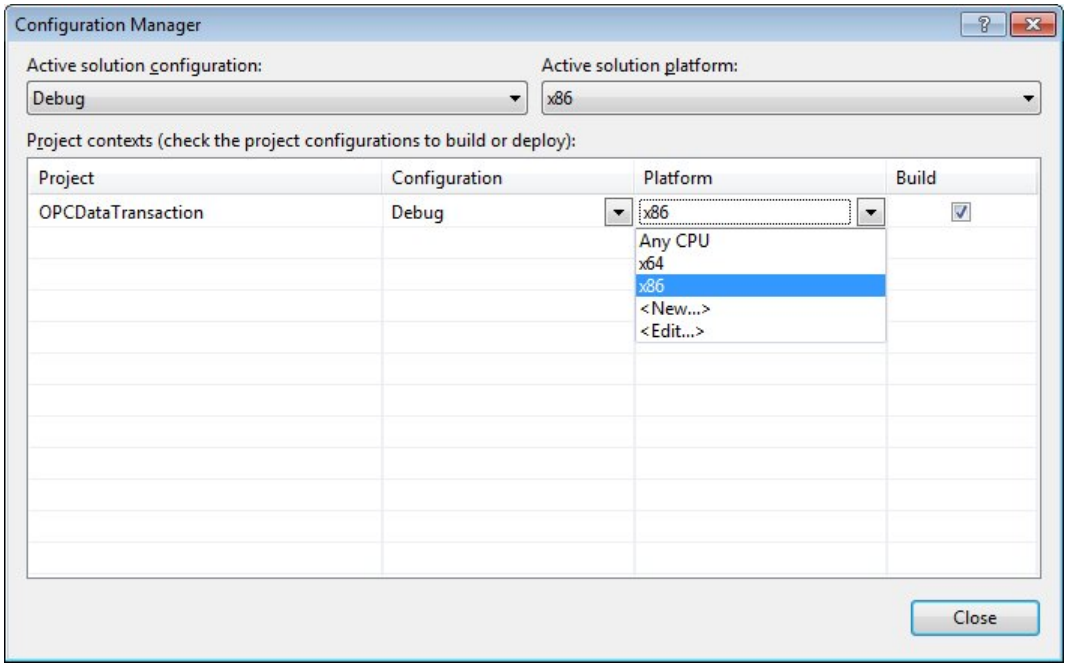
The fully compiled assemblies have been created for x89 systems. This means that the sample application has to be constructed for x86. For this purpose, the Build configuration in MS Visual Studio for debug and release has to be set to **x86**.

Table 4-6

No.	Action
1.	For **Release** and **Debug** you set the Build configuration to **x86** each. 

Copyright © Siemens AG 2014 All rights reserved

No.	Action
2.	If x86 cannot be selected, open the Configuration Manager and add **the x86 Build option**.



4.2.3 Sequence of the example program

The following chapter explains the general structure of the OPC client program. For details, please refer to the well-documented source code of the example program.

Class

The example program has been implemented in the following classes.

Table 4-7

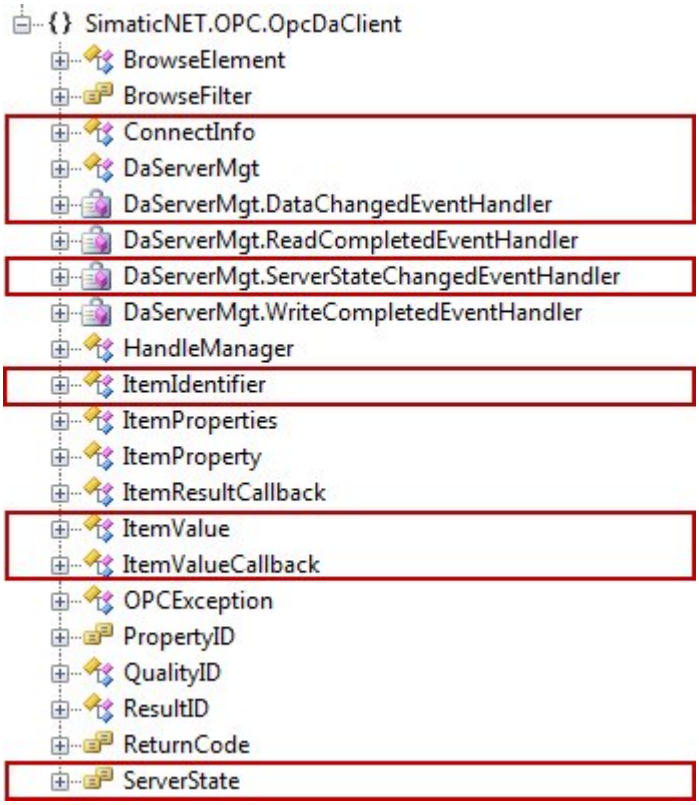
Name of the class	Description
Program	Starting and initializing the program. The main class DataTransactionGui is instanced here.
DataTransactionGui	This class contains the core functionality of the whole program. • Design of the user interface • Instancing of OPCDaClient objects from the API • Creating the threads for reading/saving data and recipe download • Management of the status machines for reading/saving data and recipe download
DBLogger	This class encapsulates all the functions concerning the writing of the data sets read from the controller into a database. The database access functions run on a separate thread.
DataSetRecord	This class contains the image of a data set from the controller.

Copyright © Siemens AG 2014 All rights reserved

SimaticNET.OPC.OpcDaClient objects used

The figure below shows the available objects in the namespace SimaticNET.OPC.OpcDaClient of the API (see also chapter 3.3) and which objects were used in this example.

Figure 4-5



Copyright © Siemens AG 2014 All rights reserved

4.3 Explanations on the simulation program in the controller

4.3.1 Process of simulation

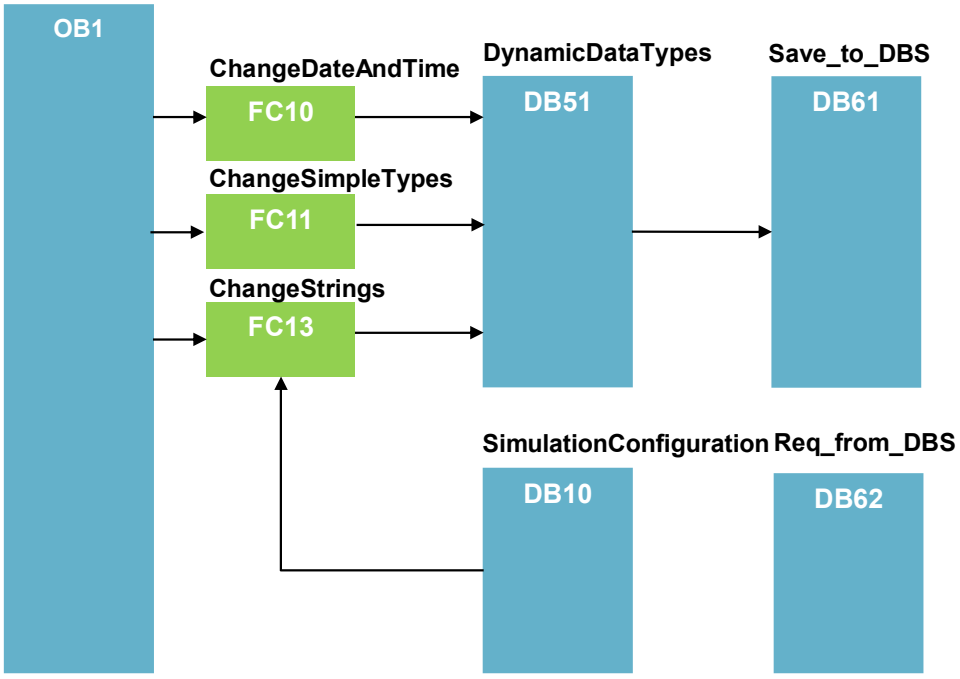
The processes within the controller are quite clear.

The OB1 block calls the functions FC10, FC11 and FC13 cyclically. These functions

- “move” the data in data block DynamicDataTypes (DB51).
- generate values which for the “Data acquisition” scenario is written into a database by the OPC Client.

If the “SimulateNewRecordSet” tag changes from false to true, the respective values in the next OB1 cycle will be copied from DB51 to DB61.

Figure 4-6



Copyright © Siemens AG 2014 All rights reserved

4.3.2 Function of the blocks

The process data is simulated using the blocks from the control program. This is achieved by means of automatic incrementation of different data types.

Table 4-8

Block	Description
OB1 Main	Main loop of the program, calls all subprograms
FC10 ChangeDateAndTime	Changes date and time in a tag in DB51
FC11 ChangeSimpleTypes	Increments simple data types in DB51
FC13 ChangeString	Changes a string tag in the DB 51 in terms of content and its length
DB51 DynamicDataTypes	Global data block with different simple data types that are moved by the simulation.
DB61 StaticDataTypes	Contains data that are used for the example “data acquisition”.
DB62 Req_from_DBS	Contains the data sent to the controller by the OPC client.
DB10 SimulationConfiguration	Contains parameters for the simulation limits of the individual tag types.

Copyright © Siemens AG 2014 All rights reserved

5 Configuration and Settings

5.1 Configuration of the PC station

Overview of IP addresses

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

Table 5-1

Focus	Module	IP address
PG/PC	NDIS network card	192.168.172.100
Control	CPU 315-2 PN/DP	192.168.172.2
Control	CPU 1516-3 PN/DP	192.168.172.1

Subnet mask: 255.255.255.0.

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 If you want to use the project provided with the delivery, it is important that the network card has the fixed IP address 192.168.172.100. It can be set with the network settings and the TCP/IP properties.

General preparations

To work with the supplied application software, please de-archive the enclosed code folder in accordance with the instructions in chapter 6.2.1.

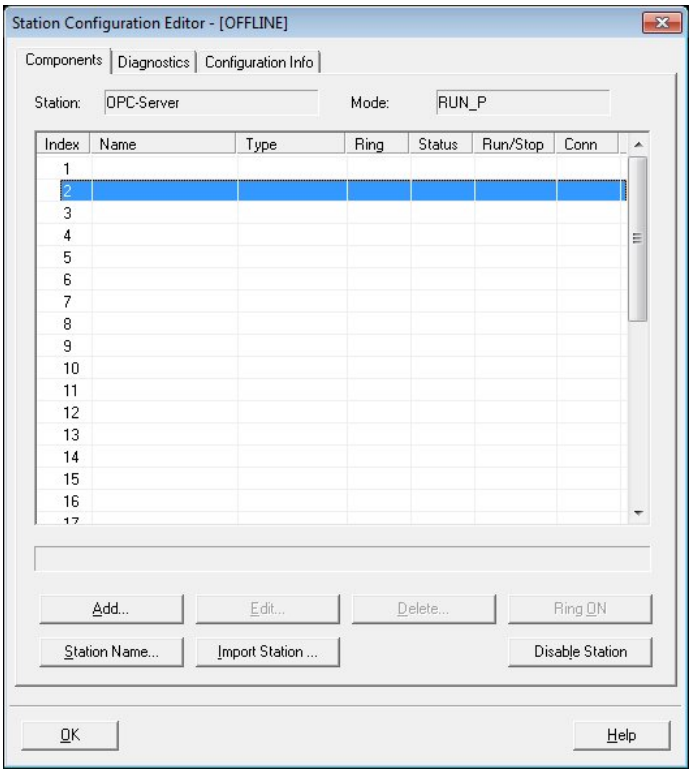
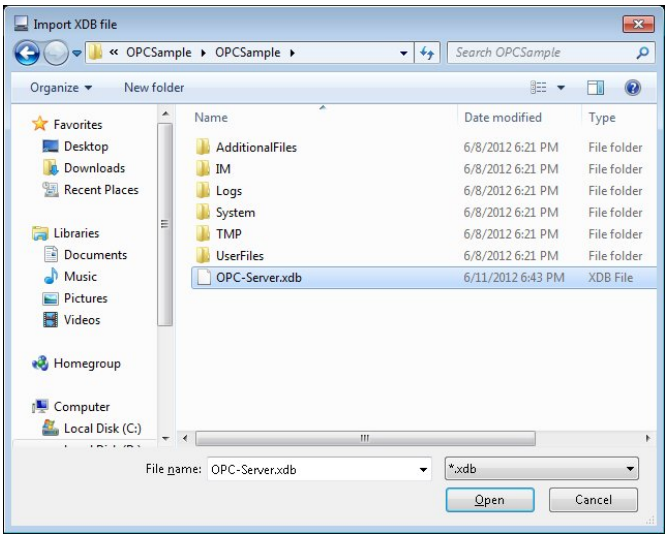
Copyright © Siemens AG 2014 All rights reserved

5.1.1 Importing the XDB file to the Components Configurator

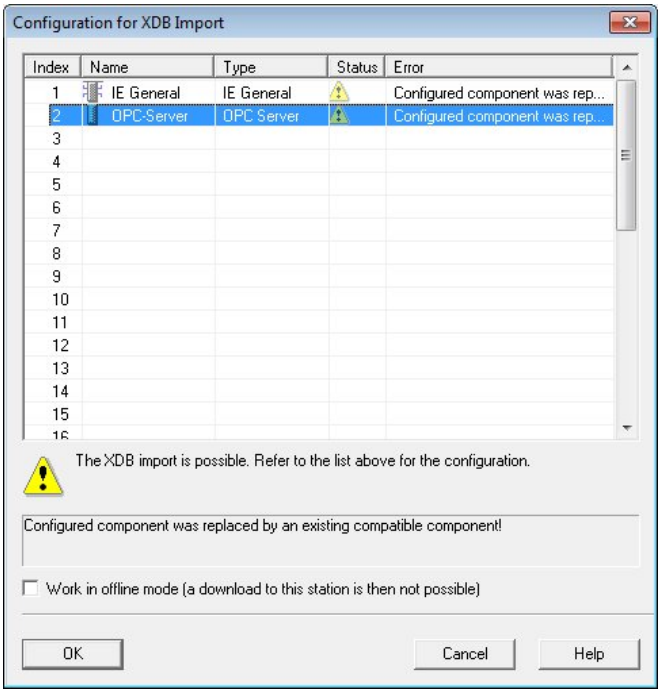
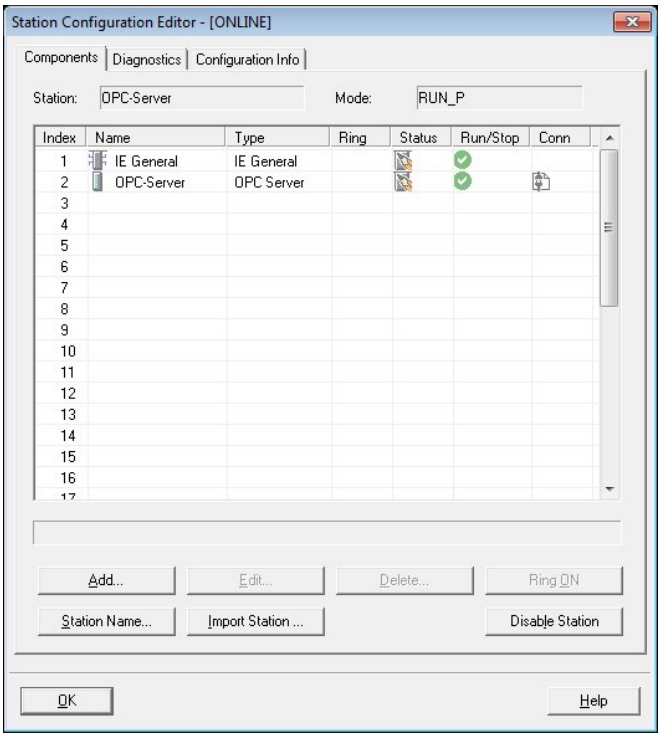
The XDB file “OPC-Server.xdb” already exists in the TIA project included.

If, for example, you wish to use another station name or different interface settings (see chapter 5.1.2) you need to regenerate the XDB file after the configuration has been modified.

Table 5-2

No.	Action	Remarks
1.	Open the Component Configurator by double-clicking on the icon in the task bar.	
2.	Click on the “Import Station...” button.	Confirm the query with Yes.
3.	Browse to the project folder of your TIA project. Select the XDB file. Press the “Open” button.	

Copyright © Siemens AG 2014 All rights reserved

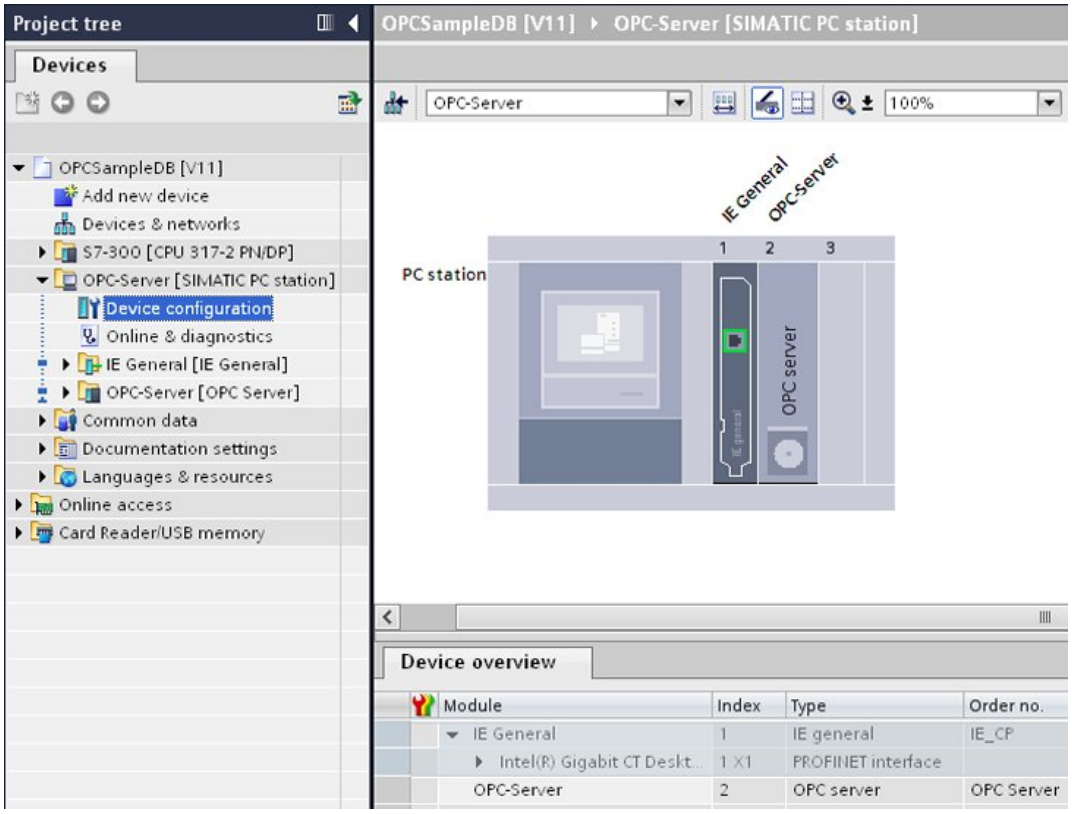
No.	Action	Remarks
4.	The import wizard confirms that import is possible. If components have been configured in a different version, they will be exchanged by existing compatible versions Click on “OK”.	
5.	After the successful import of the XDB file, your PC station is **ONLINE**.	

Copyright © Siemens AG 2014 All rights reserved

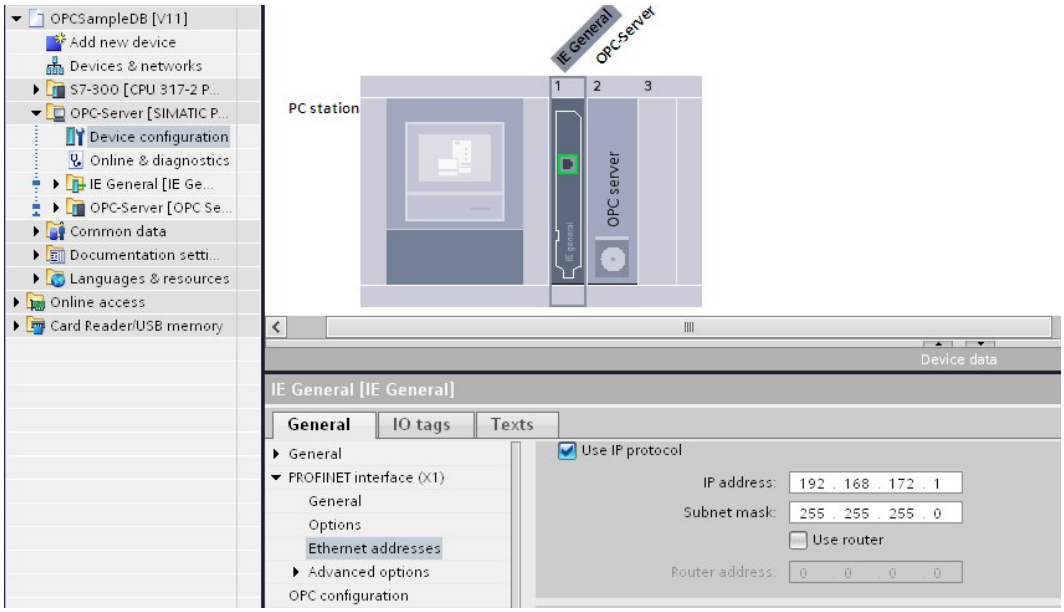
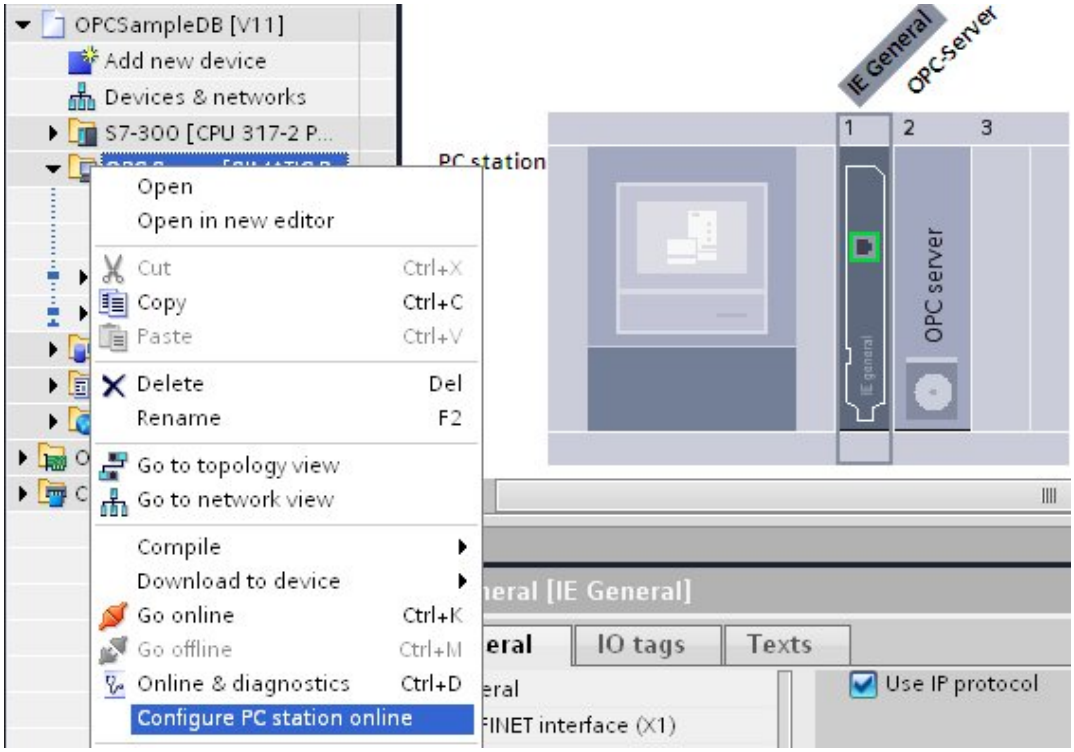
5.1.2 Changing the IP address of the PC station in STEP 7 V1x

Open the STEP 7 V1x project as described in chapter 6.3.

Table 5-3

No.	Action
1.	Select the SIMATIC PC station in the project view and open the **Device configuration**. 

Copyright © Siemens AG 2014 All rights reserved

No.	Action
2.	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. **Note:** In this case an IE General network card was used. 
3.	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. 

Copyright © Siemens AG 2014 All rights reserved

5.2 Configuring an OPC server

The main focus of this application is the usage of OPC. A number of configurations are necessary, which can be carried out in various ways.
In this example, the configuration in STEP 7 V11 is shown.

Note

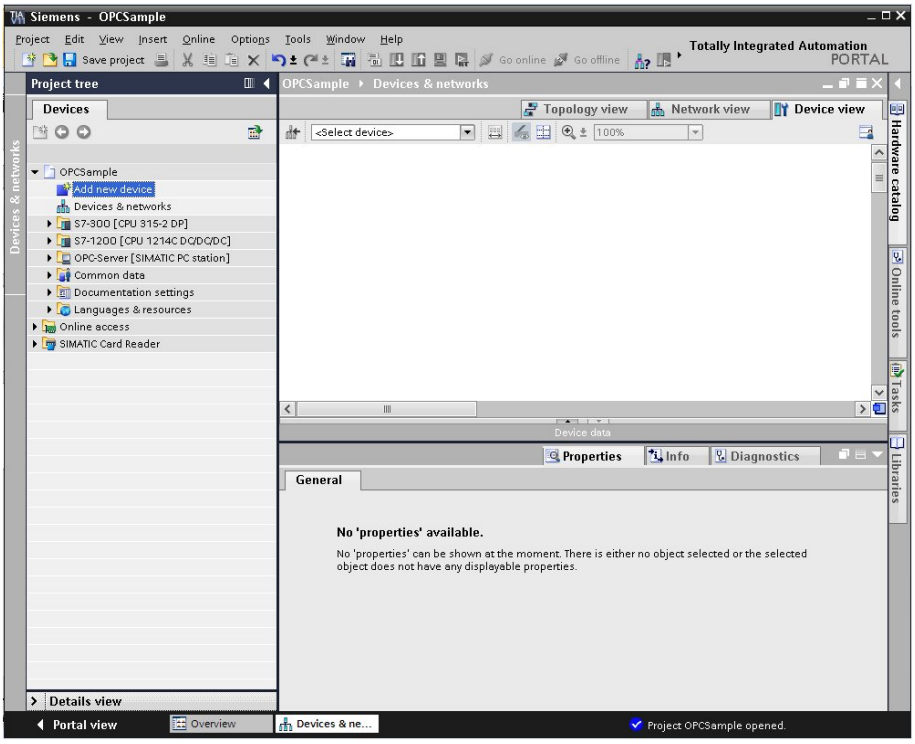
This chapter is only relevant if you are interested in details on the configuration.

In the enclosed STEP 7 project, the complete configuration has already been performed.

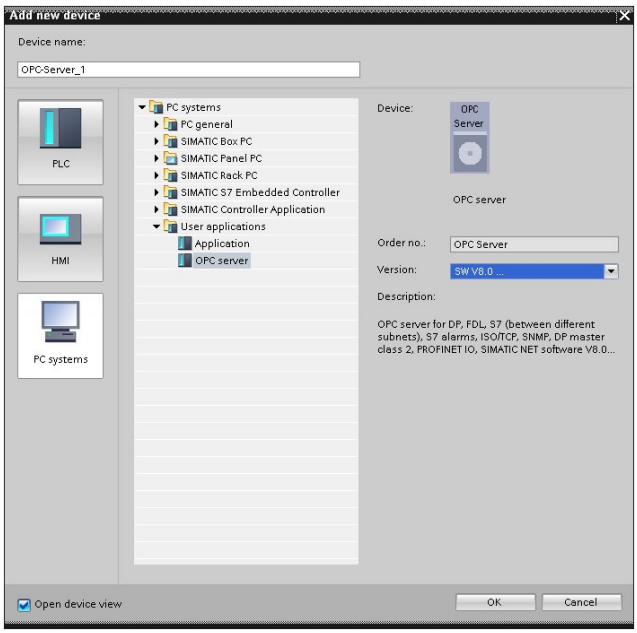
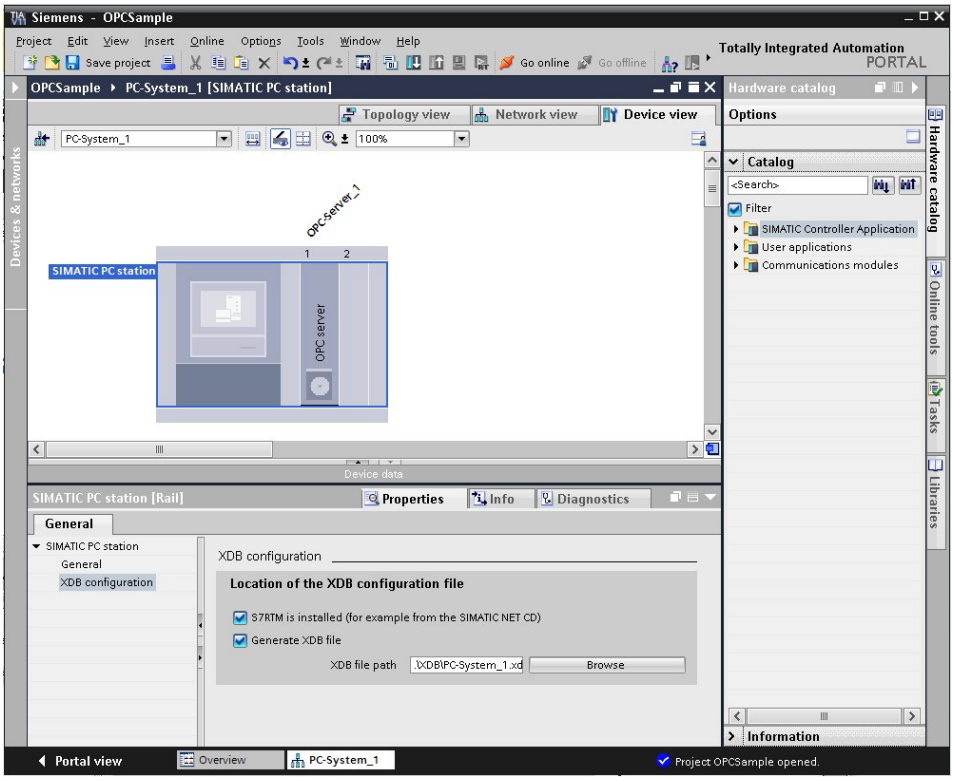
5.2.1 Configuring the OPC server in STEP 7 V1x

Execute the following steps, in order to add an OPC server to an existing STEP 7 V1x project with already configured SIMATIC station.

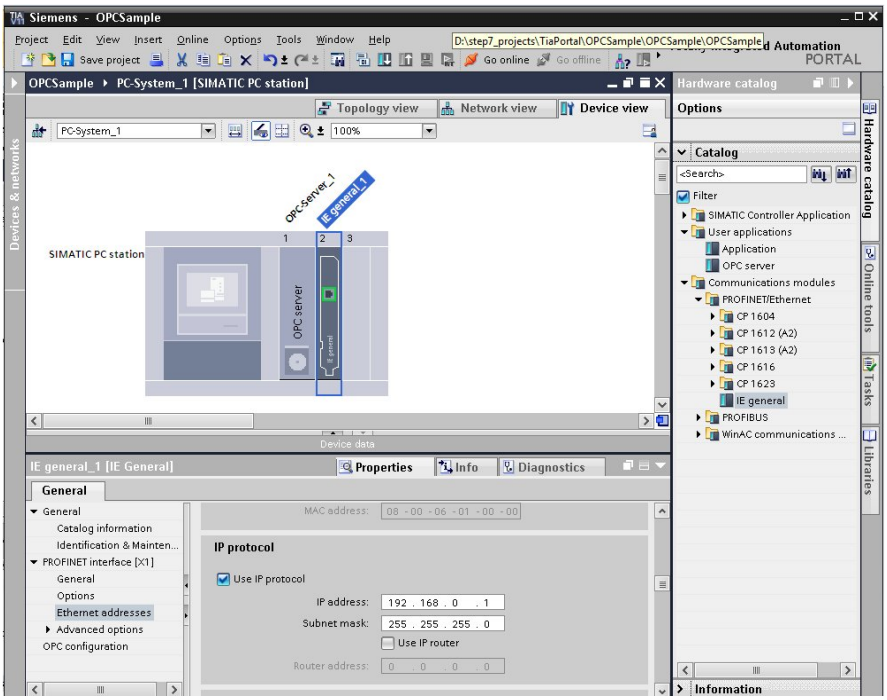
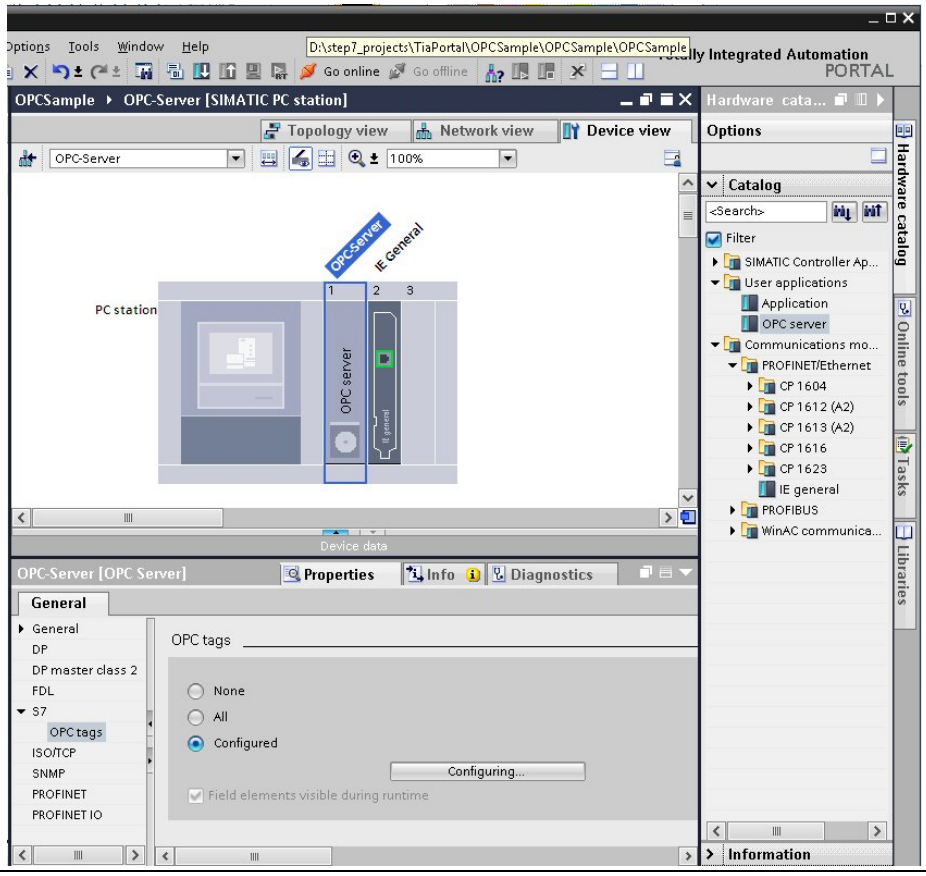
Table 5-4

No.	Action
1.	Open your STEP 7 V1x project and go to the project view.
2.	Double-click on **Add new device**. 

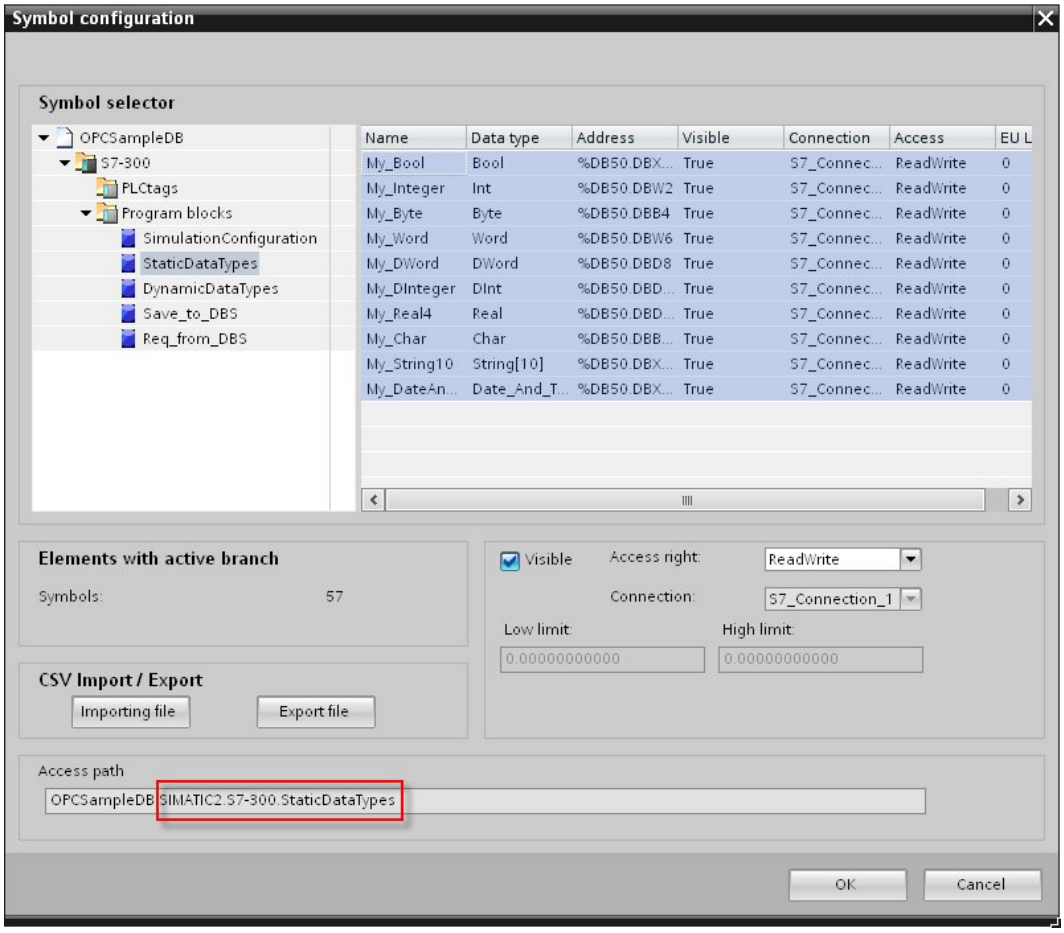
Copyright © Siemens AG 2014 All rights reserved

No.	Action	
3.	Select **PC systems**, and the **OPC server** in **User applications**.	In this application, SIMATIC NET OPC Server V8.0 (or higher) was used. Note: If possible, the version of the SIMATIC NET OPC server should correspond to the actually installed version. 
4.	A SIMATIC PC station with a SIMATIC NET OPC server is created on Index 1. Enable the **Generate XDB file**. 	

Copyright © Siemens AG 2014 All rights reserved

No.	Action
5.	Add an **"IE general" network card** and assign the IP address. **Note:** The IP address must match the address of the target system. 
6.	Enable the symbolic display of the OPC tags. When using symbols, you can configure whether you want to use **All** or **Configured** symbols. 

Copyright © Siemens AG 2014 All rights reserved

No.	Action
7.	If you only want to access a selection of process tags symbolically, please select the symbolically addressable tag with the **Configuring**... button, by checking or unchecking “visible” for each tag. The string marked in red under **Access path** is the basic string which must be entered with the symbolic address name in the user interface of the OPC client. (See chapter 7.1) **Note:** If no symbols are displayed, the S7 connection configuration is missing. An S7 connection between the OPC server and the respective S7 station is absolutely necessary. 

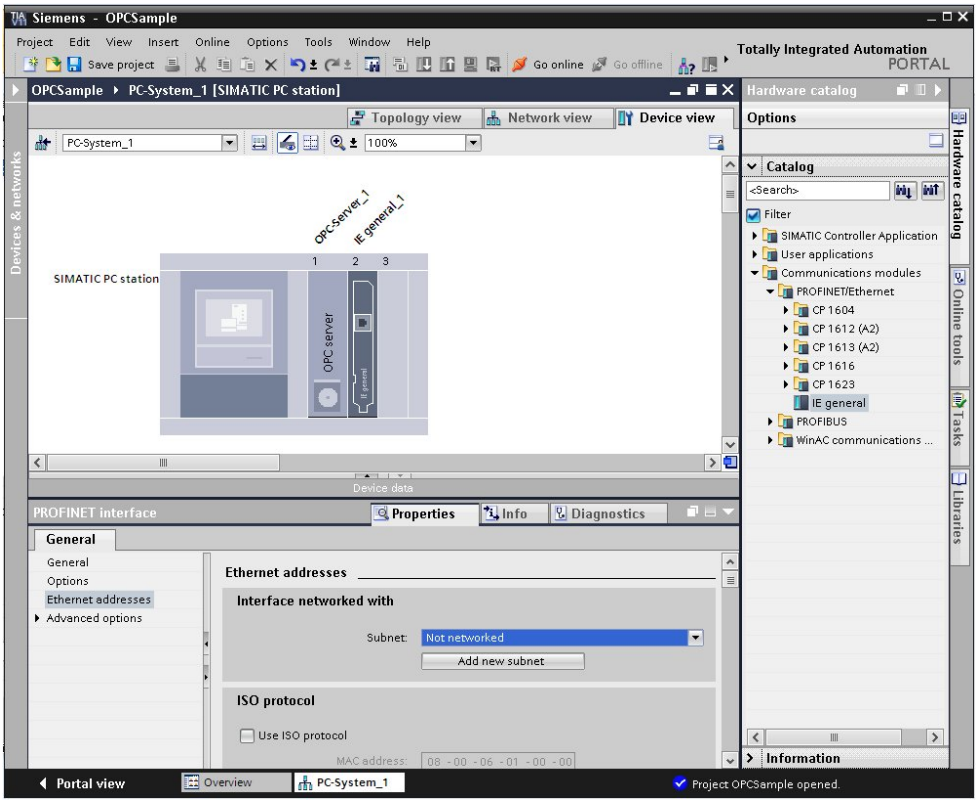
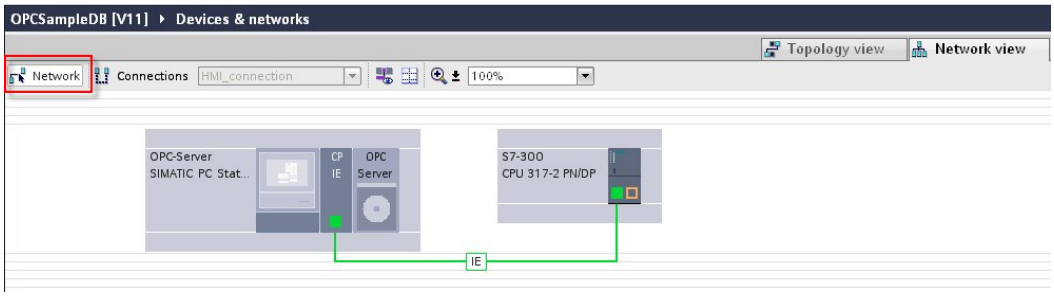
Copyright © Siemens AG 2014 All rights reserved

5.2.2 Configuring the S7 connections

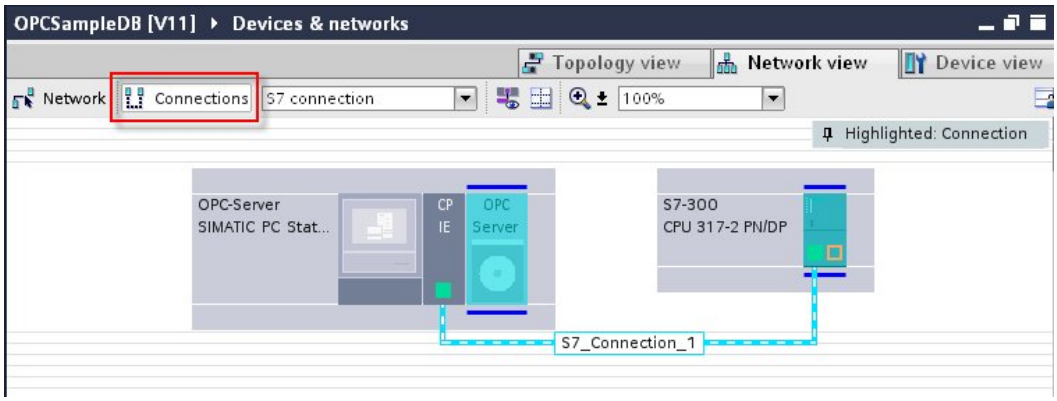
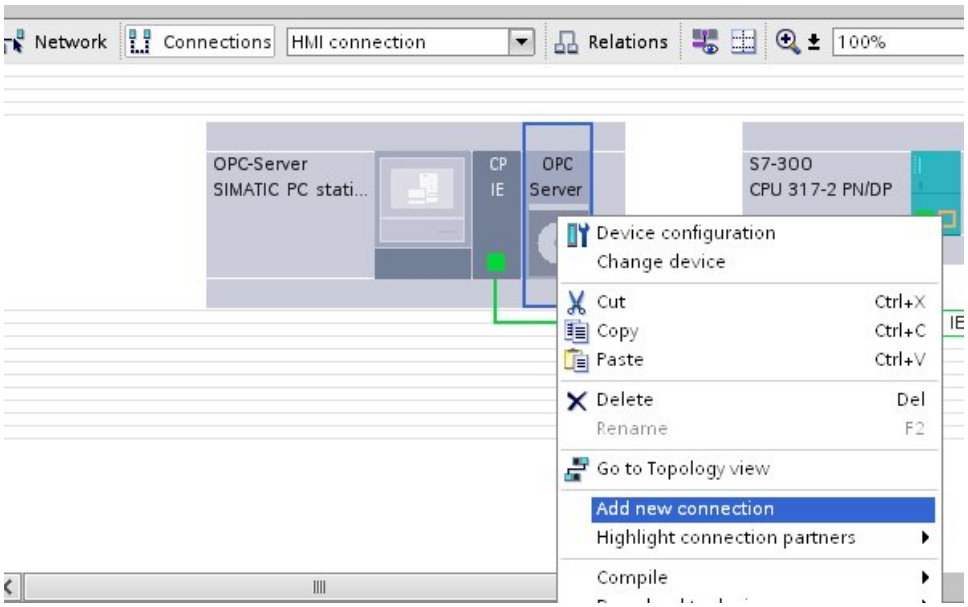
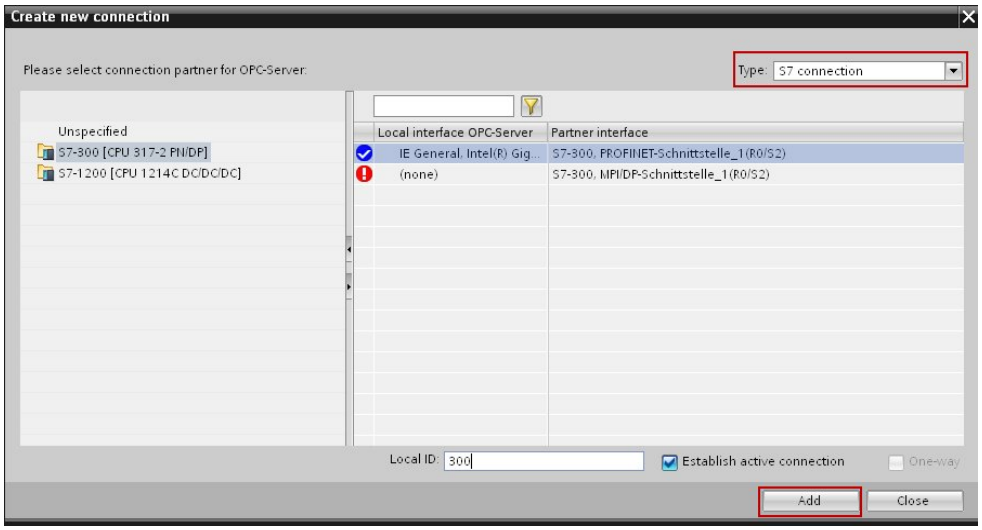
The PC station and the OPC server require configured S7 connections for using the symbolic illustration of the OPC item.

The following table shows the configuration of the S7 connection with the S7-300 as its partner station.

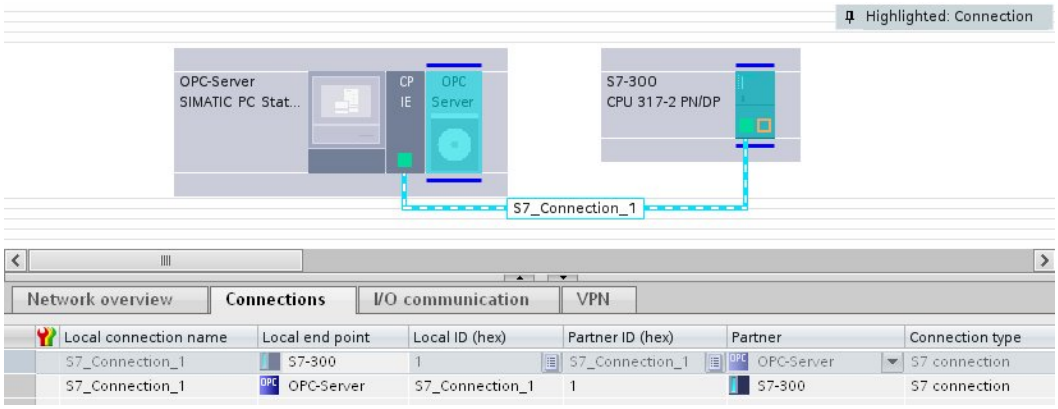
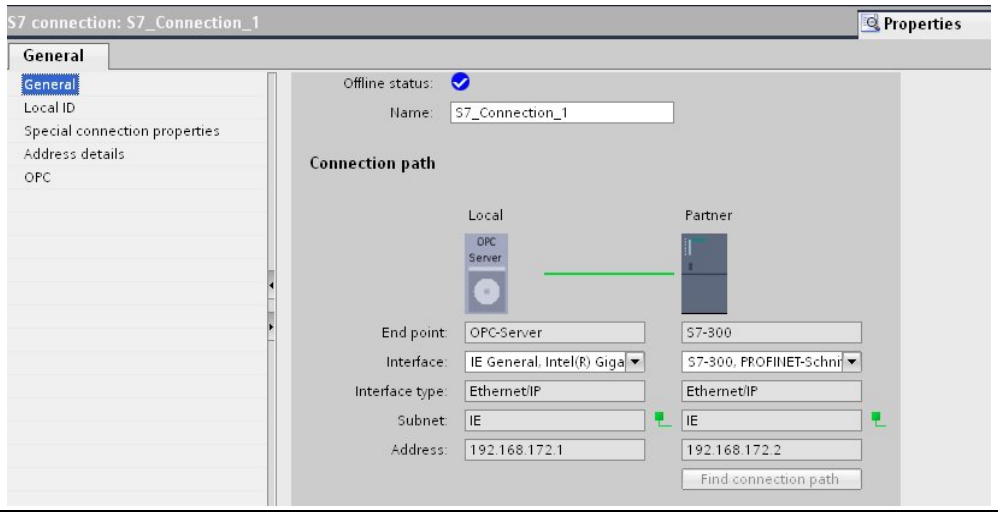
Table 5-5

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:** All network nodes should be located in the same IE network. Go to the network view, network the PC station with the SIMATIC station/s and configure one S7 connection each. 
2.	Open the network view with the **Network** button. Connect all nodes with the same IE network. 

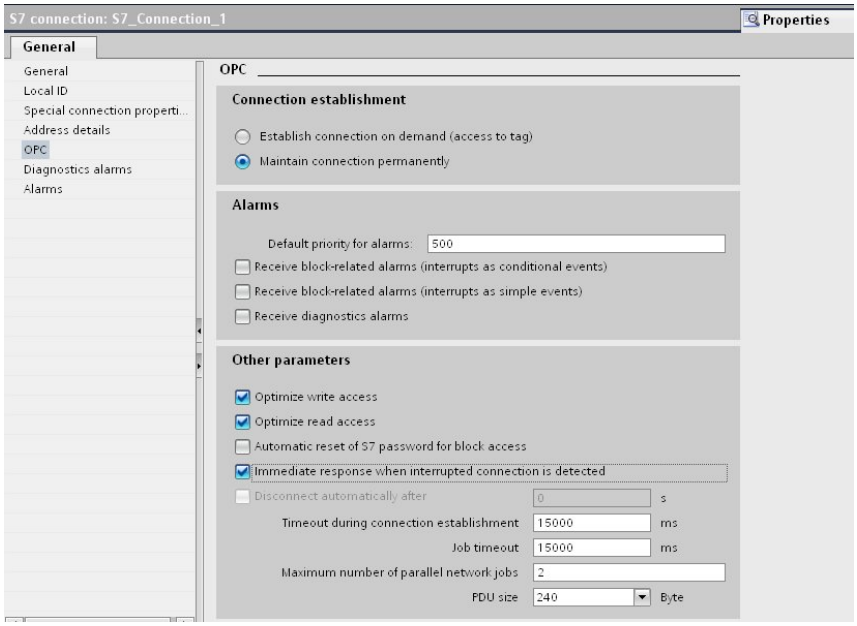
Copyright © Siemens AG 2014 All rights reserved

No.	Action
3.	To create a new connection, change to the **Connection view**. 
4.	Select the OPC server. **Add new connection** in the context menu. 
5.	Select **S7 connection** as the **connection type**, the connection partner (here: S7-300), and the physical access point (blue) supported by both partners. Assign a name for the S7 connection. 

Copyright © Siemens AG 2014 All rights reserved

No.	Action
6.	The connection now appears in the Connections list. 
7.	Configure the connection by editing the properties of the connection. 

Copyright © Siemens AG 2014 All rights reserved

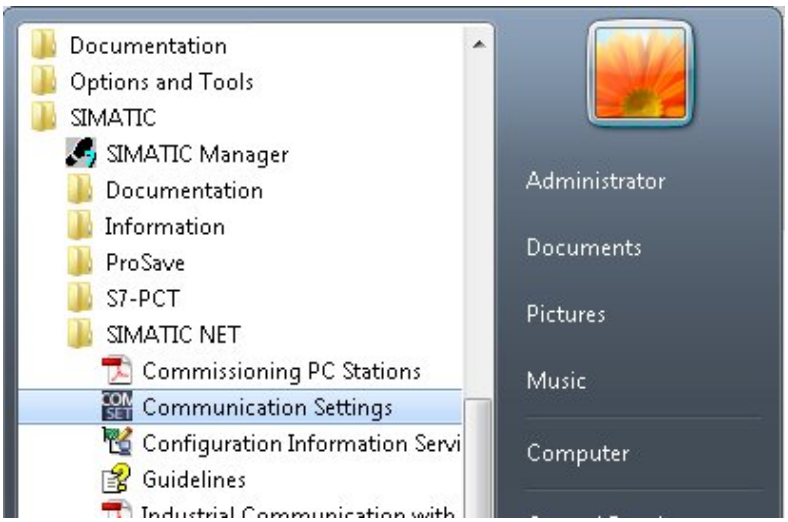
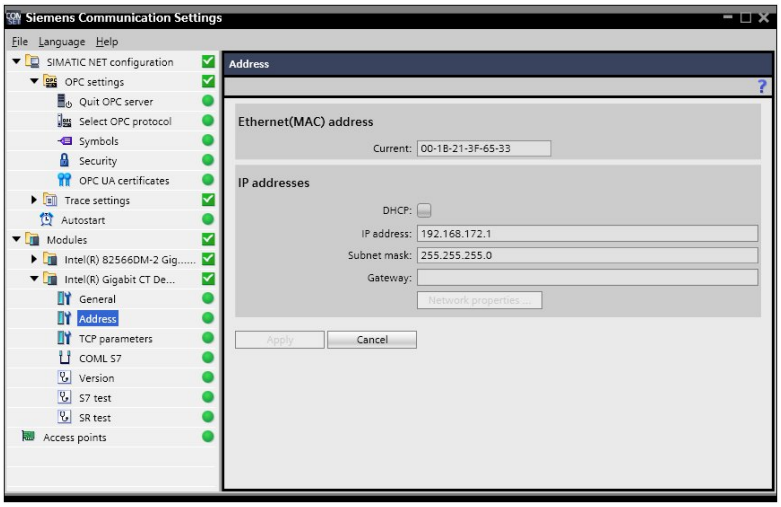
No.	Action
8.	The OPC server should always establish the connection actively and **Maintain connection permanently**. Furthermore, the OPC server should have an **Immediate response when interrupt connection detected**. Activate the respective options for this. 
9.	Save and load the controllers.

Copyright © Siemens AG 2014 All rights reserved

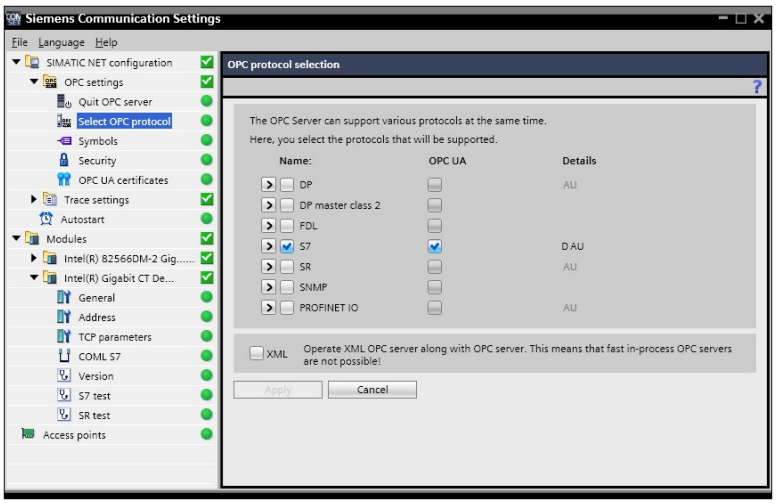
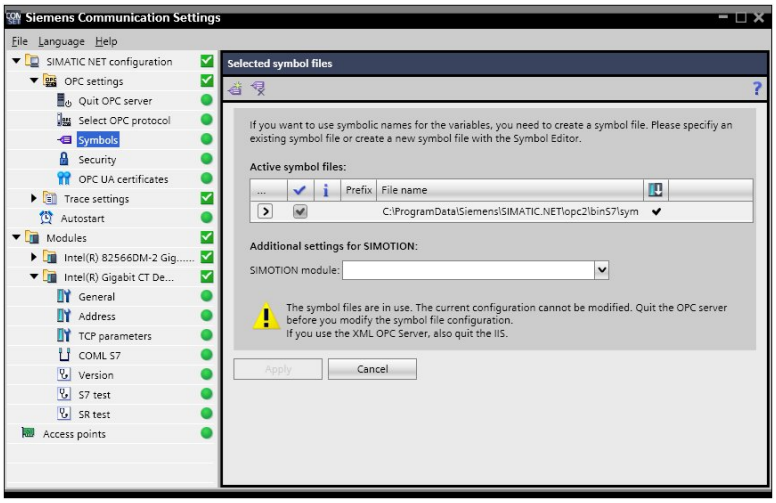
5.2.3 Final check of the settings

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

Table 5-6

No.	Action	Remarks
1.	Open the configuration dialog	
2.	Check the set IP address.	

Copyright © Siemens AG 2014 All rights reserved

No.	Action	Remarks
3.	Check the set protocols. **Note:** enabling the S7 protocol is sufficient for this application.	
4.	Check whether the symbols have been loaded.	
5.	If one of the settings does not correspond with the displayed images, perform the preceding configurations again. Close the configuration dialog box.	

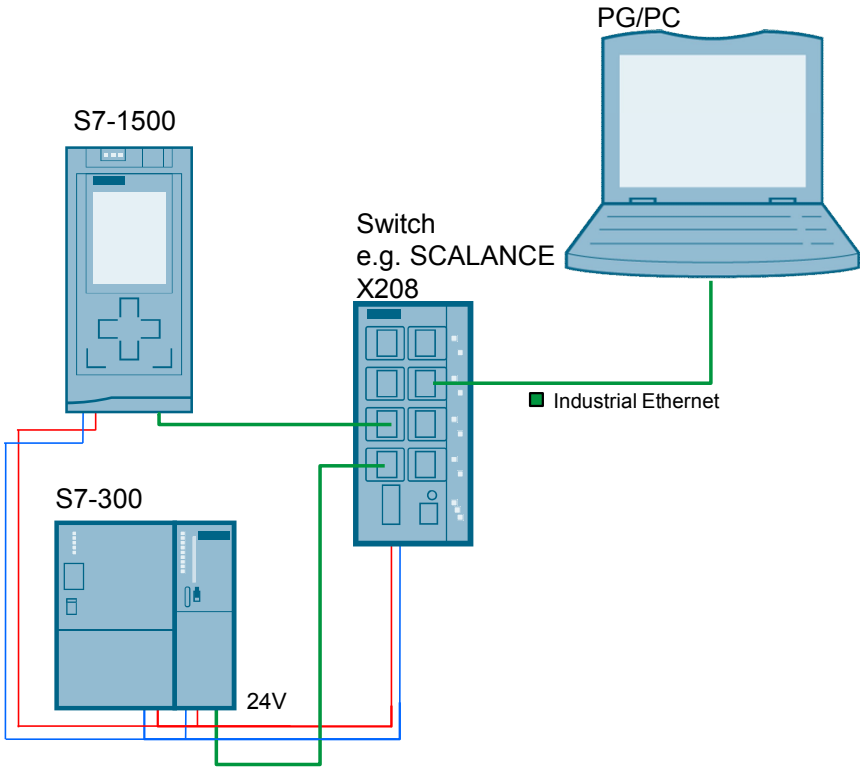
Copyright © Siemens AG 2014 All rights reserved

6 Installation and Commissioning

6.1 Hardware installation

The figure below shows the hardware setup of the application.

Figure 6-1



Connect the S7-300, the S7-1500 and the SCALANCE X208, each to a respective 24V power supply. Connect all devices with the SCALANCE X208 by means of a standard Ethernet cable.

Note In order to demonstrate the functionality, only one SIMATIC S7 station is necessary.

For the exact hardware components, please refer to chapter 2.3.

Note The installation guidelines for these components must always be observed.

Copyright © Siemens AG 2014 All rights reserved

6.2 Software installation

6.2.1 De-archiving the application software

Unzip the zipped **21447513_Csharp_OpcDataTransaction_CODE.zip** code folder from the download area into a folder your choice.

This folder contains:

- the sub-folder **OpcData Transaction** with the necessary data for the OPC application.
- the sub-folder **TIA_PROJECT** with STEP7 projects for STEP7 V11, STEP7 V12 and STEP7 V13. The projects contain the simulation program for the controllers.

6.2.2 Installing of the standard software

STEP7 V1x and SIMATIC NET must be installed on the PG/PC. STEP 7 V1x has already been preinstalled on SIMATIC PGs.

A description of the installation procedure for STEP 7 V1x 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.

NOTICE	**If you are using an older operating system than Windows 7 SP1 and if SIMATIC NET PC software V8.x was not installed on it, you have to install .NET-Framework 3.5 +SP1 first.** Information on this matter can be found on the Microsoft Internet pages http://www.microsoft.com/

6.2.3 Installing the OPC client on the PC/PG

In the **\OpcDataTransaction\bin** directory you will find this file: **OpcDataTransaction.exe**.

Please execute .exe for the installation of the user surface.

Note	EXE can only be executed if the respective assemblies are located in the same directory

Copyright © Siemens AG 2014 All rights reserved

Files included

The **OpcDataTransaction** file contains two sub-folders with the following contents:

- bin
  - Folder DB: contains a Microsoft Access database
  - OpcDataTransaction.exe. The OPC client application described here
  - SimaticNET.OPC.OpcClient.dll client API
- src
  - the MS Visual Studio Solution file
  - the source code

Deinstallation

To uninstall, delete the directory.

Copyright © Siemens AG 2014 All rights reserved

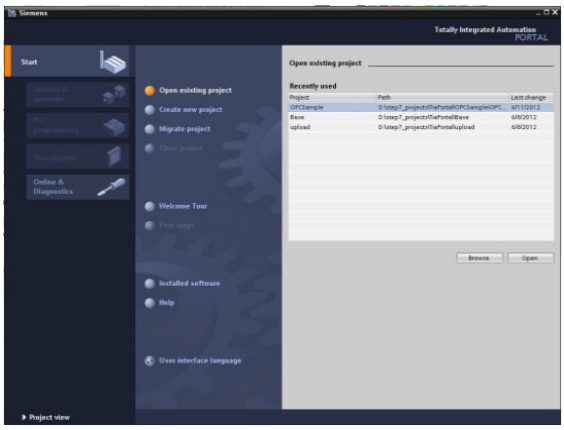
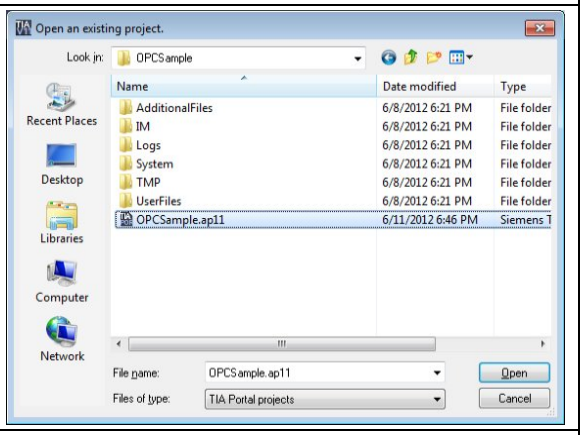
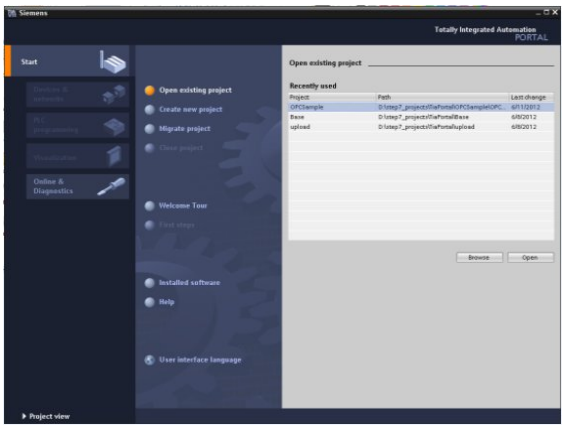
6.3 Commissioning the application

General preparations

The TIA V1x projects are located in directory **21447513_Csharp_OpcDataTransaction_CODE\TIA_PROJECT\OPCSampleDB_V1x.**

Note The following instruction is based on the TIA V11 project. The commissioning process with the TIA V12 or TIA V13 project is performed analog to this.

Table 6-1

No.	Action	Remarks
1.	Open the TIA portal and navigate to the respective directory with the browser function.	
2.	Select the TIA V11 project **OPCSampleDB.ap11** and confirm with **Open**.	
3.	Go to **Project view**.	

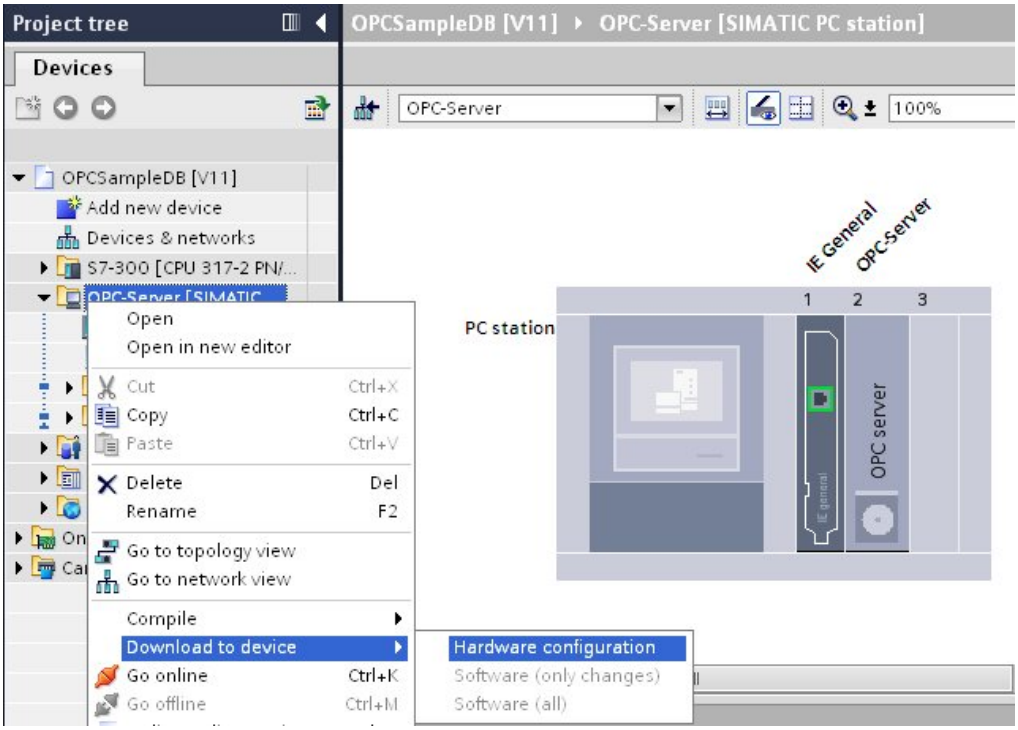
Copyright © Siemens AG 2014 All rights reserved

Loading the STEP 7 V1x project on the PC station

A precondition for loading the PC station is the opened STEP 7 V1x project.

Note Make sure that the network card of the PC is set to IP address 192.168.172.100 (see chapter 5.1).

Table 6-2

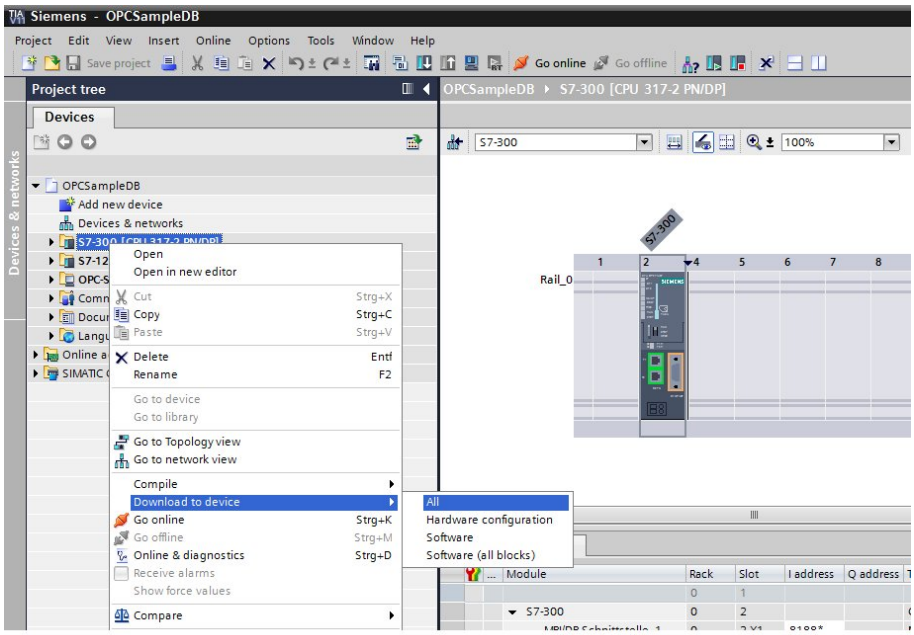
No.	Action
1.	Select the station and open it with the context menu. Alternatively: Import the XDB file (see chapter 0). 

Copyright © Siemens AG 2014 All rights reserved

Loading the simulation to the S7 stations

A precondition for loading the S7 stations is the opened STEP 7 V1x project.

Table 6-3

No.	Action
1.	Select the S7-300 station. Compile and load it with the context menu. 

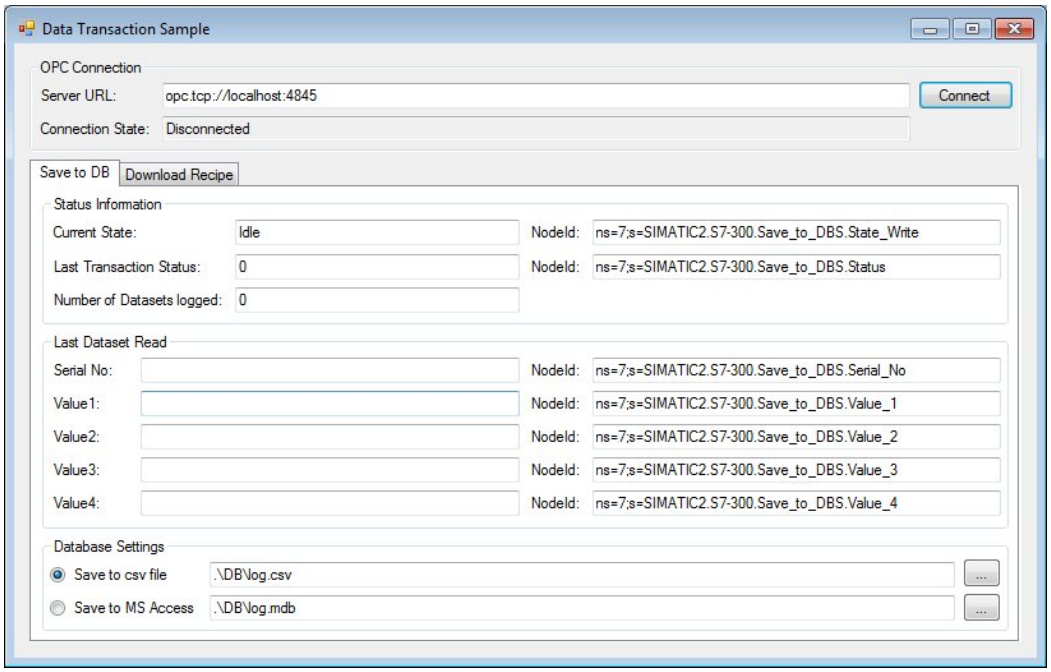
Copyright © Siemens AG 2014 All rights reserved

7 Operating the Application

In this chapter, we first explain the general command steps and then the individual steps for the scenarios “Data acquisition” and “Download Recipe”.

7.1 General operational steps at the user interface

Table 7-1

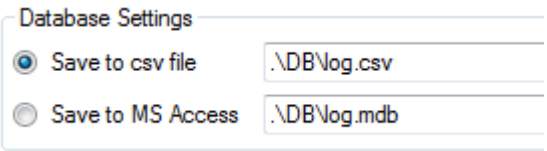
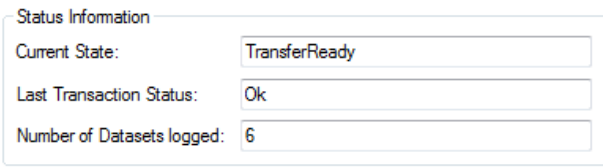
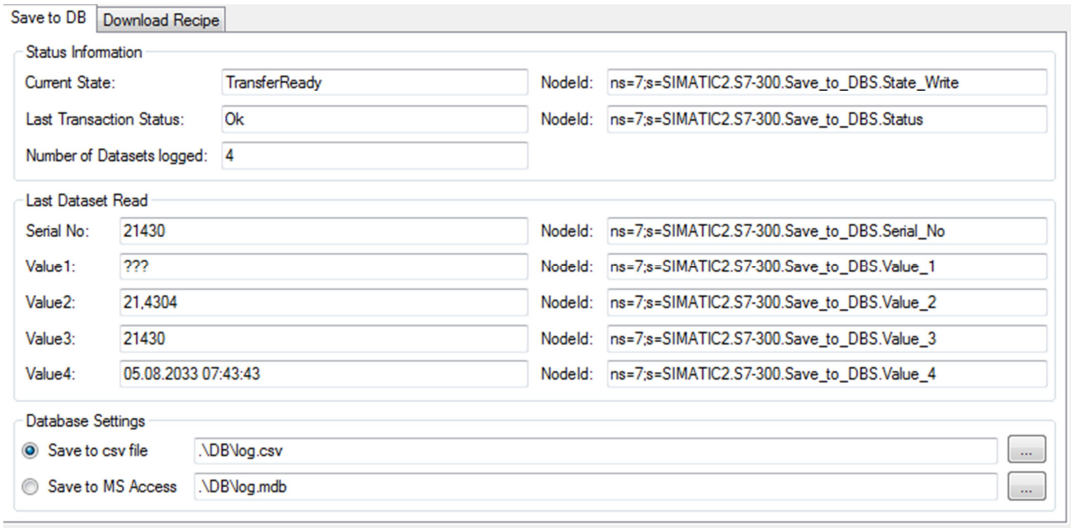
No.	Action	
2.	Start the user interface.  	
3.	Edit the **Server URL** and connect to the server with the **Connect** button.	The server URL decides whether the OPC client connects via the **COM DA** or the **OPC UA**. URL for COM DA: opcda://<computername or IP>/OPC.SimaticNET URL for OPC UA: opc.tcp://<computername or IP>:4845
4.	Select the desired scenario via the respective tab:	For the “Save to DB” tab, follow chapter 7.2. For the “Downlaod Recipe” tab, follow chapter 7.3.

Note When using the example project with the S7-1500 CPU, you need to replace the ‘SIMATIC2.S7-300’ string in the NodeIDs with the ‘SIMATIC1500.S7-1500’ string!

Copyright © Siemens AG 2014 All rights reserved

7.2 Operation of the “Data acquisition” scenario

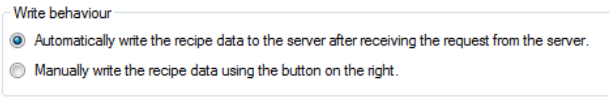
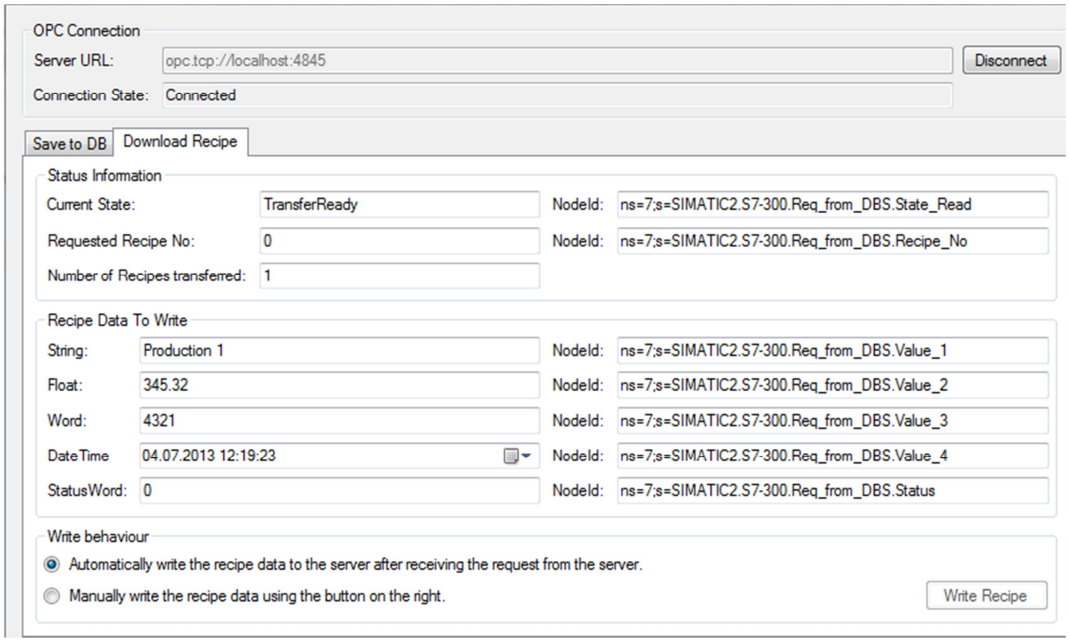
Table 7-2

No.	Action	Remarks
1.	Setting the NodeID	Depending on whether the connection is made via OPC CPM DA or OPC UA, the NodeIDs are different. With UA, the NamespaceIndex is put in front of the string identifier. In the example configuration the nodes used are all in Namespace 7. Example: NodeID for State_Write: **COM: SIMATIC2.S7-300.Save_to_DBS.State_Write** **UA: ns=7;s=SIMATIC2.S7-300.Save_to_DBS.State_Write**
2.	Entering new values in the controller.	In the **Save_to_DBS** tag table you change the values of the tag Value1 to tag Value4. With the S7-300 they can also be generated by the simulation program. Change the value of **SimulateNewRecordSet** from false to true.
3.	Optional: Definition of the database component	You can choose whether the data are to be written into a CSV file or into an MS Access database. 
4.	Set the status tag to “TransferToDatabase”	In the watch table in STEP 7, set the value of the tag **Save_to_DBS.State_Write State_Write** to “TransferToDatabase” (integer value = 1)
5.	Check the operation on the OPC client and in the **Save_to_DBS.State_Write** tag	Current State changes to the OPC client to TransferReady; Last Transaction Status is OK 
6.	Display of the data read out of the controller.  	

Copyright © Siemens AG 2014 All rights reserved

7.3 Operation of the “Download Recipe” scenario

Table 7-3

No.	Action	Remarks
1.	Setting the NodeID	Depending on whether the connection is made via OPC CPM DA or OPC UA, the NodeIDs are different. With OPC UA, the NamespaceIndex is put in front of the string identifier. In the example configuration the nodes used are all in Namespace 7. Sample NodeID for recipe number: **COM: SIMATIC2.S7-300.Save_to_DBS.Recipe_No** **UA: ns=7;s=SIMATIC2.S7-300.Save_to_DBS.Recipe_No**
2.	Optional: Setting the writing behavior of the client.	You can choose, whether the client writes the recipe data automatically, if a request is found on the controller via the state tag**Req_from_DBS.State_Read**. 
3.	Input of recipe data You can enter any data here which you check in step 5. 	
4.	Set the status tag to “TransferRecipe” in the controller.	In the watch table, set **Req_from_DBS.State_Read** to the value “TransferRecipe” (integer value = 1)
5.	Data check in the tag table	In the watch table you should see the data you entered in step 3.

Copyright © Siemens AG 2014 All rights reserved

No.	Action	Remarks
6.		

Name	Address	Display format	Monitor value	Modify value
"Req_from_DBS".State_Read	%DB62.DBB0	DEC_signed	2	1
"Req_from_DBS".Recipe_No	%DB62.DBW2	DEC_signed	0	
"Req_from_DBS".Value_1	P#DB62.DBX4.0	String	'Production 1'	
"Req_from_DBS".Value_2	%DB62.DBD26	Floating-point nu...	345.32	
"Req_from_DBS".Value_3	%DB62.DBW30	Hex	16#10E1	
"Req_from_DBS".Value_4	P#DB62.DBX32.0	Date_and_Time	DT#2013-7-4-12:19:23.3...	
"Req_from_DBS".Status	%DB62.DBW40	Hex	16#0000	
	<Add new>			

Copyright © Siemens AG 2014 All rights reserved

8 Links & Literature

Bibliographic references

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

Table 8-1

	Topic	Title
\1\	OPC	OPC DA 2.05 Specification at http://www.opcfoundation.org/ mn_opc_0.pdf and mn_opc_76.pdf of SIMATIC.NET V8.0
\2\	.NET	Inside C#, Tom Archer .NET Crashkurs, Clemens Vasters, Oellers, Javidi, Jung, Freiberger, DePetrillo Microsoft .NET Framework Programming, Jeffrey Richter

Internet link specifications

This list is by no means complete and only presents a selection of suitable information.

Table 8-2

	Topic	Title
\1\	Link to this document	http://support.automation.siemens.com/WW/view/en/21447513
\2\	Siemens Industry Online Support	http://support.automation.siemens.com
\3\	Examples for Alarm & Events Server	http://support.automation.siemens.com/WW/view/en/26548467
\4\	Sample client for .NET to the OPC UA server	http://support.automation.siemens.com/WW/view/en/42014088
\5\	SIMATIC NET Industrial communication with PG/PC Volume 2 - Interface Programming Manual	http://support.automation.siemens.com/WW/view/en/61630891

9 History

Table 9-1

Version	Date	Modifications
V1.0	04/2005	First version
V2.0	07/2013	• Complete revision • Using the .NET OPC Client API • Reduction of database complexity
V2.1	06/2014	Migration in TIA V13 Use of SIMATIC NET V12

Copyright © Siemens AG 2014 All rights reserved