

Siemens
Industry
Online
Support

APPLICATION EXAMPLE

Programming an OPC UA .NET Client with C# for the SIMATIC NET OPC UA Server

SIMATIC NET OPC UA Server

SIEMENS

Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. **The application examples are not subject to standard tests and quality inspections of a chargeable product and may contain functional and performance defects or other faults and security vulnerabilities. You are responsible for the proper and safe operation of the products in accordance with all applicable regulations, including checking and customizing the application example for your system, and ensuring that only trained personnel use it in a way that prevents property damage or injury to persons. You are solely responsible for any productive use.**

Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. Any further use of the application examples is explicitly not permitted and further rights are not granted. You are not allowed to use application examples in any other way, including, without limitation, for any direct or indirect training or enhancements of AI models.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.

By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice and to terminate your use of the application examples at any time. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.

The Siemens terms of use (https://www.siemens.com/global/en/general/terms-of-use.html) shall also apply.

Cybersecurity information

Siemens provides products and solutions with industrial cybersecurity functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial cybersecurity concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial cybersecurity measures that may be implemented, please visit www.siemens.com/cybersecurity-industry.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Cybersecurity RSS Feed under https://www.siemens.com/cert.

Table of contents

1. Automation task 5

1.1. Overview 6

1.2. Requirements 7

2. Automation 9

2.1. Overview of the overall solution 9

2.2. Description of the core functionality 10

2.3. Hardware and software components used 15

2.4. Alternatives 15

3. Basics 16

3.1. Basics of OPC 16

3.2. Basics of OPC Unified Architecture 17

3.2.1. OPC UA Specifications 17

3.2.2. Structure of the OPC UA Server address space 19

3.2.3. Interface for accessing the OPC UA Server address space 22

3.2.4. Protocols and security mechanisms 24

3.2.5. Differentiation and comparison with OPC Data Access 27

3.3. Basics of S7 Communication 29

3.3.1. General 29

3.3.2. Optimized S7 communication 31

4. Functional mechanisms of this application 32

4.1. OPC UA Client Helper API 34

4.2. Simple OPC UA Client 37

4.3. Convenient OPC UA Client 39

4.3.1. Surface 39

4.3.2. Class diagram 40

4.3.3. Sequence diagrams 42

4.4. S7 Program 45

5. Configuration and configuration 48

5.1. Configuration of the SIMATIC S7 stations 48

5.2. Configuration of the OPC Server Station 50

5.3. Configuration of OPC UA Security 53

5.3.1. OPC UA Remote Communication 53

5.3.2. Certificate 54

5.3.3. Authentication, SecurityPolicy, and MessageSecurityMode 58

6. Installation 60

© Siemens 2025

7. Commissioning of the application 61

8. Operation of the application 64

8.1. Operation Simple Client 64

8.2. Convenient OPC UA Client operation 67

9. Further notes, tips and tricks, etc. 72

10. Appendix 73

10.1. Service and support 73

10.2. Links and Literature 74

10.3. Change documentation 75

© Siemens 2025

1. Automation task

Cause

With OPC Unified Architecture (UA), the SIMATIC NET OPC Server now offers an additional, convenient and high-performance option for process coupling of PC systems to SIMATIC S7, which will gradually replace the existing OPC Data Access (DA) and Alarms & Events (A&E) functions.

The main advantages of OPC UA over conventional OPC interfaces are:

- Communication over the Internet and across firewalls.
- Optimized, robust and fault-tolerant protocol with built-in safety mechanisms.
- OPC UA can be integrated directly into applications on different operating systems with different programming languages.
- All OPC information such as data or alarms are integrated in a namespace.
- Information can be described using object-oriented means.

Target group

This application is intended for end users who need a well-founded introduction to this technology and want to gain experience with the professional creation of OPC UA clients in C# under .NET.

Content

Here you can get an overview of the use of the OPC UA communication interface, which offers data, alarms, and diagnostic information from SIMATIC S7 controllers. You will get to know the components used, standard hardware and software components, as well as specially created application software.

The application software offers examples for the creation of OPC UA clients with C# under .NET. Included are a simplified, reusable API, a simple example, and a complex example with a convenient interface. The example also provides information on how to optimize and expand the application.

© Siemens 2025 |

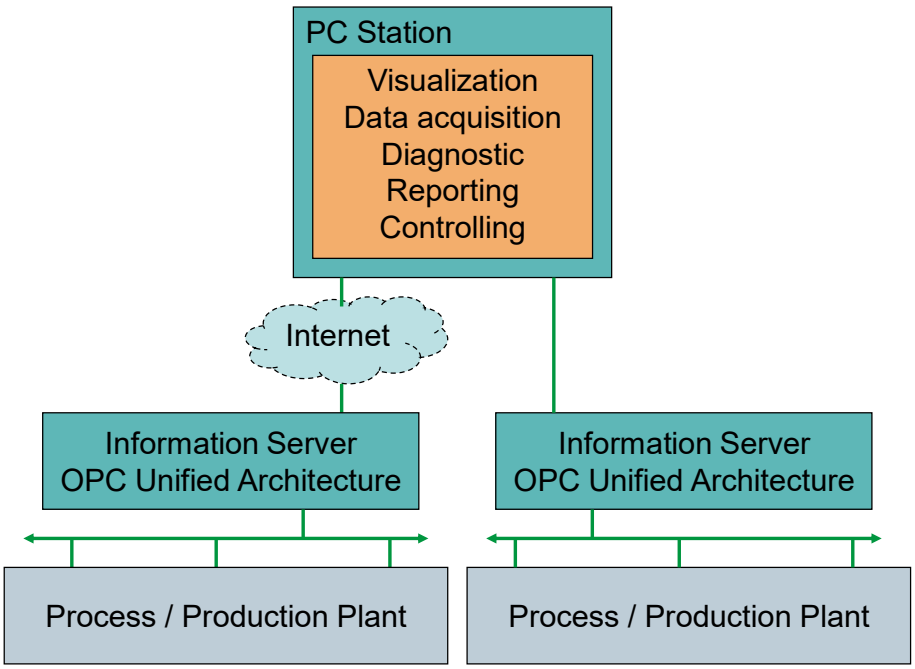
1.1. Overview

Introduction/Introduction

To realize data coupling, standardized mechanisms are preferred today in order to keep such data exchange independent of the bus system or protocol used or even manufacturer. A standardised mechanism for connecting different subsystems will also be used for the exchange of event and alarm messages. OPC UA combines this functionality and also offers authentication and encrypted data transmission as well as extended diagnostic information.

Overview of the Automation Task

The following figure shows an overview of the automation task.



Description of the automation task

In the automation system, the OPC UA Server is to be regarded as an information server that can map and describe individual components as well as the entire system. Thanks to the secure, encrypted access checked with certificates, it is also possible to connect other locations.

The core task of this example is to access process data with the OPC UA interface. This is explained by the creation of a simple but practical, individually created visualization based on the new OPC UA standard.

The application should include the following functionalities:

- Selection of the server including security settings.
- Navigation through the OPC UA namespace of the server and selection of the process variables.
- Read the attributes including the value of the selected process variable.
- Observing the value of the selected process variables.
- Write the value of the selected process variable.
- Use of block services via OPC UA.

Further data processing (e.g. storage in a database, or similar) is not an issue here.

© Siemens 2025 |

1.2. Requirements

Requirements due to the automation task

The sample application is created in C# and uses the interfaces of the OPC Foundation's .NET API.

The user is shown the practical handling of the OPC UA interface under .NET. The basic interface is the .NET Client SDK of the OPC Foundation, which is included with the SIMATIC NET installation.

This interface offers the full range of functions of OPC UA. To simplify the interface, a reduction to the functionality required for this example is performed. An efficient and practical guide to the OPC UA services will be developed.

With the design of a simple GUI interface, the basic functionality of OPC UA is demonstrated. The complete chain of effects between S7 variable(s), OPC UA namespace and access from the client is shown in C#:

- Login, logout, and authentication on the server
- Searching the namespace for variables
- Reading, writing, and observing variables
- Easy error handling

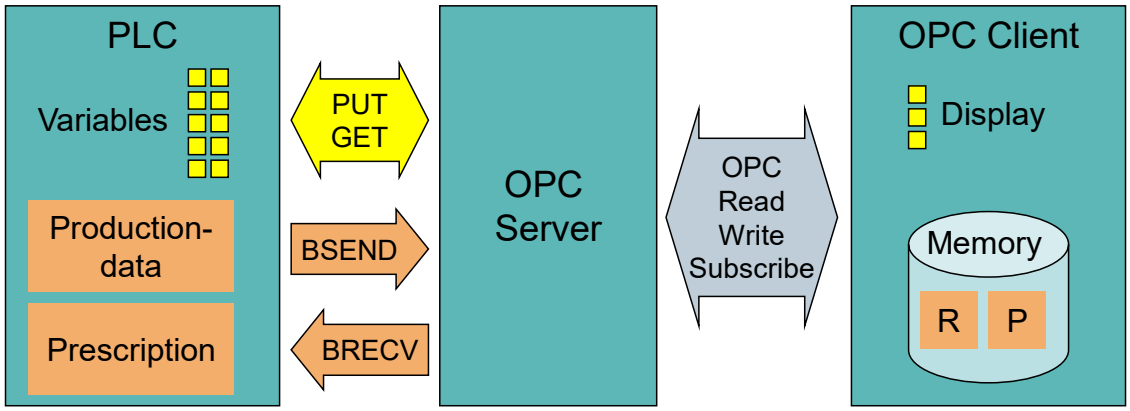
The example shows the symbolic and absolute addressing and describes the use of the variable services read, write and observe for the S7 base types as well as the use of the block-oriented services (receiving and sending large data blocks).

The various diagnostic options and the programmatic processing of error scenarios are explained. The errors can be triggered by simulating disconnections between the various components.

Requirements for the control system

The control system should be able to offer the necessary data structures and data volumes and to simulate changes in values. There is no specific control task, only access to the data is to be illustrated. The data ranges and interaction with the other components are shown in the following figure.

Figure 11



The STEP7 program in the S7 CPU simulates the individual values that are to be received and displayed by the client (variable services). Different data types are used as individual variables.

The PLC program simulates and generates the necessary structures and values for the bidirectional transmission of larger amounts of data and calls the block-oriented services (BSEND, BRECV) accordingly. This is used for STRUCT or ARRAY variables with several 100 bytes in total (recipe data, production records, etc.).

To send data, the PLC actively initiates the transfer of a block-oriented production data record to the OPC UA Server. The PLC receives a block-oriented data record (e.g. recipe) sent by the OPC UA Client and stores it in the corresponding structure in a data module.

Furthermore, the necessary variable tables are made available for testing in STEP 7.

© Siemens 2025

Requirements for the PC Station

The PC station must contain the necessary physical connection with appropriate hardware and software to communicate with the controller. The application for visualization and control should only use the OPC UA interface in order to be able to use any OPC UA server.

The application example is intended to show what generally has to be configured on the server/client PC station and the S7 controllers in order to solve the communication task.

The SIMATIC NET OPC Server is configured in STEP 7, in the configuration console for the PC station, and in the corresponding configuration files for the task (protocol, security settings, certificates, etc.).

The underlying S7 protocol and the necessary connections to the controllers are projected; including the corresponding steps that have to be configured and parameterized on the server PC for OPC UA operation.

Secure communication between client PC and server PC is established with OPC UA means under Windows.

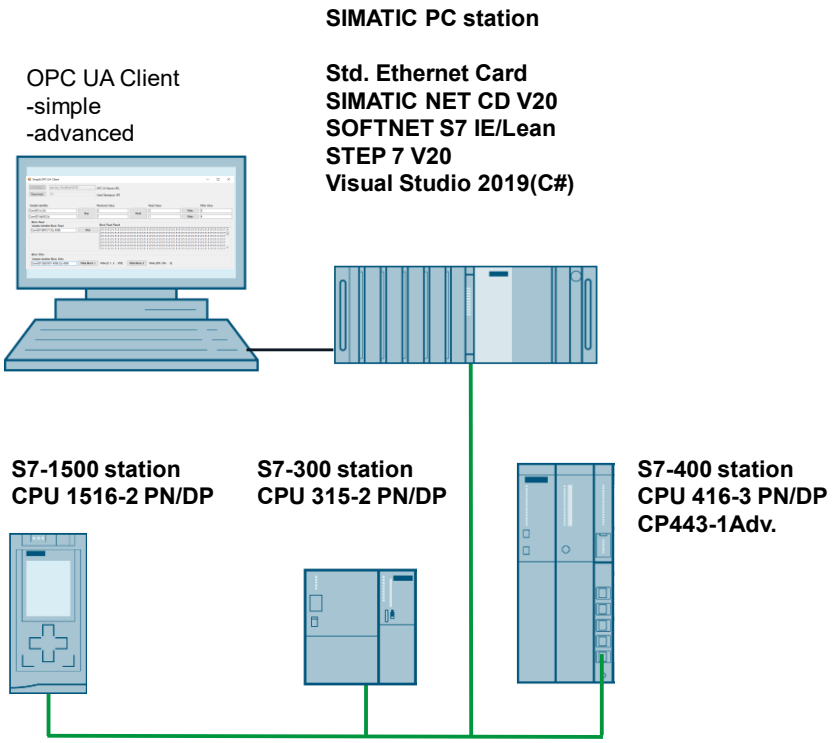
© Siemens 2025 |

2. Automation

2.1. Overview of the overall solution

Overview

The following figure schematically shows the key components of the solution:



Construction

A PC station is connected via Ethernet to a CPU 315-2 PN and a CPU 414-2. A standard Ethernet card is used in the PC.

OPC-UA Client Software

The OPC-UA client in the PC Station is implemented in two levels of complexity. A very simply designed client (Simple OPC UA Client) shows you all the basic functions for getting started with OPC UA. A more complex client (OPC UA .NET Client) with a comfortable interface demonstrates the professional handling of reusable classes.

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

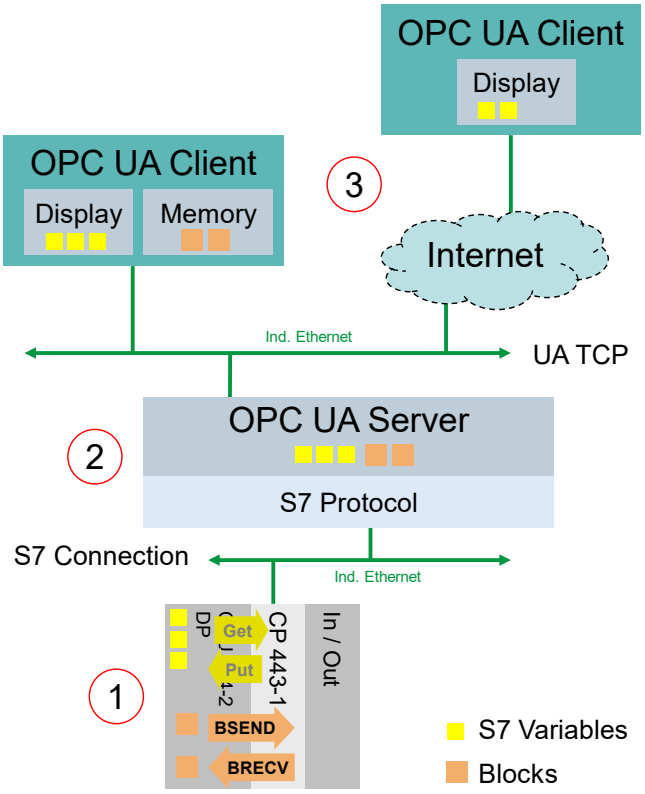
© Siemens 2025

2.2. Description of the core functionality

Overview

The core of the functionality of this example is the SIMATIC NET OPC UA server. This combines the functions and information of the classic OPC servers for data access and alarm & events in a single namespace and allows access to the information via a service-oriented architecture. Communication across the Internet and firewalls is also possible in a secure and high-performance manner.

The following figure shows the chain of effects for data access.



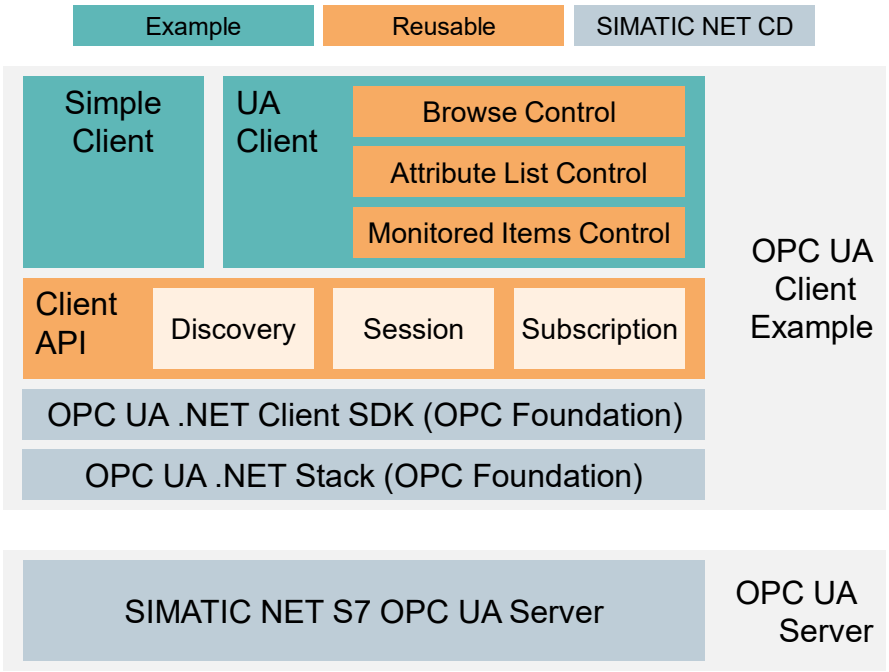
No.	Component	Description
1.	S7 station	The S7 CPU provides S7 variables in data areas such as markers or data modules. The block-oriented services BSEND and BRECV can also be used to actively send and receive larger data blocks from the user program.
2.	OPC UA server	The OPC UA server converts the S7 variables and the block services to OPC UA variables and provides the OPC services such as browse, read, write and data monitoring.
3.	OPC UA client	The OPC UA client can establish a secure connection to the server, navigate through the server's namespace, and read, write, and observe selected variables.

© Siemens 2025 |

Application software components (OPC UA .NET Client)

The following figure shows the software components that are used in the more complex application (OPC UA .NET Client). The OPC UA server and the basic libraries for OPC UA communication on the client side come from the SIMATIC NET CD.

The software components created for the application in C# can be divided into reusable modules and sample code.

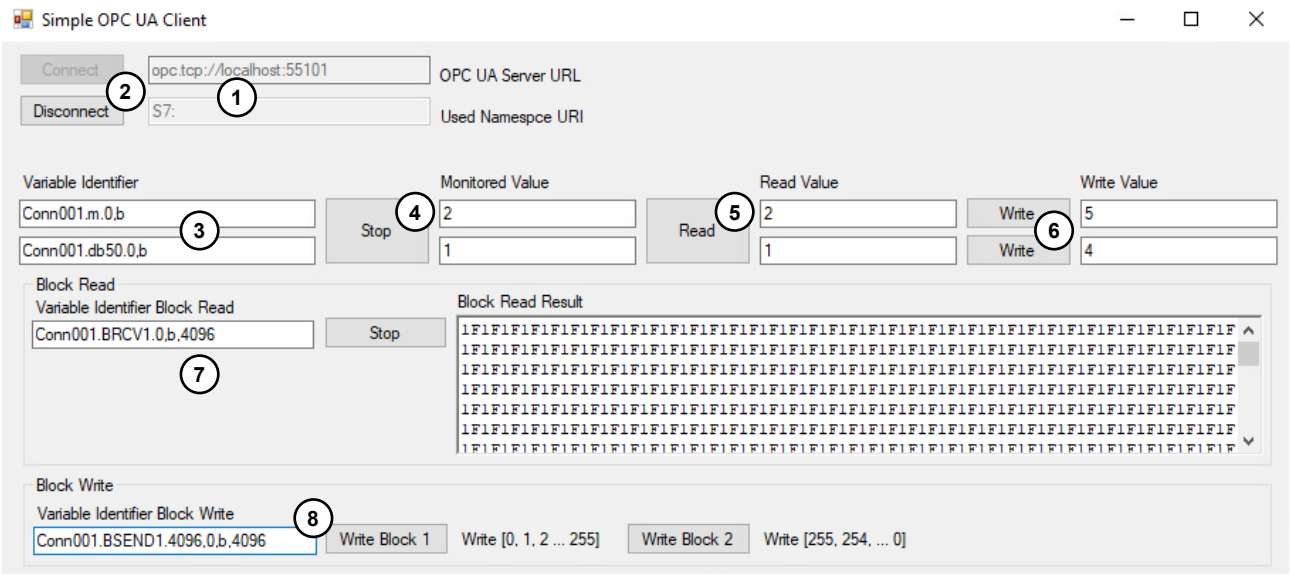


Module	Description
OPC UA .NET Stack	The .NET OPC UA stack from the OPC Foundation for the realization of network communication.
.NET Client SDK	The OPC Foundation's .NET OPC UA Client SDK. The two DLLs of the OPC Foundation are part of the delivery of the SIMATIC NET CD.
Client API	Reusable, simplified, and customized .NET Client API. It offers reusable C# classes for discovery, session and subscription handling.
Simple Client	Simple example of using the Client API with the Connect, Disconnect, Read, Write, and Data Monitoring functions. This example also shows the direct addressing and handling of namespaces.
UA Client	Convenient OPC UA client with the functions Discovery, Connect, Disconnect, Browse, Read of all attributes, Write and Data Monitoring. General functions such as browsing, listing attributes, and monitoring data variables are encapsulated in reusable controls. In this example, the symbolic variables can be browsed and used directly from the browser.
S7 OPC UA Server	The SIMATIC NET OPC UA server implements the necessary server logic for sessions and subscriptions and the data connection to the S7 stations.

© Siemens 2025 |

Simple OPC UA Client interface

The interface of the Simple OPC UA Client is operated via buttons for the individual functions. The simple example shows the use of direct addressing of S7 variables.



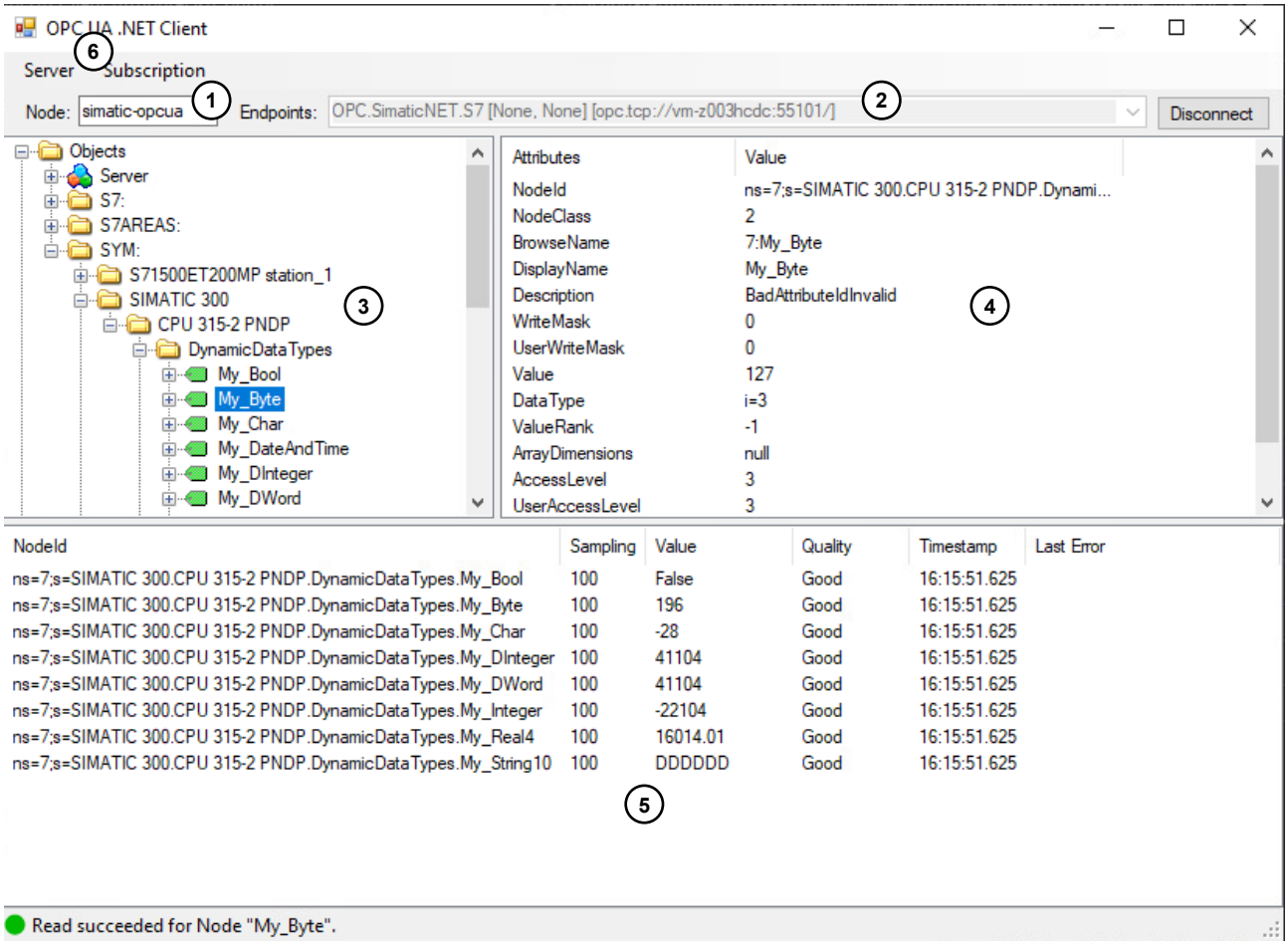
No.	Description
1.	The server URL can be specified in the text box for the server URL. For the SIMATIC NET OPC server, this is made up of opc.tcp://<name of computer>:4845. The Namespace URI text box specifies the namespace used. This is S7: for direct addressing, S7COM: for direct addressing via OPC DA compatible syntax and SYM: for symbolic addressing.
2.	The Connect and Disconnect buttons can be used to establish and disconnect the connection to the OPC UA server. The connection is only established without security. The secure connection setup is explained in the next example.
3.	In the text boxes for the variable identifier, the identifier part of the NodeId is specified. In the case of the namespace S7: this is composed of <S7Verbindung>.<Data area>. <Offset>,<Data type> The NodeId for reading and writing is formed from the identifier and the namespace.
4.	A subscription is created via the Monitor button and two monitored items are created in the subscription with the two NodeIds. The data changes are displayed in the text boxes next to the button. Errors are displayed instead of the values.
5.	The Read button reads the values (Value Attribute) of the two variables with the specified NodeIds and displays them in the text boxes next to the button.
6.	The Write button writes the value from the text box next to the button to the variable identified by the NodeId. To write, Read must first be called, as the text from the text box must be converted into the appropriate data type of the variable. The conversion is done on the basis of the data type supplied by Read.
7.	In the "Block Read" group, data can be received that is actively sent by the S7 with the block service BSEND. This can be used, for example, to send 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 received there with the BRECV block service. Two blocks with different contents can be sent. This can be used, for example, to download recipe data for the S7.

© Siemens 2025

Overview and description of the comfortable interface (OPC UA .NET Client)

The following figure and table describe the interface of the generic OPC UA Client example, which can be used to conveniently access the information of the namespace of an OPC UA server.

The interface also allows browsing the symbolic S7 variables.



No.	Description
1.	The server can be selected via the Endpoints drop-down list. For this purpose, the list of available OPC UA servers is determined by the respective network node. The computer on which the list is to be queried can be entered in the Node text field. If the field is empty, the list is determined on the local computer. The URL of the OPC UA server can also be entered manually. The URL for the SIMATIC NET OPC UA server is composed of opc.tcp://<Name of computer>:55101.
2.	The Connect button can be used to establish or disconnect the connection to the server.
3.	In Browse Control, the entire address space of the connected server is displayed in a hierarchical tree view. Only hierarchical references are displayed.
4.	For the selected node in the Browse Control, the attributes are read and displayed in this control.
5.	With Drag&Drop, variables can be dragged from the Browse Control into the observation window. For the variable, the NodeId, the sampling interval, the value, the timestamp and the status code are then displayed.
6.	The properties of the subscription and monitored items can be changed via the context menu in the observation window or via the application menu. For example, the sampling interval can be changed. In addition, the dialog can be opened for writing. The variables that are selected in the observation window are transferred to the dialog.

© Siemens 2025

Advantages of this solution

The solution presented here offers you the following advantages:

- An easy introduction to OPC UA technology
- Programming in C# for .NET
- easy extensibility of the example
- reusable components
- Access via the Internet and through firewalls possible
- Access rights can be assigned individually for users
- Dealing with certificates, encryption and authentication
- Demonstration of S7 communication

Demarcation

This application does not contain a description of how to process or store the data in the OPC UA Client, e.g. in databases.

Prerequisite knowledge

Basic knowledge of how to use the SIMATIC configuration and programming tool STEP7 as well as the Microsoft Visual Studio 2008 development environment and the C# programming language and object-oriented programming is required.

© Siemens 2025 |

2.3. Hardware and software components used

The application was created with the following components:

Hardware

Component	Anz.	Article	Note
S7-400 CPU 416-3 PN/DP	1	6ES7416-3XR05-0AB0	Alternatively, any other S7-400 with a PNIO interface can be used.
CP 443-1 Advanced	1	6GK7443-1GX20-0XE0	(Optional) Alternatively, any other S7-capable Ethernet CP can be used.
S7-300 CPU 315-2 PN/DP	1	6ES7 315-2EH14-0AB0	Alternatively, any other S7-300 with a PNIO interface can be used.
CP 343-1 Advanced-IT	1	6GK7 343-1GX31-0XE0	(Optional) Alternatively, any other S7-capable Ethernet CP can be used.
S7-1500 CPU 1516-3 PN/DP	1	6ES7 516-3AN00-0AB0	Alternatively, any other S7-1500 can be used.
Standard PC as OPC UA Server	1	-	Standard PC (e.g. PGs) running Windows 7/8/10.
Standard PC as OPC UA Client	1	-	Alternatively, the client can also be operated locally on the PC.

Software

Component	Anz.	Article	Note
SIMATIC NET DVD V20	1	6GK1704-1LW24-0AA0	-
STEP 7 Professional V20	1	6ES7822-1AA24-0YA5	-
Microsoft Visual Studio 2019	1		Alternatively, newer versions of VS are also possible.
.NET Framework 4.8	1	-	-

Sample files and projects

The following list contains all the files and projects used in this example.

Component	Note
42014088_OPC_UAClient_DOKU_V2_de.pdf	This document.
42014088_OPC_UAClient_CODE_V2.zip	This packaged file contains the OPC UA client with sources and the associated STEP 7 V20 project.

2.4. Alternatives

OPC Data Access based on COM

Today, the automation task is typically solved with the COM-based classic OPC data access interface.

Benefits of the solution with COM OPC Data Access:

- Wide spread of the interface.
- Many applications for different tasks support the interface.
- Easy access for local applications.

Disadvantages of the solution with COM OPC Data Access:

- Complicated DCOM configuration for remote access.
- No communication over firewall or internet boundaries possible.
- OPC clients can only be operated on Windows PC systems.
- Limited security mechanisms and user authentication only as part of the DCOM configuration.
- No user-related access rights possible.

3. Basics

3.1. Basics of OPC

Overview

In recent years, the OPC Foundation (an interest group of well-known manufacturers for the definition of standard interfaces) has defined many software interfaces to standardize the flow of information from the process level to the management level. In the past, different OPC specifications have been developed according to the different requirements within an industrial application: Data Access (DA), Alarm & Events (A&E), Historical Data Access (HDA) and Data eXchange (DX). Access to process data is described in the DA specification, A&E describes an interface for event-based information including acknowledgment, HDA describes functions for archived data, DX defines server-to-server cross-communication.

Based on the experience of these classic OPC interfaces, the OPC Foundation has defined a new platform called OPC Unified Architecture (UA). The aim of this new standard is to provide a generic description and uniform access to all information that needs to be exchanged between systems or applications. This includes the functionality of all previous OPC interfaces. In addition, it is to be possible to integrate the interface natively into the respective system, regardless of the operating system and regardless of the programming language used to create the system.

This example deals with the OPC Unified Architecture interface. Detailed documentation can be found on the SIMATIC NET CD. Further information can be found under www.opcfoundation.org.

What is OPC

In the past, OPC was a collection of software interfaces for data exchange between PC applications and process devices. These software interfaces were defined according to the rules of Microsoft COM (Component Object Model) and could therefore be easily integrated into Microsoft operating systems. COM or DCOM (Distributed COM) provides the functionality of inter process communication and organizes the exchange of information between applications, even across computer boundaries (DCOM). An OPC client (COM client) can thus exchange information with an OPC server (COM server) using mechanisms of the Microsoft operating system.

The OPC Server provides process information of a device at its interface. The OPC client connects to the server and can access the data offered.

The use of COM or DCOM means that OPC servers and clients can only be operated on a Windows PC or in a local network, and they usually must communicate with the corresponding automation system via proprietary protocols. For network communication between client and server, additional tunneling tools often must be used to get through firewalls or to bypass the complicated DCOM configuration. In addition, the interface can only be accessed natively with C++ applications, .NET or JAVA applications can only be accessed via a wrapper layer. In practice, these limitations lead to additional communication and software layers, which increase the configuration effort and complexity.

Due to the widespread use of OPC, the standard is increasingly used for the general coupling of automation systems and no longer only for the original use case as a driver interface in HMI and SCADA systems for accessing process information.

To solve these limitations in practice and to meet the additional requirements, the OPC Foundation has defined a new platform called OPC Unified Architecture over the past five years, which provides a uniform basis for the exchange of information between components and systems. OPC UA will also be available as an IEC 62541 standard and thus also forms the basis for other international standards.

OPC UA offers the following features:

- Summary of all previous OPC features and information such as DA, A&E and HDA in one generic interface.
- Use of open and platform-independent protocols for interprocess and network communication.
- Internet access and communication through firewalls.
- Integrated access control and security mechanisms at the protocol and application level.
- Extensive mapping capabilities for object-oriented models, objects can have variables & methods, and can fire events.
- Extensible type system for objects and complex data types.
- Transport mechanisms and modelling rules form the basis for other standards.
- Scalability from small embedded systems to enterprise applications, and from simple DA address spaces to complex, object-oriented models.

© Siemens 2025 |

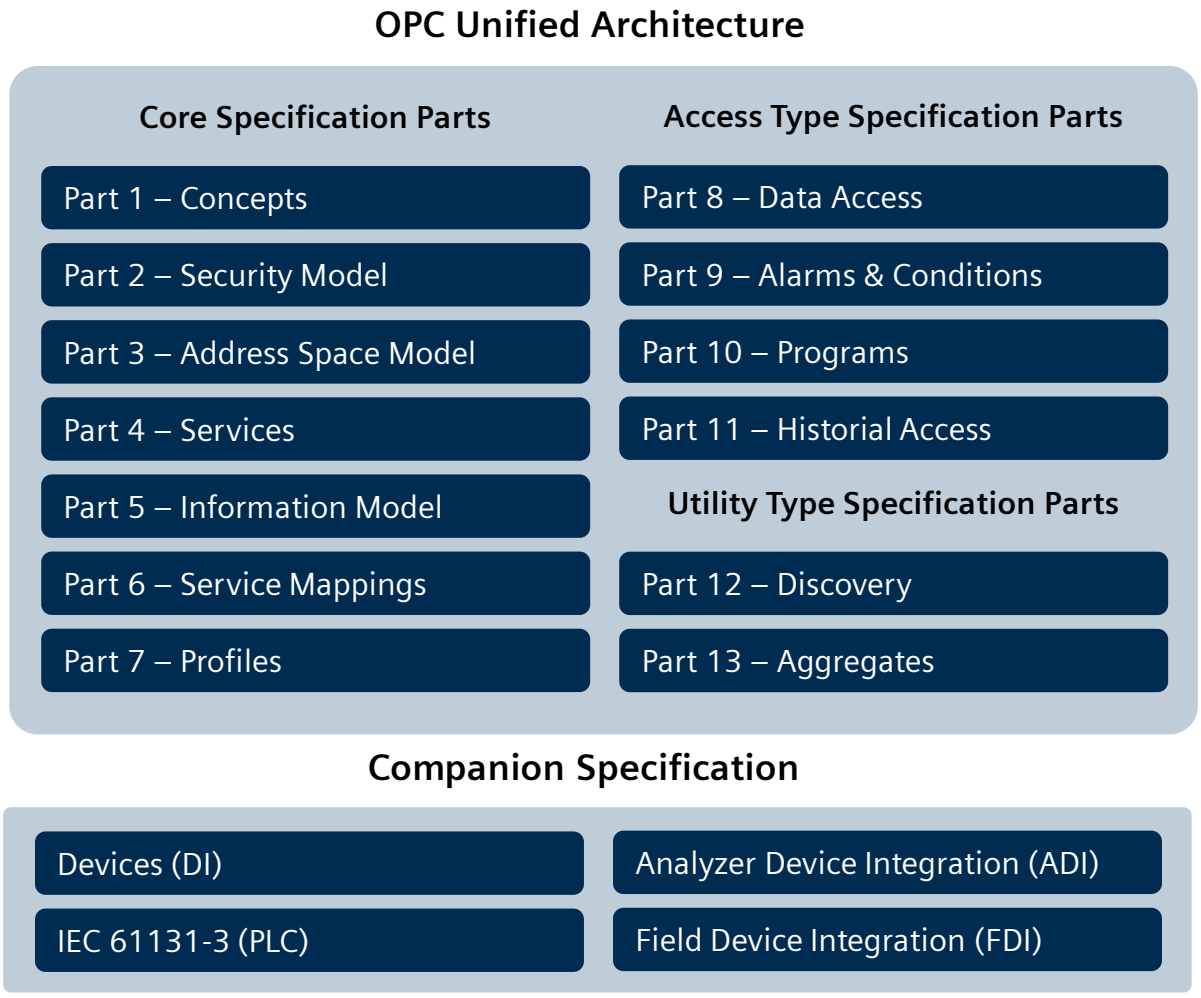
3.2. Basics of OPC Unified Architecture

This chapter explains the basics of OPC Unified Architecture necessary for the example.

3.2.1. OPC UA Specifications

Overview

The OPC UA specifications are divided into different parts due to the IEC 62541 standardization. The following illustration shows an overview of the different parts.



Parts 1 to 7 form the basis of the technology and the implementation of OPC UA applications. Parts 3 to 5 in particular form the core of the standard.

Parts 8 to 11 define OPC-specific information models for the provision of classic OPC information such as current process data or alarms.

Additional tools are defined in part 12 and 13.

In addition, so-called companion specifications are created, which define additional information models based on OPC UA in cooperation with other standardization organizations. The models and information in other standards form the basis and the Companion specification defines how this information is described and transported with OPC UA.

Note Relevant for this application are parts three to five and part eight. The description of the other parts is included to provide a complete overview of the OPC Unified Architecture.

© Siemens 2025

List of Specifications

The table, the list of the specification and its contents. The specifications currently relevant for the SIMATIC NET Server are highlighted.

Specification	Description
Part 1 – Concepts	This non-normative part gives an overview of the standard.
Part 2 – Security	The requirements for security and an introduction to the basics are described in the second part, which is also non-normative.
Part 3 – Address Space Model	This part defines the basic rules and basic elements for the structure of the address space of an OPC UA server. These rules form the basis for the information models in Part 5, 8 to 11 and the Companion Specifications.
Part 4 – Services	This document is the only part that defines the interface for accessing all OPC UA information. It specifies a list of methods, the so-called services. These services are generic and form the basis for all information models.
Part 5 – Information Model	The basic information model defines the entry points into the address space and basic types such as data types or object types. Together with Parts 3 and 4, this part forms the core of OPC UA.
Part 6 – Service Mapping	The services in Part 4 are defined independently of the transport mechanism used. This part specifies the implementation of the services on different types of serialization, backup and transport protocols for the messages between OPC UA client and server. This part forms the basis for the implementation of communication stacks and is not relevant for users of the technology.
Part 7 – Profiles	A profile defines a subset of OPC functionalities for different use cases that can be offered by an OPC UA server or used by an OPC UA client. This part defines the list of profiles for OPC UA.
Part 8 – Data Access	This part defines variable types, properties, and quality status codes for process data. All other necessary concepts are already included in Parts 3 to 5.
Part 9 – Alarms and Conditions	This part defines the model for describing condition monitors and process alarms and reporting status changes via events. The necessary concepts for events are already included in Parts 3 to 5.
Part 10 – Programs	This part defines how actions that have been running for a longer period can be started and monitored. This is done based on state machines, the handling of which is defined in OPC UA in Part 5.
Part 11 – Historical Access	This is where access to historical data and events is defined.
Part 12 – Discovery	Defines how OPC UA servers can be found on the network.
Part 13 – Aggregates	This part defines aggregate functions for data compression, such as mean or maximum value over a time range. The aggregates can be applied to current data or to historical data.
Devices (DI)	This companion specification defines a generic model for configuring and diagnosing devices.
IEC 61131-3 (PLC)	This companion specification defines a mapping of the IEC 61131-3 software model and the standardized control programming languages to an OPC UA server address space.
Analyzer Device Integration (ADI)	This companion specification defines a model for the configuration and data connection for complex devices for process analysis based on DI
Field Device Integration (FDI)	This companion specification defines a model for the complete engineering of field devices based on Electronic Device Description Language (EDDL) and Field Device Tool (FDT).

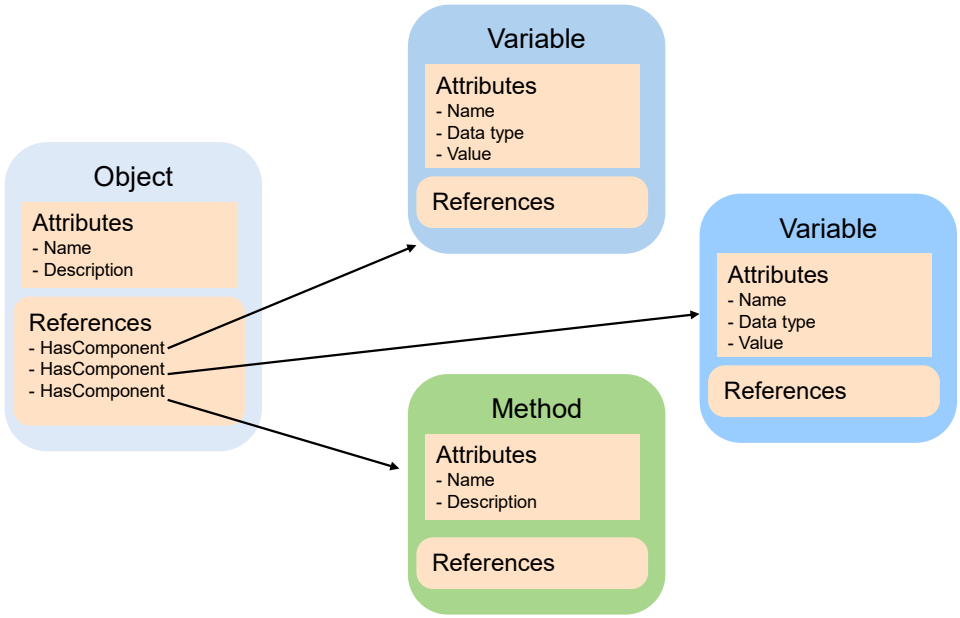
© Siemens 2025 |

3.2.2. Structure of the OPC UA Server address space

Nodes in Address Space

A node in the OPC UA address space is of a certain type, such as an object, variable, or method, and is described by a list of attributes. All nodes have common attributes such as name or description and specific attributes such as the value of a variable. The list of attributes is not expandable. Additional information about the node can be added as a property. Properties are a special type of variable.

The nodes are connected to each other with references. The references are typified. There are two main groups, hierarchical references such as HasComponent for the components of an object or non-hierarchical references such as HasTypeDefinition for a connection from an object instance to an object type. An example of nodes and the connecting references is shown in the following figure.



Available types of nodes in the address space

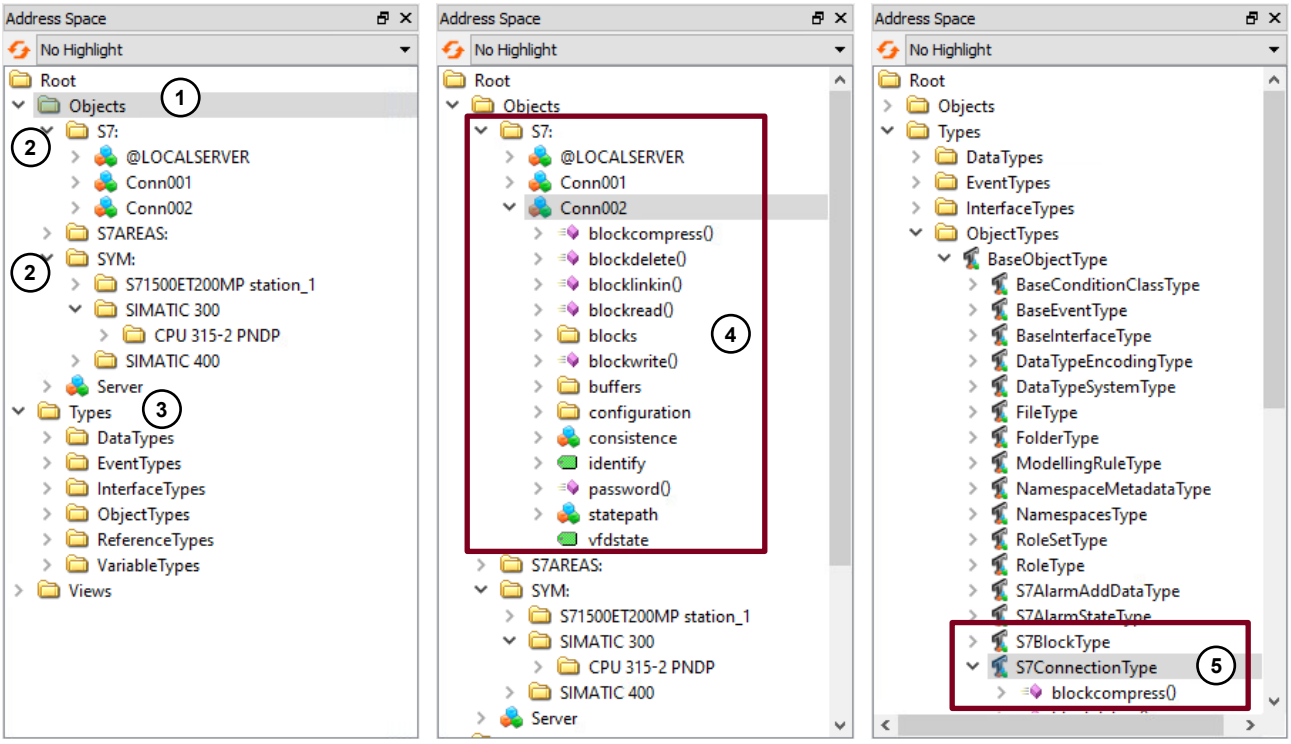
The defined node types are listed in the table. The list of types is not expandable.

Node	Description	Example
Object	An object serves as a typed container for variables, methods, and events.	The objects that represent an S7 connection are always structured in the same way.
Variable	Variables represent the data of objects or, as a property, the properties of a node.	S7 variable in a data module.
Method	Methods are components of objects and can have a list of input or output parameters. The parameters are described via defined properties.	Method blockRead( ) on an S7 connection object, which can be used to read a block from an S7.
View	Views represent a part of the address space. The node serves as an entry point and as a filter when browsing.	Views do not exist in the SIMATIC NET Server.
Object type	Object types provide information about the structure or components of an object.	S7ConnectionType describes the components that exist on an S7 connection object.
Variable Type	VariabletypesTypes typically describe which properties or data types can be found on an instance of the type (variable).	AnalogItemType defines that a variable of this type provides the properties EngineeringUnits and EURange.
Reference Type	Reference types define the possible types of references between nodes.	A method is referenced by an object with HasComponent.
Data type	Data types describe the contents of the value of a variable.	The value of a variable can have a data type of Double.

© Siemens 2025

Structuring the address space

The basic structure of the OPC UA address space is defined in Part 5. The following figure shows part of this structure and SIMATIC NET specific parts. The different areas are described in the table below.



No.	Description
1.	In the Objects directory, instances such as objects and variables can be found. A data access client finds the variables for data access under this directory. In addition to the SIMATIC NET specific directories, you will also find the Server object, which is defined by OPC UA. It contains information about the range of functions and status of the server.
2.	The two directories S7: and SYM: under Objects are specific to the SIMATIC NET OPC UA Server. Under S7: the configured S7 connections are listed as objects. Under SYM: you can find the symbols from the STEP 7 project.
3.	The Types directory contains the various type nodes for DataTypes, ObjectTypes, ReferenceTypes, and VariableTypes.
4.	An S7 connection object provides various status information and methods. For example, blocks in the S7 can be edited or read out via methods. In addition to the methods, properties provide information about the configuration of the S7 connection.
5.	The S7ConnectionType associated with the S7 connection object can be found in the ObjectTypes directory. It describes which methods and variables are at least present on the instance. The rules for the type system are described in detail in /2/.

© Siemens 2025 |

Namespaces and NodeId

Each node in the OPC UA address space is uniquely identified by a NodeId. This NodeId is composed of a namespace to distinguish identifiers from different subsystems and an identifier, which can be either a numeric value, a string, or a GUID.

Strings are typically used for the identifier. This is analogous to OPC Data Access, where the ItemId as an identifier is also a string. Numeric values are used for static namespaces such as type system.

OPC UA defines a namespace for the nodes defined by OPC. The OPC UA servers also define one or more namespaces. The relevant namespaces for the SIMATIC NET OPC UA Server are listed below.

Namespace	Description
http://opcfoundation.org/UA/	Used for nodes defined in OPC UA Part 5. These are nodes that form the basic structure of the address space and nodes represent the types defined by OPC UA.
S7:	Namespace for direct addressing of S7 variables with an optimized syntax.
S7COM:	Namespace for addressing with old identifiers. The namespace is S7-OPC-DA compatible. This namespace can be used with UA, but it is not searchable.
S7AREAS:	Namespace for areas with alarm hierarchy.
SYM:	Namespace for symbolic addressing of S7 variables. The symbol information is exported from the STEP 7 project.

Attributes of the nodes

The most important attributes of nodes are listed as examples in the following table. The focus is on the variable node type.

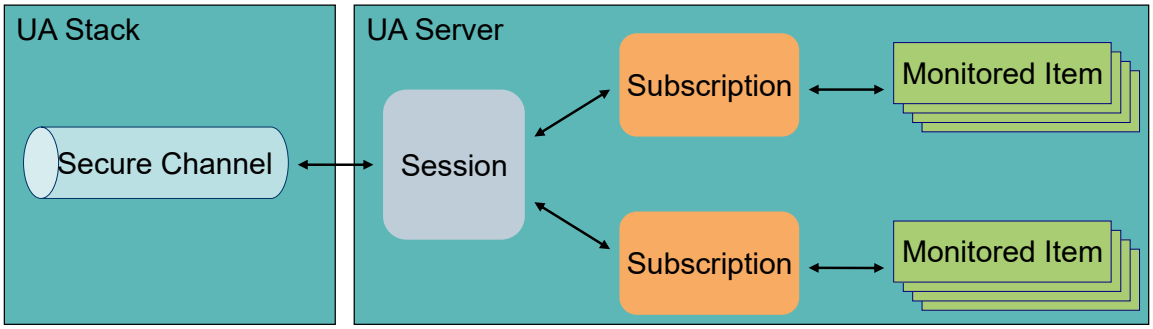
Attributes	Node	Description
NodeId	All	Unique address of the node.
DisplayName	All	Localized display name for the node. The language depends on the language requested by the client for the connection and the languages supported by the server.
BrowseName	All	Unlocalized name for the node. The name contains a namespace and is especially relevant when using types.
NodeClass	All	Type of node such as object, variable or method.
Description	All (optional)	Optional localized description of the node.
Value	Variable	Value of the variable. The timestamp and status of the value, as with all other attributes, are supplied together with the value of the attribute when read.
Datatype	Variable	Data type of the variable or value attribute. Data types are e.g. OPC UA defined data types such as Int32, Double or String or structured data types.
ValueRank	Variable	Specifies whether the value attribute is a scalar value, an array, or a multidimensional array.
AccessLevel	Variable	Specifies whether the variable can be read or written.

© Siemens 2025 |

3.2.3. Interface for accessing the OPC UA Server address space

Communication channel and application objects

The following figure shows the various objects that can be created during data exchange between OPC Client and Server. The objects are described in the following table.



Object	Description
Secure Channel	The secure communication channel is implemented in the OPC UA stack. The objects at the application level are viable independently. However, they can only be created, used or modified in the context of a secure channel. If a new Secure Channel is established after a connection interruption, it must be assigned to the session at the application level.
Session	The session in the server is the logical connection between the OPC UA client and the server. It keeps the user information and language settings for the connection. The session is deleted by the server if no calls are received from the client within the timeout. The timeout is specified by the client. The session is bound to a secure channel, but can be assigned to a new secure channel if communication has been disrupted.
Subscription	A Subscription object can be created by the client for grouping monitored items. Monitored items are used to observe changes in values or to receive event messages. The subscription is deleted from the server if no data or KeepAlive message could be sent to the client within the timeout. The timeout is specified by the client.

Methods of establishing a connection

The table explains the most important methods of the OPC UA interface for establishing connections.

Method	Description
OpenSecureChannel	Opens a secure communication channel between client and server. To open the connection, the URL of the server, the application certificates and the security settings are required.
CreateSession	Create an application session in the context of a secure channel.
ActivateSession	Activate the session by passing user authentication and language settings. This method is also used to assign an existing session to a new Secure Channel or to switch users.
CloseSession	Closes the application session.

© Siemens 2025 |

Methods of the Session Object

The table explains the most important methods of the OPC UA interface regarding the session.

Method	Description
Browse	Returns the list of nodes that can be reached from a starting node via a reference. The number of nodes can be limited by filters. For each node, information is provided, which is necessary for displaying it in a tree view, for example.
Read	Reads a list of attributes of nodes. The method can be used to read values of variables (value attribute) as well as metadata such as the data type of a variable (DataType attribute).
Write	Writes a list of attributes of nodes. This method is typically used to write values of variables. If the server allows it, other attributes can also be written.
CreateSubscription	Create a subscription to receive data changes or event messages. The subscription is used to group information that is to be observed. All new data or events are delivered as a package at adjustable time intervals for a subscription.
DeleteSubscription	Delete a subscription.

Methods of the Subscription Object

The table explains the most important methods of the OPC UA interface regarding subscription.

Method	Description
ModifySubscription	Changes the settings of a subscription, such as the publish interval, in which new data is collected for the client and sent together.
CreateMonitoredItems	Create a list of monitored items in a subscription. A monitored item is used either to monitor a value of a variable or to monitor event messages. Both types of monitored items can be mixed in one call to this method. In this application, only data changes are monitored.
ModifyMonitoredItems	Changes the settings of a list of MonitoredItems, such as the sampling interval for monitoring for value changes.
DeleteMonitoredItems	Deletes a list of monitored items in a subscription.
Publish	Method for transferring data packets for a subscription with value changes and event messages in the publish interval. This method is not visible in the Client API. The functionality is implemented there as a callback to the client application.

© Siemens 2025 |

3.2.4. Protocols and security mechanisms

OPC UA Communication Architecture

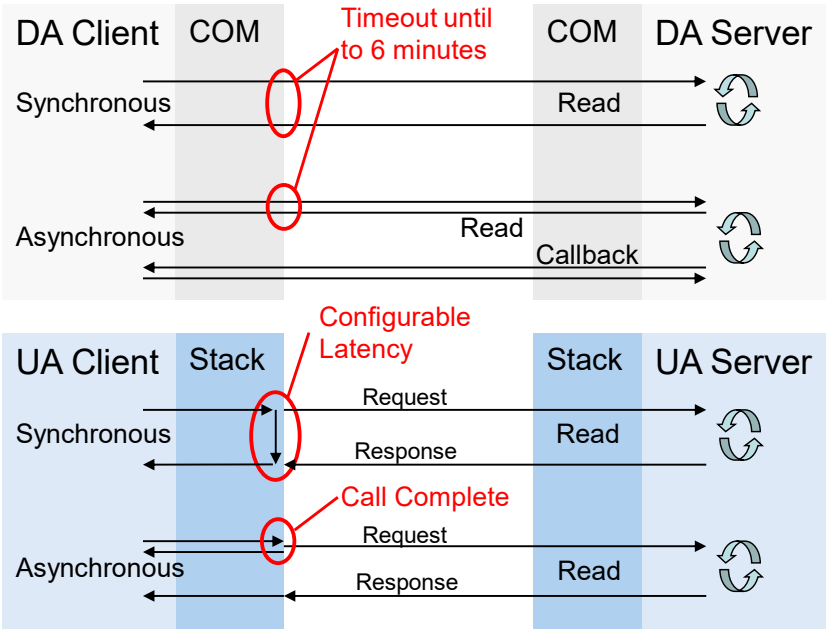
The services for accessing the information in an OPC UA Server address space such as Browse, Read and Write are specified abstractly and independently of the transport protocol in Part 4.

The various bindings for the transmission of service messages between OPC UA client and server are defined in Part 6. A binding is composed of a protocol, security mechanism and serialization type for the data.

The bindings are implemented in communication stacks. Currently, there are three implementations from the OPC Foundation, namely in ANSI C, C# / .NET and JAVA. This application uses the C# / .NET stack.

The methods on the API of the stack for the application correspond to the services in Part 4 with specific data types from the respective programming language. This allows application development to access a native API in the corresponding programming language. The application can also be implemented independently of the binding used. New bindings can be extended by replacing the OPC UA stack.

Synchronous and asynchronous calls



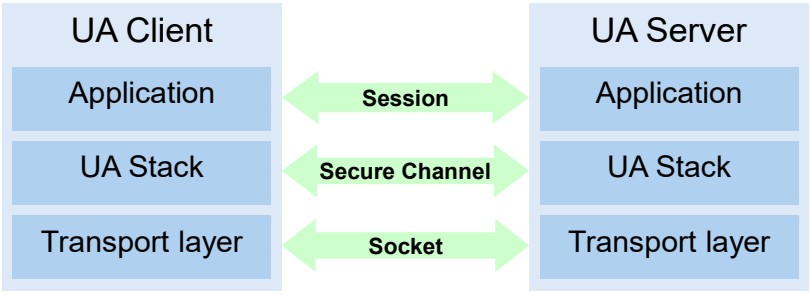
With COM, all calls to the server are synchronous. For this reason, additional asynchronous functions have been defined there for a few actions such as read and write. A synchronous call starts the action in the server. After the action is completed, a synchronous callback is sent from the server to the client. The synchronous call to start the action can also block asynchronous calls if the network connection is lost.

In OPC UA, all calls to the server are asynchronous, and the specification does not make a distinction between synchronous and asynchronous methods. After the request message is written to the network, the asynchronous call is returned to the client application. Blocking is therefore not possible with asynchronous calling. Since an asynchronous call can always be made synchronously, the stacks also offer all OPC UA methods as synchronous calls. To do this, the call is held in the stack until the response message has arrived from the server or until the timeout has expired. The timeouts can be set individually for each call. For the server, there is no difference between synchronous and asynchronous calls.

© Siemens 2025 |

Security layers

The different security layers of OPC UA are described in the following figure and table.



Layer	Description
Socket	At the socket level, in addition to or as an alternative to the Secure Channel, a connection-oriented security of the socket connection can be used by Secure Socket Layer (SSL) or by a Virtual Private Network (VPN).
SecureChannel	At the SecureChannel level, the applications are mutually authenticated and the communication is secured on a message-based basis. Each message is signed and encrypted to ensure the integrity and secrecy of the messages. These mechanisms are based on certificates that uniquely identify the applications based on a Public Key Infrastructure (PKI) system. The replacement of these certificates as the most important step in the security configuration is described in the next section.
Session	User authentication is performed at the session level.

Security configuration options

The table describes the different configuration options for the security mechanisms.

Option	Description
Security Policy	**None** – No security is used in the Secure Channel. **Basic128Rsa15** – Set of algorithms for security. **Basic256** – Set of algorithms for security with longer keys.
Message Security Mode	**None** – The messages are not backed up. **Sign** – The messages are signed. **Sign&Encrypt** – The messages are signed and encrypted.
User Authentication	**Anonymous** – No user authentication is required. **User Password** – User authentication is performed with username and password. **Certificate** – User authentication is done by a certificate.

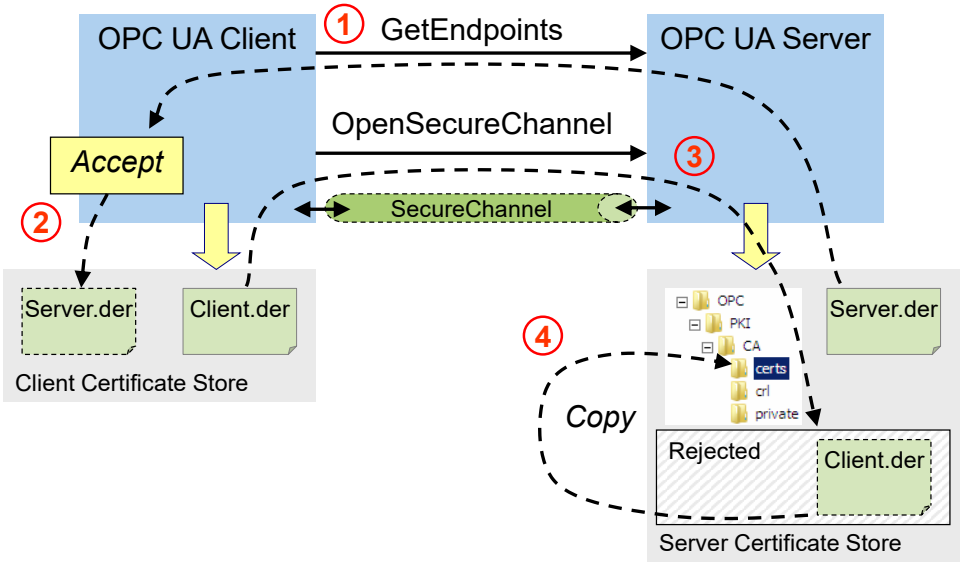
© Siemens 2025 |

Exchange of certificates

The exchange of certificates between client and server and the acceptance of certificates is explained in the following figure and table.

If all participating applications implement the OPC UA guidelines for security configuration, only a manual step on the server is necessary to exchange the certificates, as the certificates are automatically exchanged between the applications and only the certificates have to be accepted by an administrator.

Since not all applications implement the automatic steps yet, the manual exchange of certificates is explained in chapter 5.3.2.

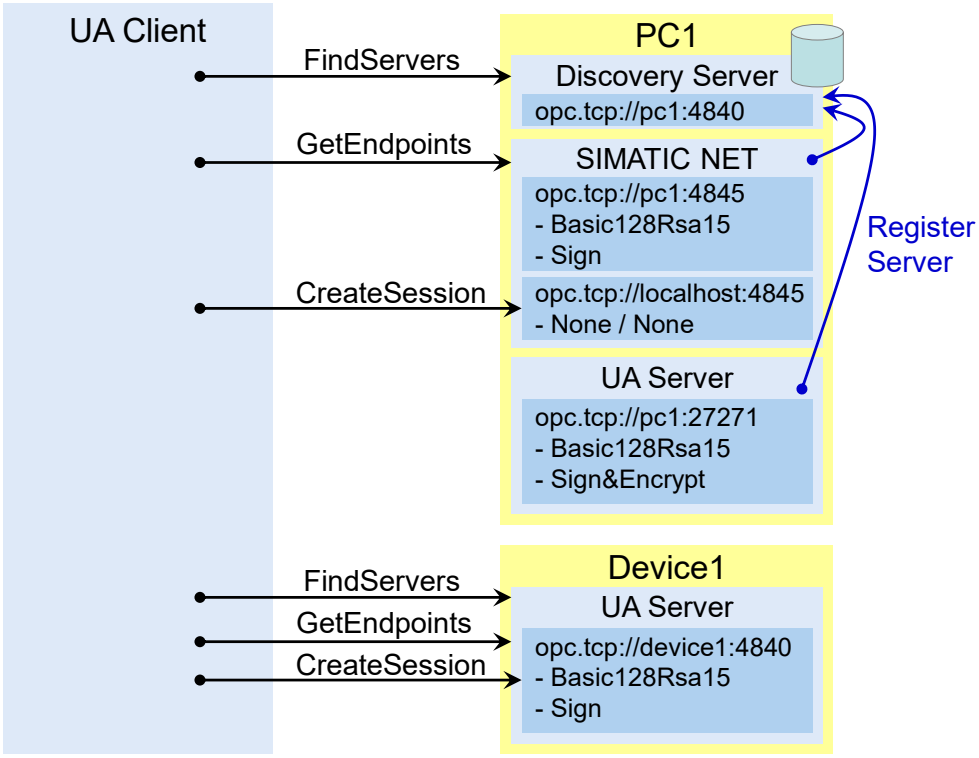


Step	Description
1.	Before the client can connect to a server, it needs the necessary information such as the security mechanisms required by the server, protocol and the address to connect. This information identifies a so-called endpoint. The call GetEndpoints delivers the available endpoints of a server. With the description of the endpoint, the server also provides its certificate.
2.	After selecting the endpoint with the security settings, the user is asked if they want to accept the certificate. If so, it is stored in the client's certificate store.
3.	When OpenSecureChannel is called, the client certificate is passed to the server. If the certificate is not known in the server, it is stored in a rejected directory.
4.	With a configuration tool of the server, certificates from the rejected directory can be accepted. In the process, they are moved to the server's certificate store.

© Siemens 2025 |

Server Discovery

So far, a Local Discovery Service (LDS) has been defined for OPC UA Discovery, which is comparable to the OPC Enum in classic OPC in terms of basic functionality.



An LDS provides a list of OPC UA servers available locally on the network node. By default, the LDS runs on port 4840 registered for OPC UA. This means that the LDS can always be accessed via opc.tcp://<Node>:4840 as a URL. The servers on a PC register with the LDS.

A client can select a server and establish a connection by following these steps:

- Establishing a connection without security with port 4840 and calling FindServers. This call provides the list of available servers and their discovery URL.
- Establish a connection without security to the discovery URL of the desired server and call GetEndpoints. This call provides the list of enpoints with the endpoint URLs and the security settings of the endpoints.
- Establish a connection with the endpoint URL and the required security settings. CreateSession can then be used to open an application session with the server.
- If only one OPC UA server is available on a system, it can run on the standard port 4840, as all servers must also implement FindServers and then only deliver themselves. In this case, the endpoints also use port 4840.

3.2.5. Differentiation and comparison with OPC Data Access

Overview

OPC Unified Architecture takes up all the features of the classic OPC interfaces and implements them only in a common approach for the various classic OPC interfaces.

This application deals with the Data Access functionality in OPC UA. For this reason, the implementation of OPC Data Access on OPC UA is explained in this chapter.

© Siemens 2025

Structure of the address space

The table explains the implementation of the OPC DA address space to the concepts of OPC UA.

OPC Data Access	OPC Unified Architecture
Nodes in Address Space	
Directories are used for the hierarchical structuring of the address space.	Directories can be created with folder objects. The hierarchy is built with Organize's references.
OPC items represent data points in the address space. They are the pages of the directories.	Variable nodes are used for mapping OPC items.
Properties and attributes	
ItemId for OPC Items	NodeId for all nodes in the address space
Property Item Canonical Data Type	DataType, ValueRank, and ArrayDimension attributes
Properties Item Value, Quality and Timestamp	The Value attribute provides value, status, and timestamp.
Property Item Access Rights	AccessLevel and UserAccessLevel attributes

Access to data

The table explains the implementation of the OPC DA access to data on the concepts of OPC UA.

OPC Data Access	OPC Unified Architecture
Context	
COM Object OPCServer	OPC UA Session
COM Object OPCGroup	OPC UA Subscription
OPCItem in a group	Data Monitored Item in a Subscription
Creating context	
CoInitializeEx CoInitializeSecurity CoCreateInstanceEx creates OPCServer	OpenSecureChannel CreateSession ActivateSession
AddGroup RemoveGroup IOPCGroupStateMgt::SetState	CreateSubscription DeleteSubscriptions ModifySubscription
AddItems RemoveItems	CreateMonitoredItems DeleteMonitoredItems

OPC Data Access	OPC Unified Architecture
Access to information	
ChangeBrowsePosition / BrowseOPCItemIDs GetItemID / QueryAvailableProperties	Browse
IOPCItemIO::Read IOPCSyncIO::Read IOPCSyncIO2::ReadMaxAge IOPCAsyncIO2::Read IOPCAsyncIO3::ReadMaxAge IOPCItemProperties::GetItemProperties	Read
IOPCItemIO::WriteVQT IOPCSyncIO::Write IOPCSyncIO2::WriteVQT IOPCAsyncIO2::Write IOPCAsyncIO3::WriteVQT	Write
OnDataChange	Publish
GetStatus ShutdownEvent	Reading or Monitoring the ServerState and ServerStatus variables

© Siemens 2025 |

3.3. Basics of S7 Communication

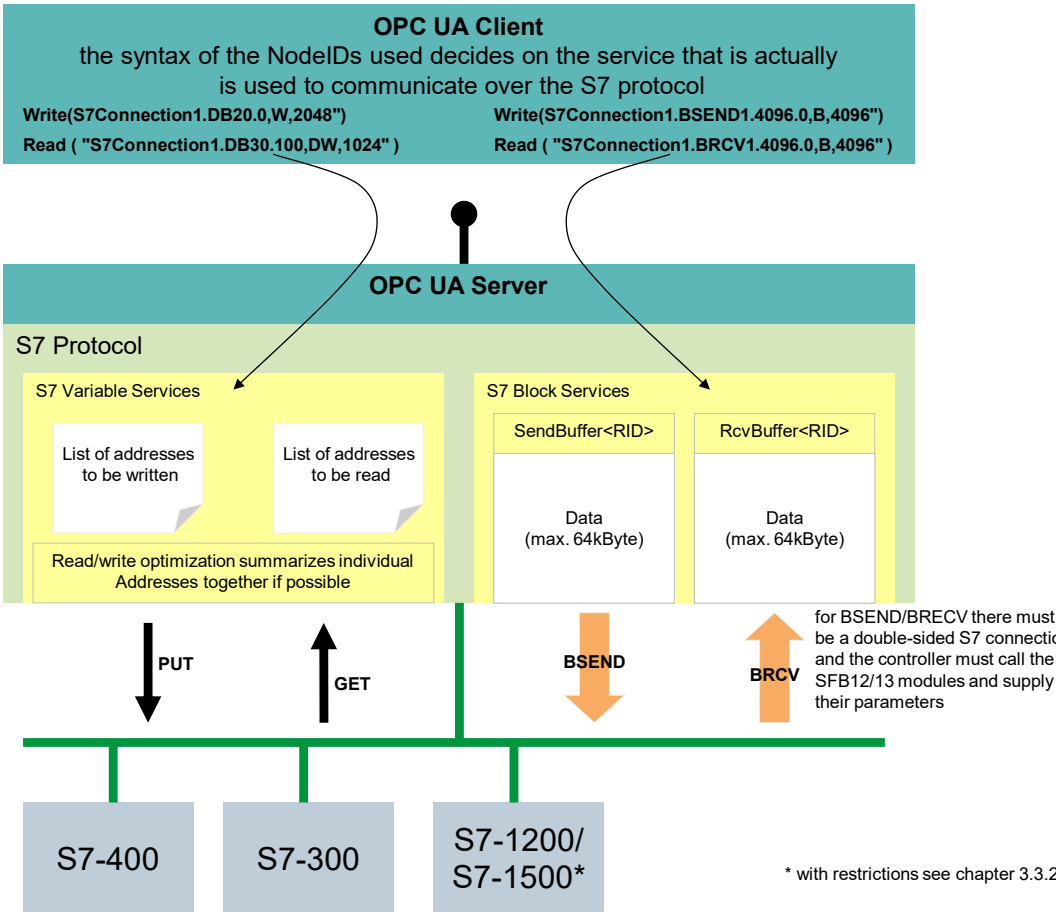
3.3.1. General

This article describes how to access S7 controllers using the SIMATIC S7 protocol via the UA server.

Chain of effects of communication

S7 communication is divided into two fundamentally different communication services, variable services and block services. These are almost completely obscured at the level of the OPC UA server. Only the NodeId can be used to determine which communication service is used for control. The namespace "S7:" specifies that this is the new addressing type, which is based on the syntax of anypointers. Especially with a large number of variables, it offers much more performant access. For compatibility with the former syntax of the ItemIDs in classic OPC Data Access, the old structure of the identifiers under the namespace "S7COM:" is retained.

The OPC UA server internally breaks down the NodeId into its components and recognizes from its structure which communication service must be used to communicate with the S7 controller. Here, the connection name identifies the communication partner (this name represents, among other things, an IP address) and the keyword "BRCV" or "BSEND" causes the use of block services instead of variable services. The S7 type identifier and offset address indicate the position of the data within the controller, and the data type determines the interpretation of this data.



Variable Services

An S7 controller responds to requests via variable services, for which only a one-sided connection is required. Each S7 controller is a so-called "S7 server" and answers PUT/GET requests without having to implement anything in the PLC control program. All data areas of the controller can be accessed directly (E, A, M, DB, etc.). This communication service is very flexible and, above all, easy to use.

© Siemens 2025 |

NodeIds for Variable Services

Syntax in namespace S7:

<connectionname>. <S7object>. <address>{,{<S7type>}{,<quantity>}}

Example: S7-Verbindung_3.DB10.20,W

A variable of type word (16 bits unsigned) that is located in data module 10 and starts at the offset byte address 20 (i.e. consists of bytes 20 and 21). This variable is fetched with Put/Get via the connection named "S7-Verbindung_3", i.e. from the S7 control behind this connection.

Symbolic NodeIDs

In addition to direct addressing via the new syntax ("S7:"Namespace") and the old compatible addressing (S7COM:-Namespace), there is the possibility 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 that are connected to an OPC server via an S7 connection, a symbol export can be initiated. The resulting symbol file with the extension ATI is made known to the OPC UA server via download from STEP 7 or via XDB import. The ATI file (Advanced Tag Information) contains an illustration of the symbolic names to the direct addresses.

Note Symbolic addressing in the fast, highly optimized ATI variant is only available for variable services of the S7 protocol. In other words, all symbols are ultimately fetched from the controls via PUT/GET. Symbols that represent a BSEND or BRCV variable are not possible.

Block Services

The more effective block service is available for the exchange of large amounts of data. Large amounts of data (up to 64kByte) can be exchanged on a connection configured on both sides. Communication is based on the exchange of data buffers. However, the corresponding system function modules (BSEND/BRECV) must be called up in the control program. The OPC UA server provides the corresponding counterparts on the PC when the corresponding NodeIds (formerly OPC items) are created.

Structure of the NodeIds for block services

Syntax BRCV in namespace S7:

<connectionname>.brcv<rid>.<offset-within-in-the-buffer>{,{<S7type>}{,<array-size>}}

Example: S7-Verbindung_5.brcv3 or byte array example: S7-Verbindung_5.BRCV31.0,b,4096

The complete receive buffer for the BSEND/BRECV pair with ID 3, which is connected via the connection with the name "S7-Verbindung_5", is mapped into a ByteString for OPC UA. This ByteString always contains the last data sent by the communication partner (on the other side of "S7-Verbindung_5") with BSEND. On an S7 connection, there can be several BSEND/BRECV pairs that belong together and are connected via their RID. Here it is the BRECV, which belongs to the BSEND with the ID 3.

Syntax BSEND in namespace S7:

<connectionname>.bsend<rid>.<bufferlength>,< offset-within-the-buffer >{,{<S7type>}{,<quantity>}}

Example: S7-Verbindung_2.bsend1.1024,100,W,20

When writing to this NodeId, an array of words (unsigned integer 16 bits) with 20 elements is written from the byte offset address 100 to the send buffer of length 1024. In the 1024 byte buffer, the range from 100 to 140 is overwritten. The entire block is sent with ID 1 to the communication partner, who must provide a BRECV with ID 1 and a minimum length of 1024 bytes to receive the data.

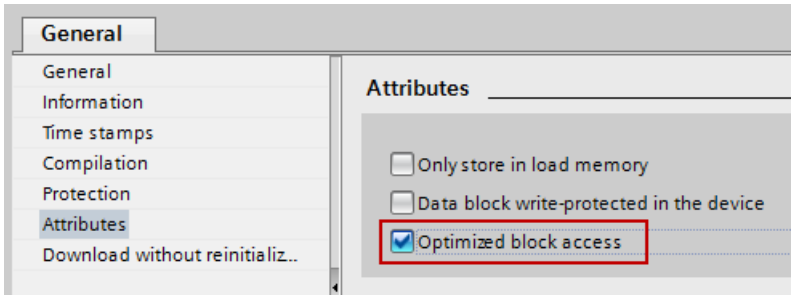
Note The use of the BSEND/BRCV block services requires a connection configured on both sides. The controller must independently call up the SFB12/13 modules and supply their parameters.

On the subject of block-oriented services, please also refer to the information in the SIMATIC NET manual "Industrial Communication – Volume 2 - Interfaces".

3.3.2. Optimized S7 communication

Background

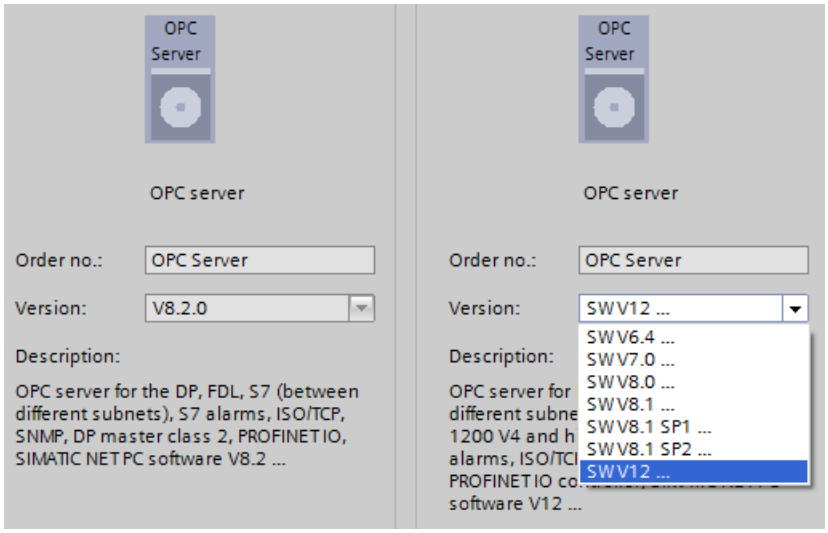
With an S7 connection, access to optimized data modules in the S7-1200/S7-1500 CPUs is not possible. In order to retrieve data from an S7-1200/S7-1500 with an OPC server via an S7 connection, the data modules from which the data is to be read must not be optimized.



Disadvantage: The performance of the innovative S7-1200/S7-1500 controllers is impaired by the use of non-access-optimized data modules.

Remedy

In order to avoid performance losses in the S7-1200/S7-1500 controllers due to data modules that are not access-optimized, the so-called "optimized S7 communication" is available from SIMATIC NET OPC V12 onwards.



This type of connection is created automatically when using the SIMATIC NET OPC Server V14 and an innovative controller. With the help of the optimized S7 connection, access to access-optimized data modules is then also possible.

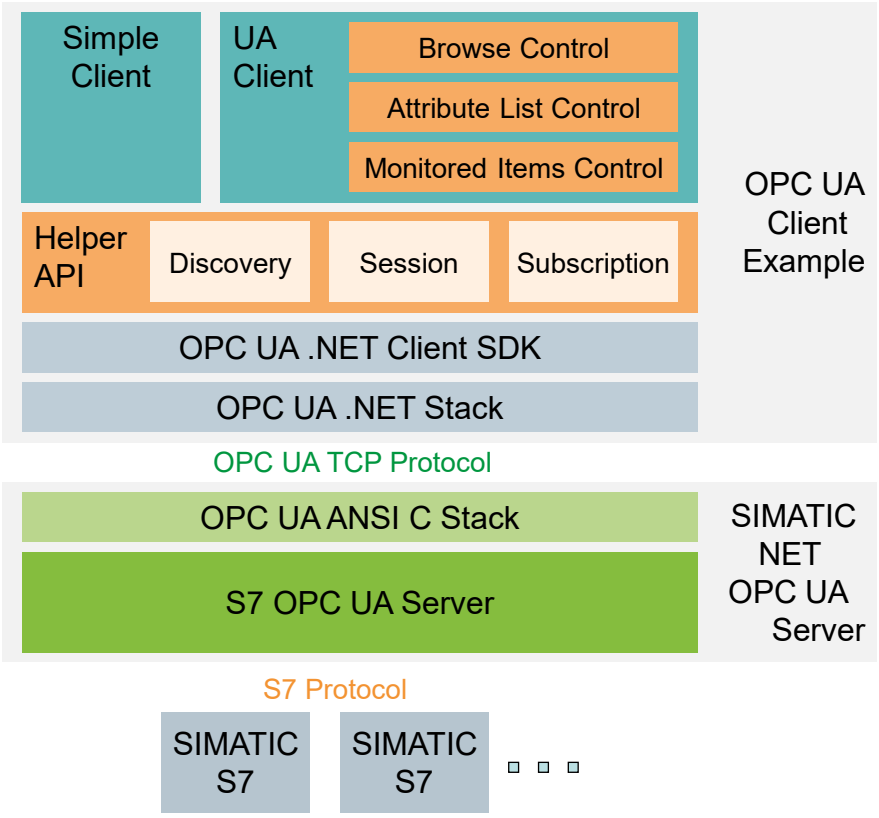
Access with OPC Client

The access point to the optimized S7 connections is via the SimaticNET.S7OPT server on port 50101.

© Siemens 2025 |

4. Functional mechanisms of this application

Overview



Module	Description
OPC UA .NET Stack	The .NET based OPC UA communication stack from the OPC Foundation.
.NET Client SDK	The OPC Foundation's .NET based OPC UA Client SDK.
Helper API	Reusable and simplified .NET class. It offers functions for discovery, session and subscription handling.
Simple Client	Simple example of using the Client API with the functions Connect, Disconnect, Read, Write and Data Monitoring. This example also shows the direct addressing and handling of namespaces.
UA Client	Convenient OPC UA client with the functions Discovery, Connect, Disconnect, Browse, Read of all attributes, Write and Data Monitoring. General functions such as browsing, listing attributes, and monitoring of data variables are encapsulated in reusable controls. In this example, the symbolic variables can be browsed and used directly from the browser.
ANSI C UA Stack	The SIMATIC NET OPC UA Server uses the optimized and portable OPC UA ANSI C stack from the OPC Foundation.
S7 OPC UA Server	The SIMATIC NET OPC UA server implements the necessary server logic for sessions and subscriptions and the data connection to the S7 stations.

.NET Client SDK

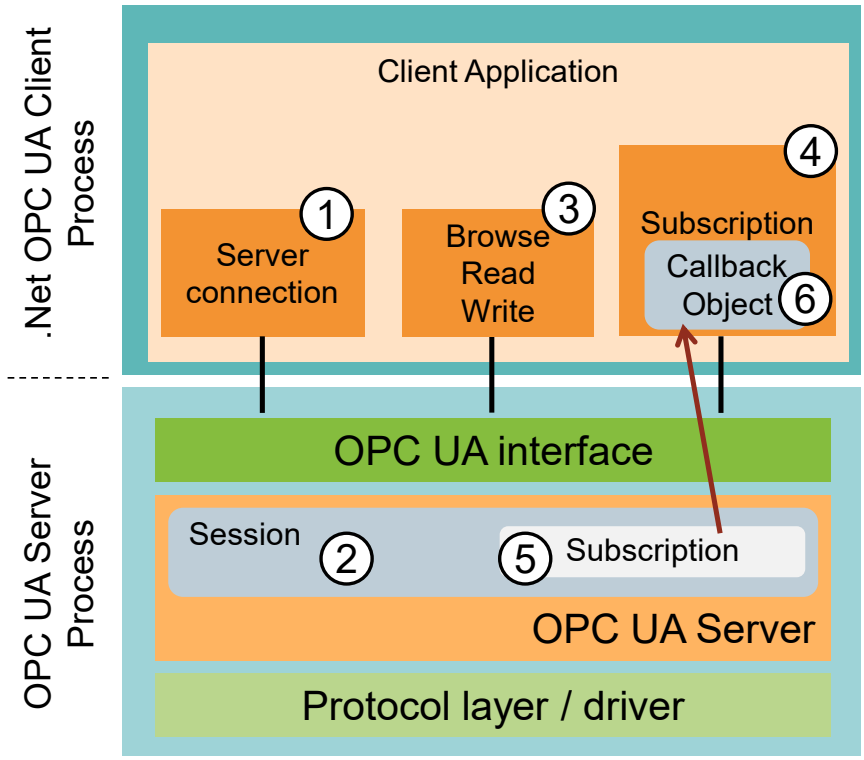
The OPC Foundation SDK is no longer maintained or supported by the OPC Foundation. This does not affect the functionality of the application example.

SDKs from commercial providers such as Unified Automation are recommended for use in your own applications.

© Siemens 2025

Program Overview

The following figure shows the function blocks in the OPC UA Client and the interaction with the OPC UA Server.



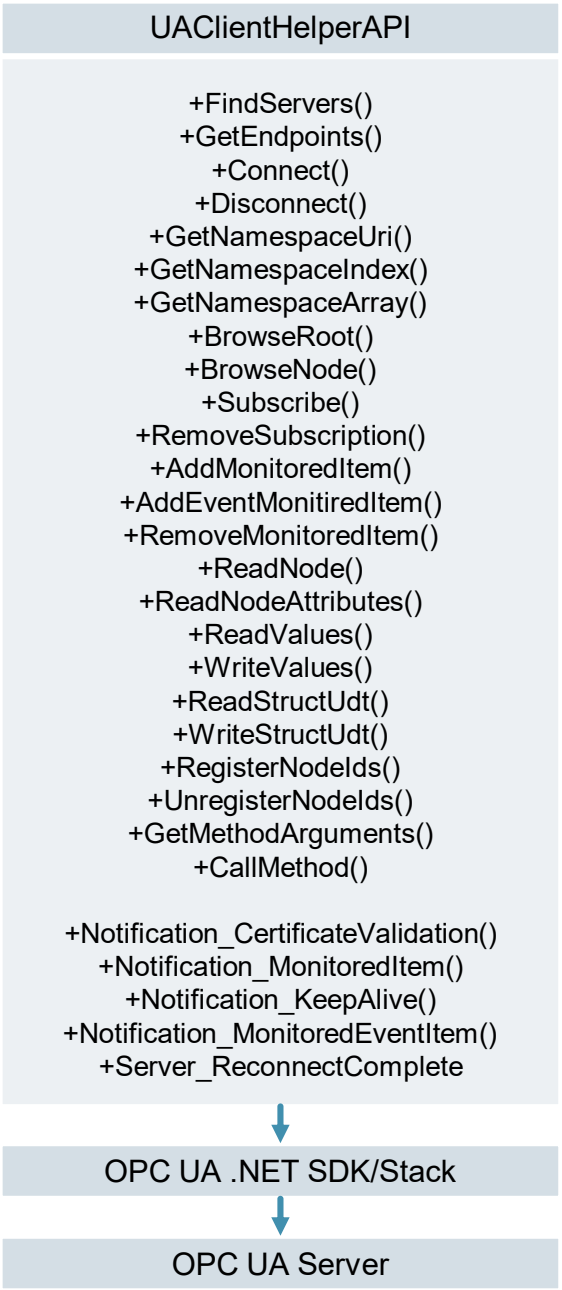
No.	Description
1.	When connecting the interface to the OPC UA Server, a Client API object is created on the client side. This object manages the connection to the server (2). It also provides all OPC UA services except for services related to a subscription.
2.	A session object is created in the server via the OPC UA interface.
3.	When establishing a connection between the interface and the OPC UA Server, the first level of the address space of the OPC UA Server is also displayed. The Browse Service of the OPC UA interface is used for this. If you select a node in the tree view of the Browse Control in the client, the attribute values of the node are displayed in another window using the Read Service.
4.	When registering variables to monitor value changes, a Subscription object is created that provides all OPC UA services related to a subscription.
5.	Via the OPC UA interface, a subscription object is created in the server that manages all subscription-relevant settings.
6.	In order to be able to receive value changes from the server, a callback connection is established. To do this, a SubscriptionCallback object is created in the client and connected to the subscription in the server. If changes are sent from the server to the client, the client enters the changes in the monitoring window.

© Siemens 2025

4.1. OPC UA Client Helper API

The following figure shows the class diagram for the UAClientHelperAPI class. The most important access methods to an OPC UA server are encapsulated and summarized in this class.

The UAClientHelperAPI accesses the . NET assemblies Opc.UA.Client.dll and Opc.UA.Core.dll the OPC Foundation.



© Siemens 2025 |

Method description

The following table explains the functions of the public methods within the UAClientHelperAPI class, which are used to implement the OPC UA Client functionalities:

Method	Description
FindServers	Searches for OPC UA servers on the network. Prerequisite: An LDS (Local Discovery Server) or GDS (Global Discovery Server) must be available.
GetEndpoints	Identifies the available endpoints on a server that can be used to connect.
Connect	Establishes the connection to a server and creates a secure channel and a session to the server.
Disconnet	Ends an existing session and disconnects from the server.
BrowseRoot	Returns a collection of nodes that can be found in the root directory of the server.
BrowseNode	Returns a collection of nodes that can be found hierarchically under a specific node.
BrowseByReferenceType	Returns a collection of nodes that can be found under a specific node under the specification of a reference type.
GetNamespaceUri	Returns the namespace uri to a passed namespace index.
GetNamespaceIndex	Returns the namespace index to a given namespace Uri.
GetNamespaceArray	Returns the namespace array.
Subscribe	Creates a subscription on the server.
RemoveSubscription	Deletes a specific subscription from the server.
AddMonitoredItem	Adds a MonitoredItem to an existing Subscription for observation.
AddEventMonitoredItem	Adds an (Event)MonitoredItem to an existing Subscription for observation.
RemoveMonitoredItem	Deletes an existing MonitoredItem of a subscription.
ReadNode	Reads the metadata of a specific node.
ReadNodeAttributes	Reads the most important attributes of a specific node.
ReadValues	Reads the values of variable nodes.
WriteValues	Writes values to variable nodes.
ReadStructUdt	Reads values of user-defined structures and UDTs using the server's "TypeDictionary".
WriteStructUdt	Writes values to user-defined structures and UDTs that were previously read.
RegisterNodeIds	Registers node ids with the server for optimized access to them.
UnregisterNodeIds	Deletes the registration of already registered node ids.
GetMethodArguments	Determines the available input and output arguments of a method.

© Siemens 2025

Method	Description
CallMethod	Calls a method on the server.
Notification_CertificateValidation	Event that is fired if a server certificate cannot be accepted.
Notification_MonitoredItem	Event that is fired if the values of a MonitoredItem have changed.
Notification_KeepAlive	Event that is fired when a KeepAliveNotification arrives.
Notification_MonitoredEventItem	Event that is fired when an OPC UA event arrives.
Server_ReconnectComplete	Event that is fired when a reconnect event arrives.

Note Not all available PLC data types can be converted in this example. This limitation affects the ReadValues, WriteValues, ReasStructUdt, WriteStructUdt, and Subscription methods.

© Siemens 2025 |

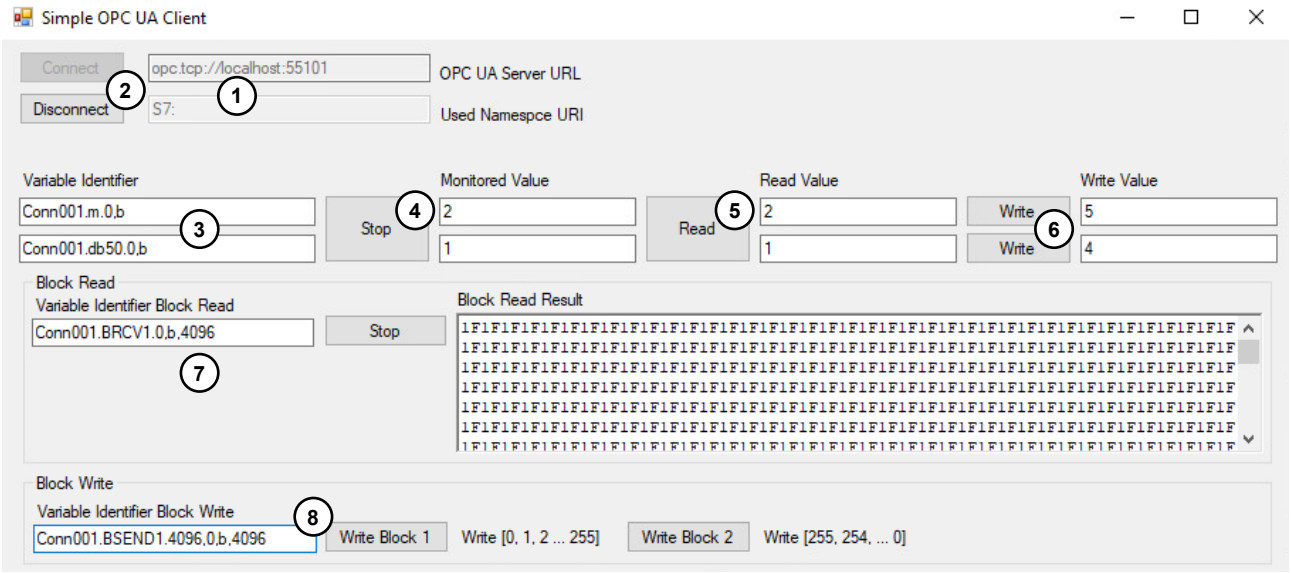
4.2. Simple OPC UA Client

The Simple Client is a simple example of how to use the Client API. The most important functions such as connect, disconnect, read, write and monitoring of data are shown in a file or class with a dialog. The code for the example can be found in the MainForm.cs file in the SimpleClient project.

S7 communication between S7 CPU and PC station (siemens.com)

Surface of the simple example

The interface is operated via buttons for the individual functions.



No.	Description
1.	The server URL can be specified in the text box for the server URL. For the SIMATIC NET OPC Server, this is composed of opc.tcp://<name of computer>:55101. The Namespace URI text box specifies the namespace used. This is S7: for direct addressing, S7COM: for direct addressing via OPC DA compatible syntax and SYM: for symbolic addressing.
2.	The Connect and Disconnect buttons can be used to establish and disconnect the connection to the OPC UA server. The connection is established exclusively without security. The secure connection setup is explained in the next example.
3.	In the text boxes for the variable identifier, the identifier part of the NodeId is specified. This is the case with the namespace S7: e.g. composed of <S7Connection>.<data set>. <Offset>,<data type> The NodeId for reading and writing is formed from the identifier and the namespace index. The namespace index results from the position of the namespace in the namespace table of the server. This table can be read by the server with Read.
4.	A subscription is created via the Monitor button and two monitored items are created in the subscription with the two NodeIds. The data changes are displayed in the text boxes next to the button. Errors are displayed instead of the values.
5.	The Read button reads the values (Value Attribute) of the two variables with the specified NodeIds and displays them in the text boxes next to the button.
6.	The Write button writes the value from the text box next to the button to the variable identified by the NodeId. To write, Read must first be called, as the text from the text box must be converted into the appropriate data type of the variable. The conversion is done on the basis of the data type supplied by Read.
7.	In the "Block Read" group, data can be received that is actively sent by the S7 with the block service BSEND. This can be used, for example, to send 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 received there with the BRECV block service. Two blocks with different contents can be sent. This can be used, for example, to download recipe data for the S7.

© Siemens 2025

Functions of the simple example

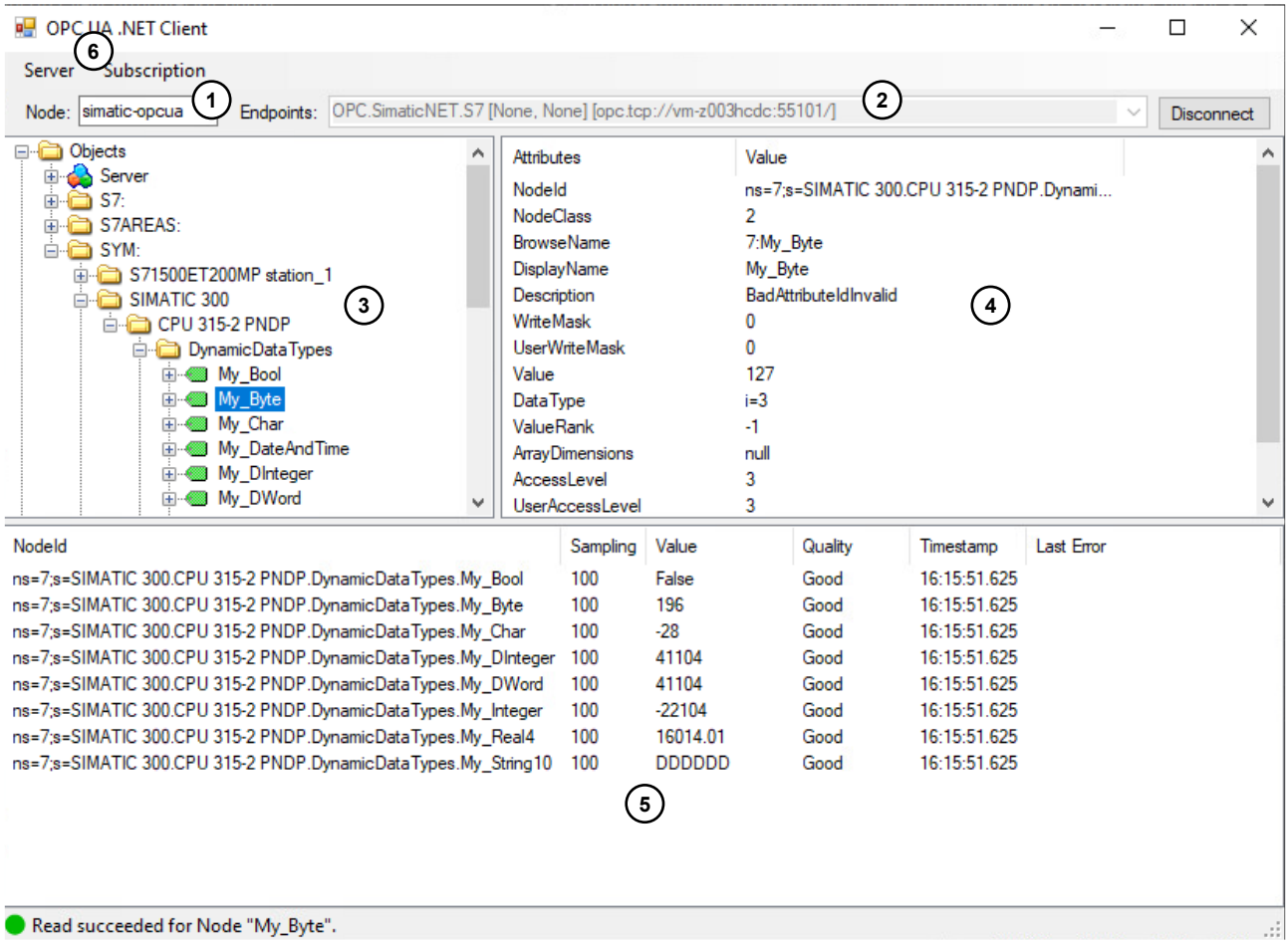
The functions can be found in the MainForm class in the MainForm.cs file. Simple error handling is implemented in the functions. If an exception occurs during OPC UA calls, a dialog with the error message is displayed. If an error occurs with one or more variables during the variable-related calls, the error is displayed in the corresponding text boxes.

Function	Description
btnConnect_Click	In this function, the connection to the OPC UA Server is established via the Server::Connect() function of the Client API. The URL string is passed from the corresponding text box. Once the connection has been successfully established, the namespace table is read by the server via the Server::ReadValues() function. In the returned table, the namespace is searched for from the Namespace URI textbox. The index in the table is stored in a class variable.
btnDisconnect_Click	In this function, the connection to the OPC UA Server is disconnected via the Server::D isconnect() function of the Client API
btnRead_Click	In this function, the two NodeIds are first formed from the namespace index and the identifier texts. After that, the two values are read via the Server::ReadValues() function. The result is written to the corresponding text box. The result can be either the read value or an error code.
btnMonitor_Click	If no subscription is created, a subscription is first created in this function with Server::Subscribe(). After that, the two NodeIds are formed from the namespace index and the identifier texts. Then the two monitored items are created with Subscription::AddMonitoredItem(). The corresponding text box for the output of the values is passed as a client handle. If a subscription has been created, it is deleted with Server::RemoveSubscription().
ClientApi_ValueChanged	The function is specified in Subscription::AddMonitoredItem() as a callback function. The function first checks whether the call arrives in the main thread of the dialog. If not, the call is passed to the main thread of the dialog via BeginInvoke. Otherwise, access to the dialog is not possible. Then it checks whether the client handle is a textbox. If not, it is a callback for the Read block. If it is the value of a normal variable, the client handle is cast back onto a text box and, after checking the status, either the value or an error code is entered into the text box. If, on the other hand, it is the value of a block variable, the byte array is extracted and output as a sequence of HEX values for the individual elements of the byte array.
btnWrite1_Click	The function composes the NodeId for variable 1 and calls the writeNewValue function with the NodeId, the new value as text from the text box, and the last value read.
btnWrite2_Click	The function composes the NodeId for variable 2 and calls the writeNewValue function with the NodeId, the new value as text from the text box, and the last value read.
writeNewValue	In this function, the new value of a text is first converted to the data type that was returned when it was last read. Then the value is written via the Server::WriteValues() function and the result is checked.
btnMonitorBlock_Click	If no block subscription is created, a subscription is first created in this function with Server::Subscribe(). Then the NodeId is formed from the namespace index and the identifier text. The monitored item is then created with Subscription::AddMonitoredItem(). If a subscription has been created, it is deleted with Server::RemoveSubscription().
btnWriteBlock1_Click	The function composes the NodeId for the block variable and simulates record 1. The simulation creates a byte array with the length specified in the interface and fills the values with a variable value that starts at 0 and is incremented after each assignment. After that, the writeNewBlockValue function is called with the NodeId and the new value.
btnWriteBlock2_Click	The function is identical to btnWriteBlock1_Click; however, the variable value assigned here starts with 255 and is decremented after each assignment.
writeNewBlockValue	In this function, the value is written via the Server::WriteValues() function and then the result is checked.

4.3. Convenient OPC UA Client

4.3.1. Surface

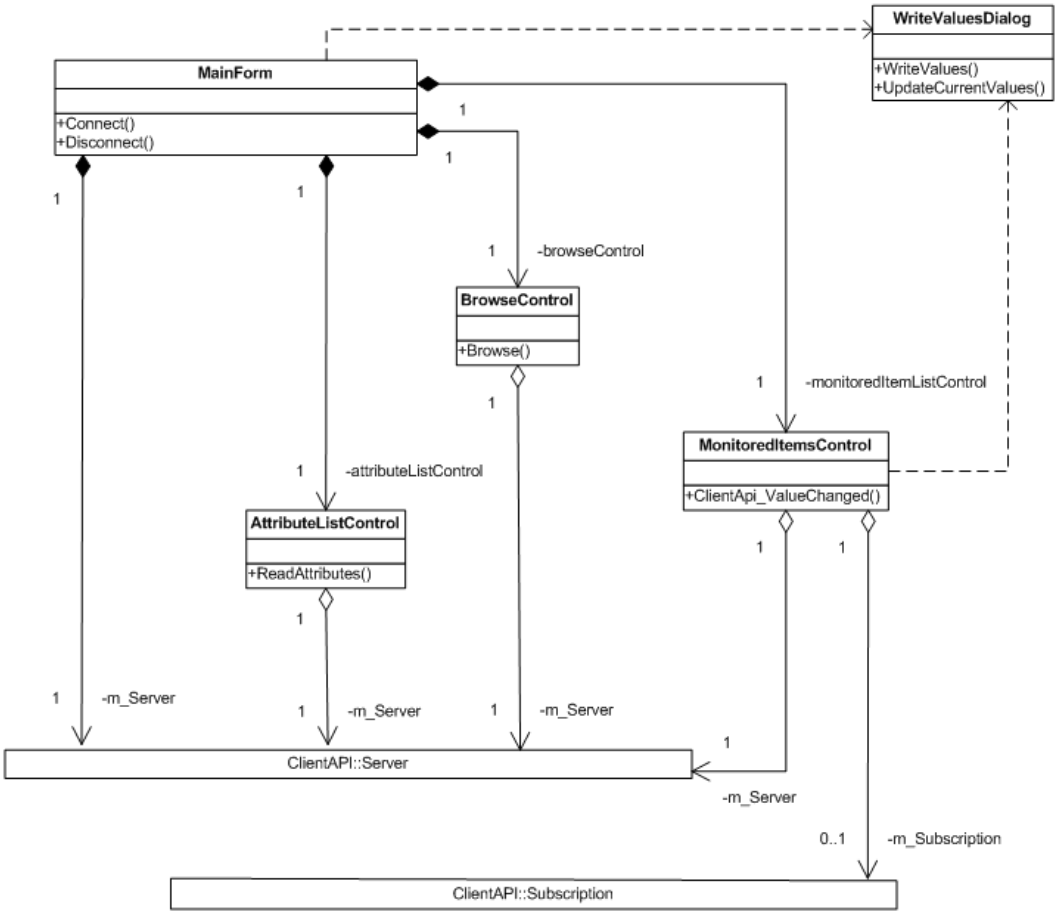
The following figure and table describe the interface of the generic OPC UA Client example, which can be used to conveniently access the information of the namespace of an OPC UA server.



No.	Description
1.	The server can be selected via the Endpoints drop-down list. For this purpose, the list of available OPC UA servers is determined by the respective network node. The computer on which the list is to be queried can be entered in the Node text field. If the field is empty, the list is determined on the local computer. The URL of the OPC UA server can also be entered manually. The URL for the SIMATIC NET OPC UA server is composed of opc.tcp://<Rechnername>:55101.
2.	The Connect button can be used to establish or disconnect the connection to the server.
3.	In Browse Control, the entire address space of the connected server is displayed in a hierarchical tree view. Only hierarchical references are displayed.
4.	For the selected node in the Browse Control, the attributes are read and displayed in this control.
5.	With Drag&Drop, variables can be dragged from the Browse Control into the observation window. For the variable, the NodeId, the sampling interval, the value, the timestamp and the status code are then displayed.
6.	The properties of the subscription and monitored items can be changed via the context menu in the observation window or via the application menu. For example, the sampling interval can be changed. In addition, the dialog can be opened for writing. The variables that are selected in the observation window are transferred to the dialog.

4.3.2. Class diagram

The following class diagram shows the classes of the OPC UA sample client. These classes realize the functionality of the interface and use the UA Client API classes. On the following pages, the individual classes are explained in detail.



Class MainForm

The MainForm class described in the following table implements the functionality of the main dialog of the client application.

The class name corresponds to the file name in the UAClient project.

Method	Functionality
MainForm	Constructor of the class.
Connect	Implements the functionality to establish the connection with the server and initializes the Browse Control.
Disconnect	Disconnects from the server.

Class BrowseControl

The BrowseControl class, described in the following table, implements the functionality for displaying the server's address space.

The class name corresponds to the file name in the UAClient project.

Method	Functionality
BrowseControl	Constructor of the class.
Browse	Navigates through the tree view starting from the selected node.

© Siemens 2025

Class AttributeListControl

The AttributeListControl class, described in the following table, implements the display of the values of all attributes of a node in the address space.

The class name corresponds to the file name in the UAClient project.

Method	Functionality
AttributeListControl	Constructor of the class.
ReadAttributes	Reads the values of a node's attributes and displays them in a list.

Class MonitoredItemsControl

The MonitoredItemsControl class described in the following table implements the creation of a subscription and MonitoredItems and the display of the values and status of the monitored items.

The class name corresponds to the file name in the UAClient project.

Method	Functionality
MonitoredItemsControl	Constructor of the class.
ClientApi_ValueChanged	Callback function for the Data Change Events from the Client API. The new value, timestamp and status will be entered in the corresponding row. To do this, the client handle is cast onto the object that represents the line.
MonitoredItems_DragDrop	In this function, a NodeId, which is dragged and dropped onto the control, is created as a Data Monitored Item. If there is no subscription yet, a subscription is created first.

Class WriteValuesDialog

The WriteValuesDialog class, described in the following table, implements the display and modification of the current values of one or more variables.

The class name corresponds to the file name in the UAClient project.

Method	Functionality
WriteValuesDialog	Constructor of the class.
WriteValues	Writes the value of one or more variables.
UpdateCurrentValues	Updates the values in the dialog by calling Read.

Using the Client API in the Example

The following table lists the files and functions in which the Client API is used.

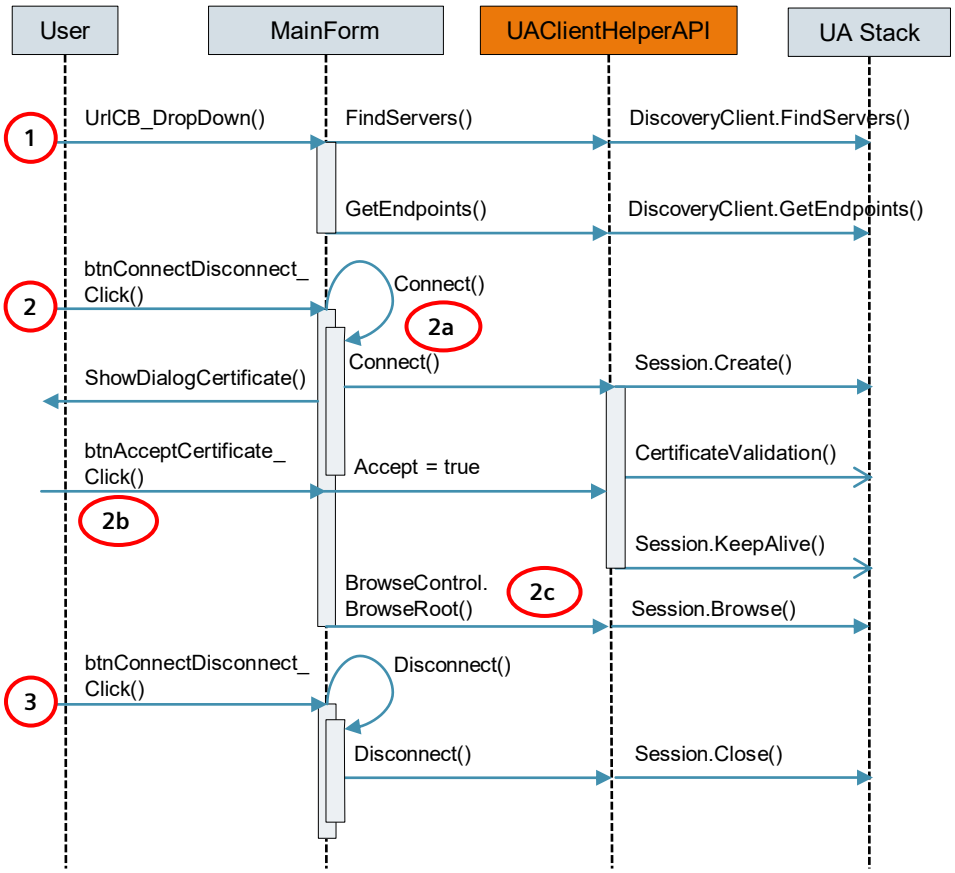
Client API	Used in
Class Discovery	
FindServers	MainForm.cs in the function UrlCB_DropDown
GetEndpoints	MainForm.cs in the function UrlCB_DropDown
Class Server	
Connect	MainForm.cs in the Connect function
Disconnect	MainForm.cs in the Disconnect function
Browse	BrowseControl.cs in the Browse function
Read	AttributeListControl.cs in the ReadAttributes function
ReadValues	WriteValuesDialog.cs in the UpdateCurrentValues function
WriteValues	WriteValuesDialog.cs in the WriteValues function
AddSubscription	MonitoredItemsControl.cs in the function MonitoredItems_DragDrop
ModifySubscription	MonitoredItemsControl.cs in the function MonitoringMenu_PublishingInterval_Click
RemoveSubscription	MonitoredItemsControl.cs in the RemoveSubscription function
Class Subscription	
AddDataMonitoredItem	MonitoredItemsControl.cs in the function MonitoredItems_DragDrop
ModifyMonitoredItem	MonitoredItemsControl.cs in the function MonitoringMenu_SamplingInterval_Click
RemoveMonitoredItem	MonitoredItemsControl.cs in the function MonitoringMenu_RemoveItems_Click

4.3.3. Sequence diagrams

Establishing and terminating the connection to the OPC UA server

The following sequence diagram shows the processes that are necessary to establish the connection to the OPC UA server. By clicking on the Combobox Endpoints, the user first selects an available endpoint.

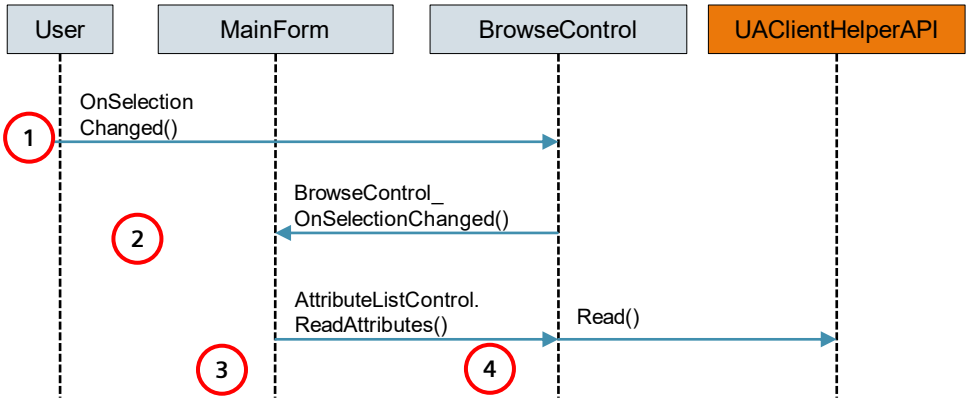
The connection can be started via the Connect button in the interface or via the Server menu. After a successful connection has been established, the Disconnect button will be displayed. The sequence diagram also shows the processes that are triggered by the "Disconnect Server" action via the Disconnect button.



No.	Description
1.	When the selection list is opened, a ClientAPI::DiscoveryClient object is created and the methods FindServers and GetEndpoints are called there.
2.	The Connect user action invokes the ConnectDisconnect_Click method on the MainForm object. In this method, the actions to establish a connection to the server are performed. In the first step, the ClientAPI object is created and method Connect is called there (2a). An authentication mask opens in which credentials or, if configured, Anonymous access can be activated. The connection to the OPC UA server - defined by the URL - is then established. If a certificate is not trusted, the user can still accept it and the connection will continue (2b). In the case of an existing connection, the second step is to call the BrowseRoot method on the ClientAPI object (2c). This will display the first level of the address space of the OPC UA server in the BrowseControl.
3.	The Disconnect Server user action also calls the ConnectDisconnect_Click method on the MainForm object. In the case of an existing connection, this method calls the Disconnect method on the ClientAPI object. This terminates the connection to the OPC UA server.

Reading attributes

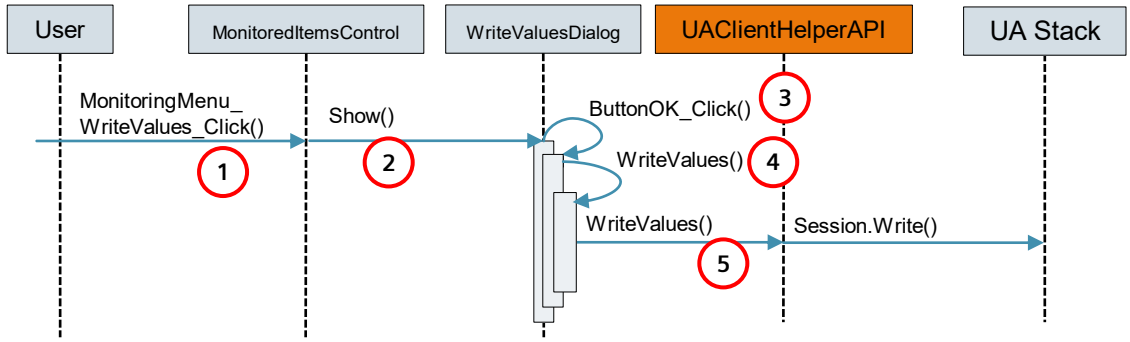
The following sequence diagram shows the processes involved in reading attribute values.



No.	Description
1.	If a node in the tree of the BrowsControl is selected, an OnSelectionChanged event is triggered.
2.	The event is forwarded to the MainForm.
3.	The event calls the ReadAttributes method of the AttributeListControl.
4.	There, Read is called on the ClientAPI object and the values are displayed.

Writing Values

The following sequence diagram shows the processes involved in writing values of variables.

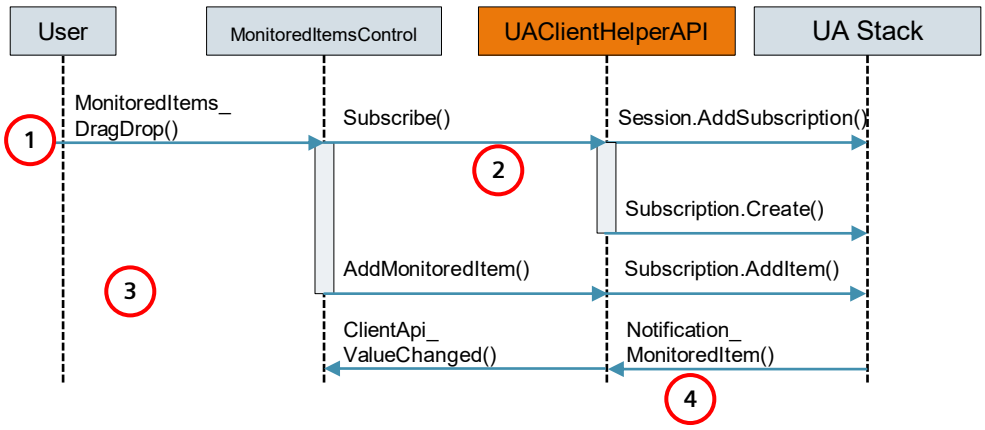


No.	Description
1+2	Clicking on the "WriteValues" entry in the context menu of MonitoredItemsControl (1) opens the "Write Values" dialog (2). The current values of the variables previously selected in the MonitoredItemsControl appear there.
3+4	If the user presses the Ok button (action buttonOk_Click) after entering new values, WriteValues is called on the WriteValuesDialog object.
5	There, WriteValues is called on the ClientAPI object.

© Siemens 2025 |

Registering variables for observation

The following sequence diagram shows the processes involved in registering variables for the observation of value changes.



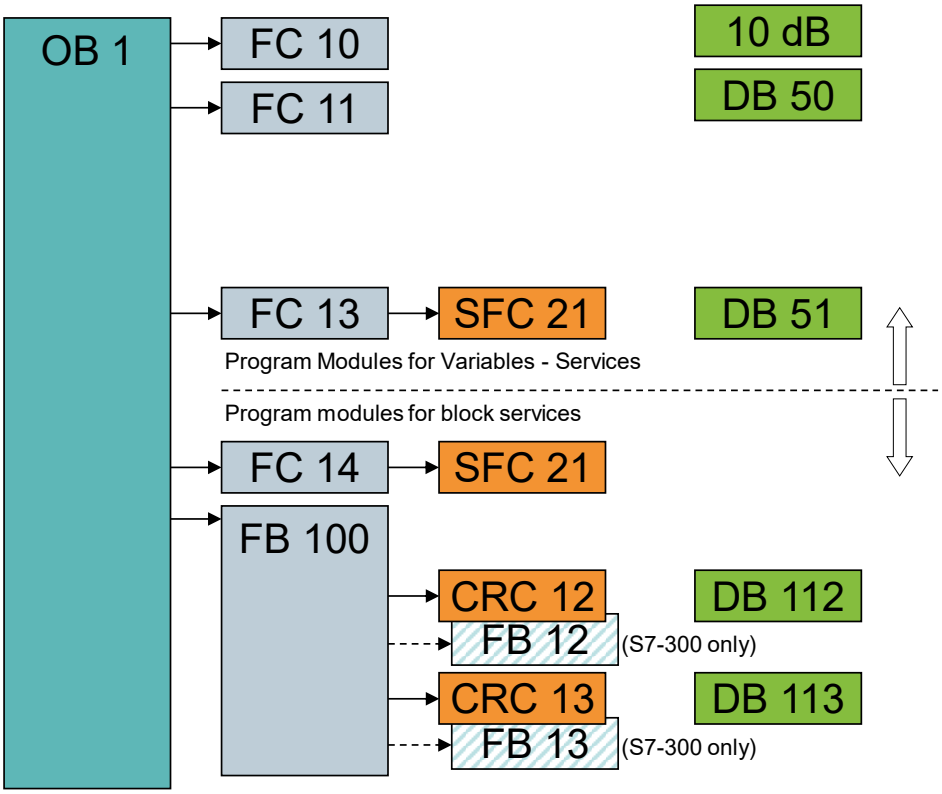
No.	Description
1.	A drag-and-drop action with the MonitoredItemsControl as the target first checks whether a subscription already exists. If not, Subscribe is called at the Client API.
2.	A Subscription object is created and then a Create is called on the Subscription object.
3.	The next step is to call AddMonitoredItem on the Client API object.
4.	From now on, value changes are received by the server and forwarded to the MonitoredItemsControl to be displayed there.

© Siemens 2025 |

4.4. S7 Program

Overview

The S7 program is essentially divided into two parts. First, the dynamic data for the variable services is simulated, then the send data is simulated and the block services are called in FB 100.



ATTENTION The S7 program cannot be used together with optimized modules due to the absolute addressing used!

Note The S7 program for the S7-1500 only contains the data module "StaticDataTypesOpt" (DB1), in which variables are stored in an optimized way. The S7 program is only used to provide variable nodes for the S7 opt server (port 55105).

© Siemens 2025 |

Simulation of dynamic data

The following table gives a brief overview of the program parts and their function for data simulation. Details are deliberately omitted here, further comments can be found in the AWL code.

Building block	Note
OB1	Cyclic Main here a variable timer is first set, with the interval of which the other program functions are called. The data change speed can be set above DB10 bytes 0.
FC10	ChangeDateAndTime increments a date value and time in DB51.
FC11	ChangeSimpleTypes increments the data of the 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 variables for configuring the data simulation.
DB50	StaticDataTypes contains simple data types that have been given symbolic names. The values are pre-initialized with maximum value range final value.
DB51	DynamicDataTypes contains simple data types that have been given symbolic names. The values are incremented according to their value ranges using the FC10 through FC13 functions.
SFC21	FILL Auxiliary function to fill data ranges with values, memory initialization.

© Siemens 2025 |

Block-Oriented Data

The S7 communication on the S7-300 is realized via FBs (loadable function modules) and not on the basis of SFBs (integrated system function modules) as in the S7-400. However, if you call an SFB instead of FB in the S7 program of the S7-300, the device returns an "ERROR" and outputs the "STATUS = 27". This status indicates that the function module for S7 communication does not exist on the S7-300. The communication FBs for the S7-300 are located in the Task Card in the Instructions tab under Communication > S7 Communication.

The following table gives a brief overview of the program parts and their function in relation to BSEND/BRCV. Details are deliberately omitted here, further comments can be found in the AWL code.

Building block	Note
OB1	Cyclic Main Call of the transmitter module (SFB12) for BSEND and the receiver module (SFB13) for BRCV via function block 100. For S7-300, FB12 and FB13 from the library are used, as the S7-300 communicates via loadable FBs and not system functions.
FC14	ChangeSendData increments a byte, which is then copied to the entire send buffer
FB100 + DB100 (Instance DB)	InvokeBSENDandBRCV calls up the real communication modules and provides their parameters
DB112	SendData Data module with a length of 4096 bytes, to which the transmission module is passed
DB113	ReceiveData Data module with a length of 4096 bytes to which the receive module is passed
SFB12 + DB12 (instance DB)	BSEND The send module passes the send buffer to the communication processor (CP), which then sends it to the communication partner according to RID and connection ID.
FB12	BSEND (S7-300 only)
SFB13 + DB13 (instance DB)	BRCV The receive module fetches the last received data packet from the communication processor (CP) according to RID and ConnectionID and places it in the receive buffer.
FB13	BRCV (S7-300 only)
SFC21	FILL Helper function to fill data ranges with values, memory initialization.

The program logic sends or receives data blocks and supplies the corresponding parameters via the FB100. If larger data packets are sent, a multiple call to the BSEND or BRCV module is required. This multiple call is performed by the FB100.

Note If the transmit and receive modules are called cyclically, a high communication load can arise on the one hand and there is a risk that the data buffers will be overwritten before they can be processed by the other side. The program logic should therefore include flow control to ensure data consistency. For this purpose, the parameters DONE (done) and NDR (new data received) must be used.

In order to be able to exchange data between two S7-300 stations via a configured S7 connection, communication functions must be called in the S7 program. The FB12 "BSEND" module is used to send data and the FB13 "BRCV" module is used to receive data.

The S7 connection must be configured on both sides, as the S7 communication via FB12 "BSEND" and FB13 "BRCV" is based on the client-client principle.

© Siemens 2025

5. Configuration and configuration

5.1. Configuration of the SIMATIC S7 stations

General

It is assumed that all hardware and software components have been successfully installed and wired accordingly.

Here is an overview of the IP addresses used in the example project:

Address	Station	Note
192.168.0.200	PC OPC Server Station	
192.168.0.2	S7-300, CP 343-1	
192.168.0.3	S7-400, CP 443-1 EX/GX 20	optional
192.168.0.4	S7-1500, CPU 1516-3 PN/DP	optional
192.168.0.1	Laptop OPC Client Station	optional

Special feature S7-1200 from FW V4.0 and S7-1500

If you are using optimized data modules in an S7-1200 station from FW V4.0 or a S7-1500 station, then you must use the OPC Server V12 or higher to access the data modules of the controllers in an optimized way.

STEP 7 then automatically creates "optimized S7 connections", as standard access to optimized data modules via OPC is not possible.

The OPC client must then access the SimaticNET.S7OPT server via port 55105.

If you do not want to use optimized S7 connections, then the following conditions must be met:

1. Use of the OPC server at most in version 8.2.
2. Use of non-optimized data building blocks.

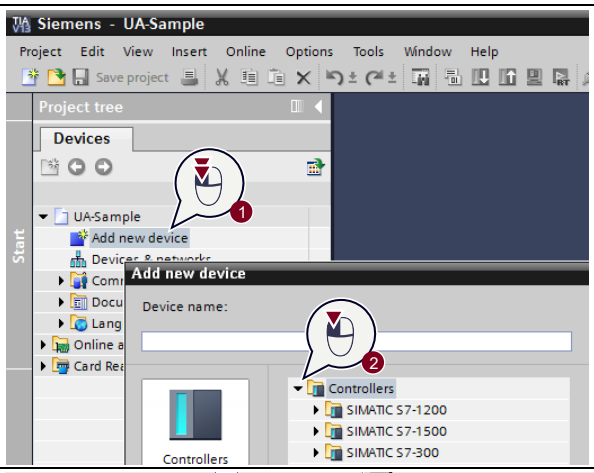
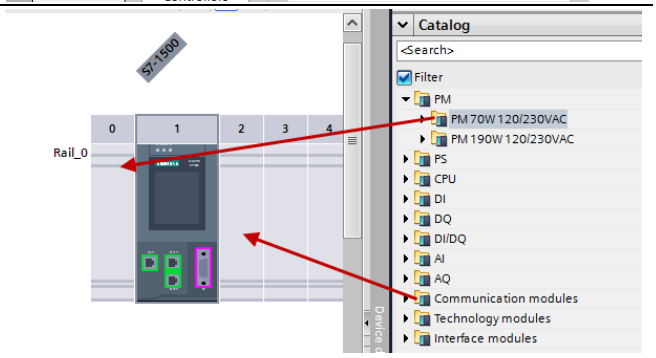
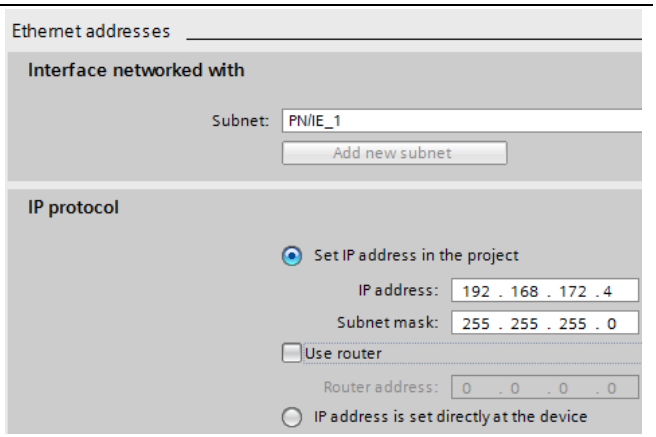
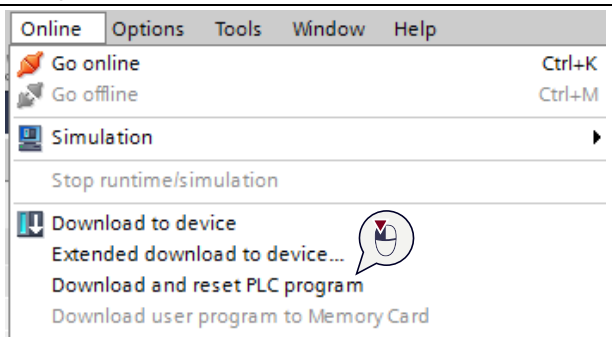
Procedure

The following configuration steps for the SIMATIC S7 stations describe the procedure as an example. Make adjustments on your own according to your hardware.

Note After saving and translating, all configuration and configuration information is overwritten.

© Siemens 2025 |

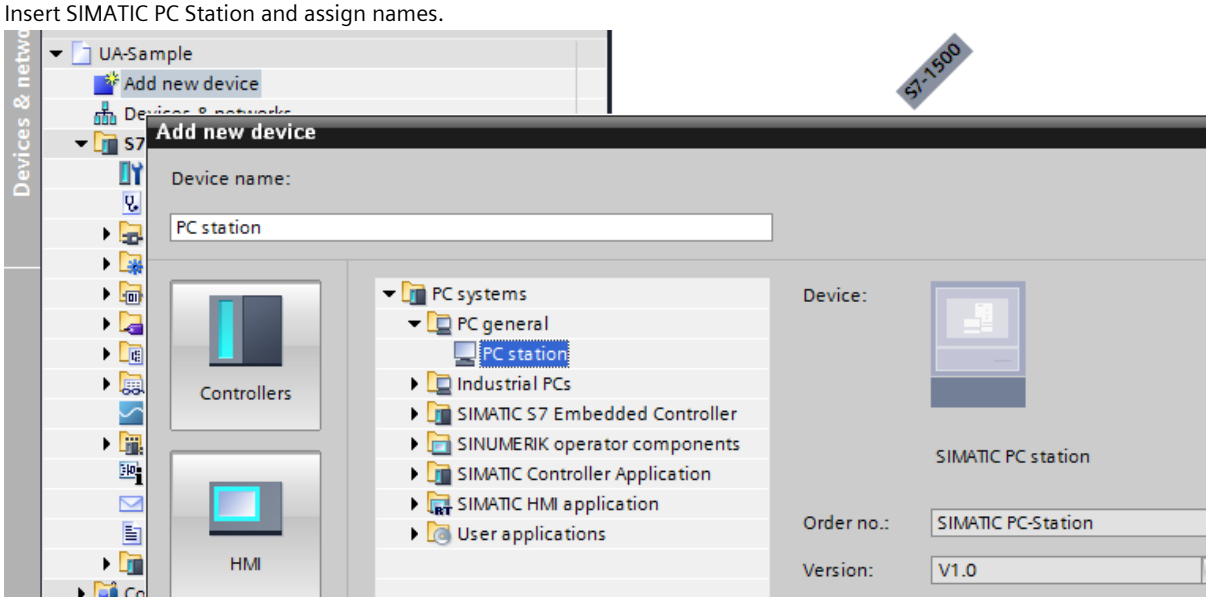
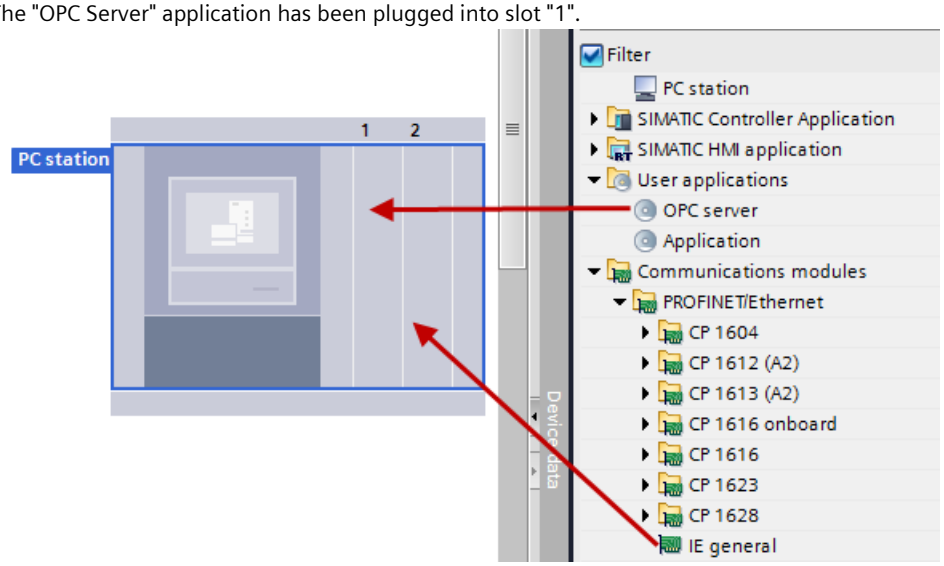
The following table shows the configuration and configuration of the SIMATIC S7 stations.

No.	Action	Note
1.	Open STEP 7 V20 and create a new project.	The name "UA sample" was used here.
2.	SIMATIC 400 Station and assign names (here "S7-400"). SIMATIC 300 station and assign names (here "S7-300"). Insert SIMATIC 1500 station and assign names (here "S7-1500").	
3.	Open SIMATIC stations in the device view and insert any CPs and other components used according to the hardware actually used.	
4.	To set the IP address, go to the Properties tab in the Inspector window. Select Ethernet addresses there. Setting the IP address and subnet mask. Create an Ethernet network. A MAC address is only entered if the station is to communicate via ISO Transport Layer 4.	
5.	Repeat the steps accordingly for all SIMATIC stations.	
6.	Creating S7 Connections	The connection configuration is described together with the configuration of the PC Station (in Chapter 6.2).
7.	When all settings are activated, load the configuration into the S7 CPUs with a restart.	

© Siemens 2025

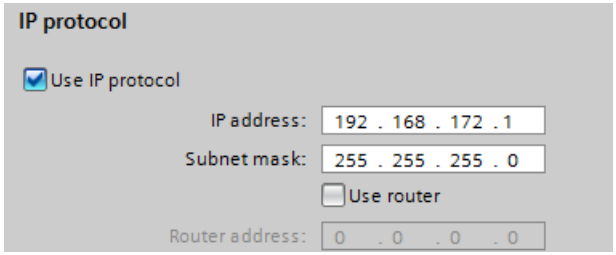
5.2. Configuration of the OPC Server Station

The configuration and configuration of the SIMATIC PC Station is carried out using STEP 7 and described step by step. Alternatively, project planning can also be carried out with the help of the NCM-PC software package. The procedure is identical, but unilaterally projected connections are used.

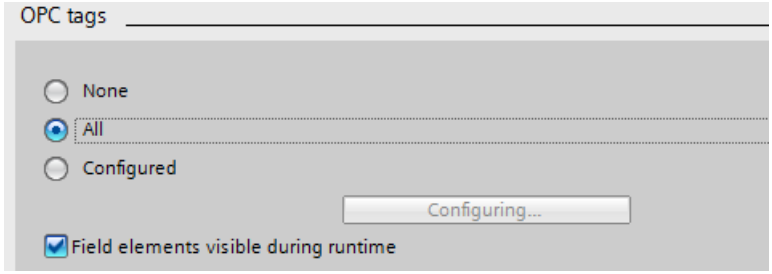
No.	Action
1.	Starting STEP 7 V20. Open the previously created "UA Sample" project.
2.	Insert SIMATIC PC Station and assign names. 
3.	Open the PC Station device view. Insert the Ethernet card and the OPC server there. Note If you want to use an S7-1200 or an S7-1500 CPU with optimized data modules, then you must use the OPC Server V12 or higher. The slot must be identical to the index assigned in the configuration console, in this case index "2" for Ethernet card. The "OPC Server" application has been plugged into slot "1". 

© Siemens 2025

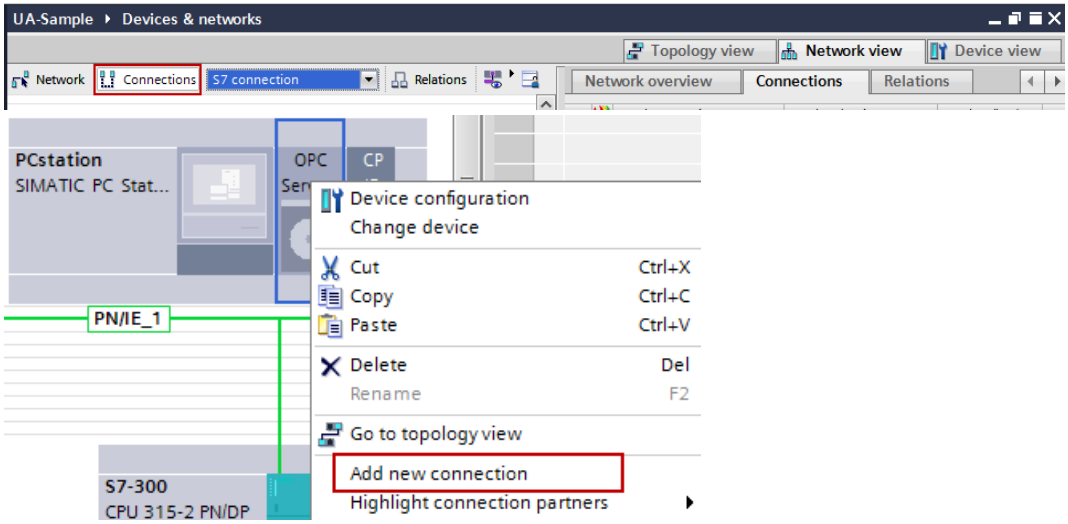
No.	Action
4.	An IP address (here "192.168.172.1") is assigned to the Ethernet card in the properties and the card is connected to the Ethernet network.

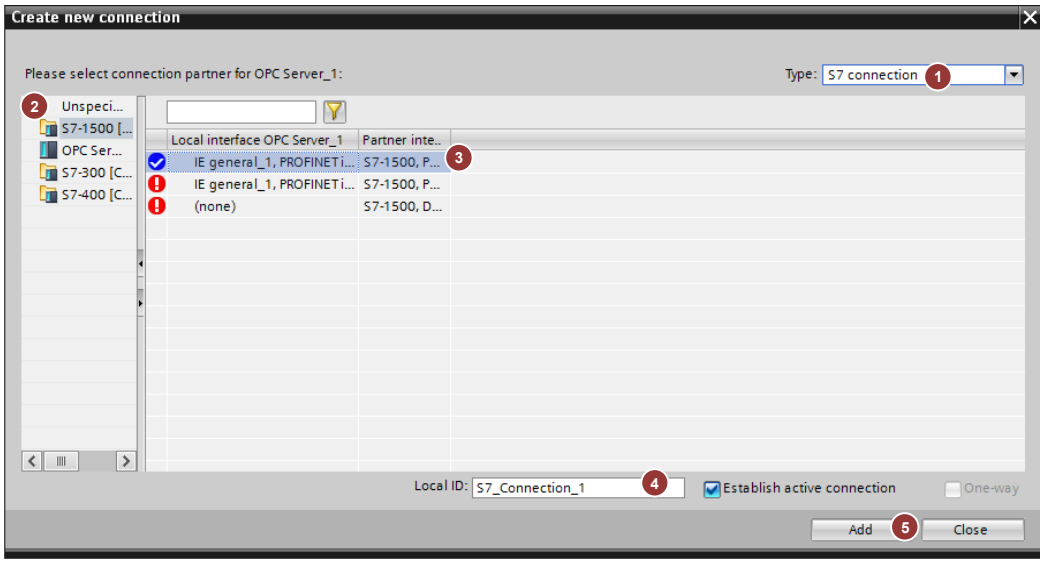


5.	In the properties pages of the OPC Server, the use of symbolic addressing is activated in the "S7" tab.

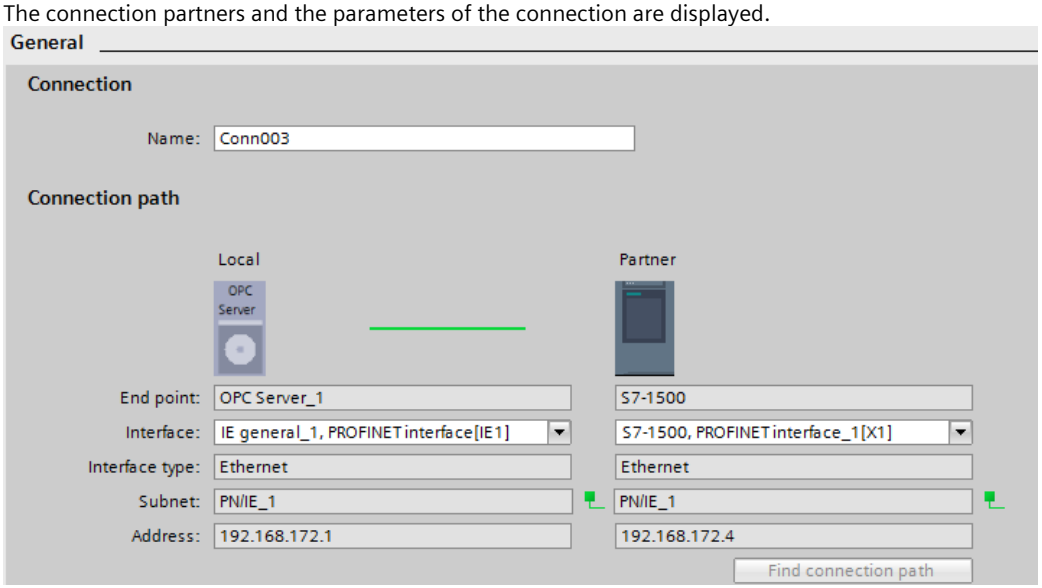
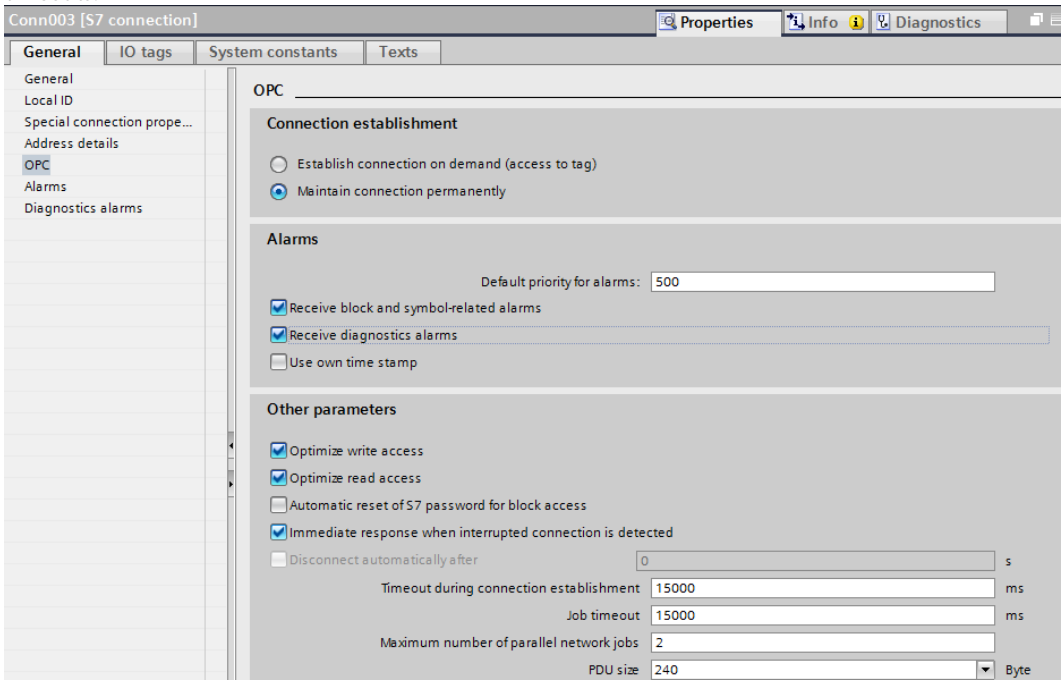


6.	In the network view, three S7 connections are created from the OPC server to the S7 controllers. The connections are created via Ethernet.





© Siemens 2025 |

No.	Action
7.	S7 connection via Ethernet: After the S7 connections have been created, further settings can be made by clicking on the connection and the Properties tab in the inspector window. After the connection path has been selected, the connection name can be changed (here "Conn001" for the connection to the S7-300, "Conn002" for the connection to the S7-400 and "Conn003"). The connection partners and the parameters of the connection are displayed. 
8.	In the properties of the S7 connections, the "OPC" menu is selected. Connection-specific settings are made here. The connection is permanently set so that the connection is maintained even when there is no communication. The connection is configured to transmit alarms and diagnostic events. Furthermore, the immediate response to connection interruptions is activated to avoid unnecessary waiting times for timeouts. 
9.	After the connections have been created, the new configuration must be loaded into the stations. The PC Station can also be loaded with the XDB file as described in Chapter 6.

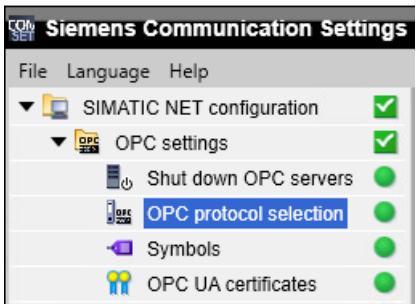
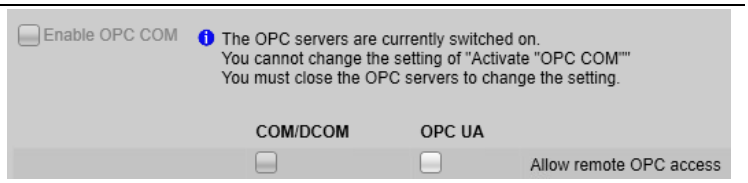
© Siemens 2025

5.3. Configuration of OPC UA Security

The security mechanisms of OPC Unified Architecture are set in different levels. Encryption and signing of the transmission as well as authentication for connection establishment can be set separately from each other. After installing the SIMATIC NET OPC UA Server, secure connections are generally possible. In addition to this encrypted communication, an unencrypted connection is also possible. The server accepts authentication with user and password or anonymous connection. These settings are "unsafe" and only serve to facilitate commissioning. The OPC UA Server can be configured to accept only encrypted transmission with user authentication.

5.3.1. OPC UA Remote Communication

All server-side settings regarding the Windows Firewall can be easily set and removed with the configuration tool "Set PC Station" (Configuration Console).

No.	Action	Note
1.	Start the configuration tool: "Communication Settings" and select the "SIMATIC NET configuration >> OP>C settings > OPC protocol selection" menu	
2.	With a single push of a button, all the necessary settings are made in the firewall to allow or block remote communication.	

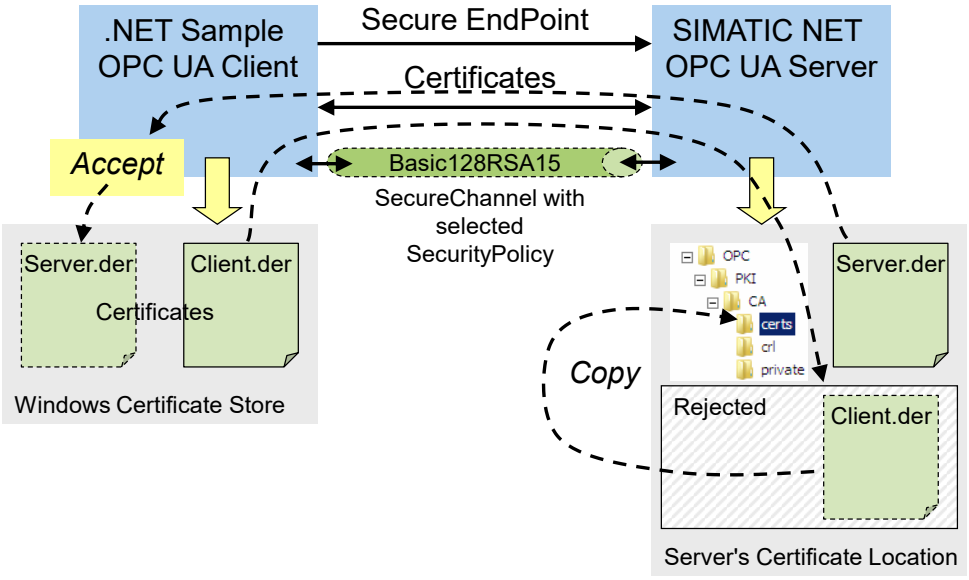
Note Note that an exception for the application and the TCP port (4845) must also be entered on the firewall of the PC "OPC Client Station".

© Siemens 2025 |

5.3.2. Certificate

When establishing a secure connection between OPC UA Client and OPC UA Server, certificates are exchanged. Both applications must check and accept the corresponding certificate of the other party in order to establish a connection.

Figure 51



The OPC UA client in this example uses Windows Certificate Store (Windows Certificate Store). This is where the client's public certificate is located.

The sample clients generate a public certificate (UA Test Client) when they are first started in this certificate management. To do this, the test client must be started with administration rights at the first start.

When establishing a connection to an encrypted connection, the server and client exchange their certificates. The client displays the certificate and the user must trust this certificate. By accepting, the server certificate is stored in Windows Certificate Management.

The OPC UA Server uses its own certificate directory and is independent of Windows Certificate Management. For security reasons, the OPC UA Server first rejects any certificate from an unknown client and stores it in a "quarantine folder" (rejected-folder). As with other server services, an administrator must copy this client certificate to the list of trusted certificates to allow the corresponding client to access the server. The place where the OPC UA Server stores and manages its own certificates and the certificates of OPC UA clients is the data directory of the OPC UA Server.

C:\Documents and settings\All Users\Application Data\Siemens\OPC\PKI\CA

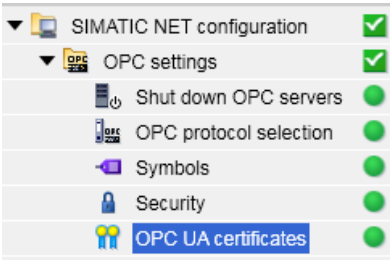
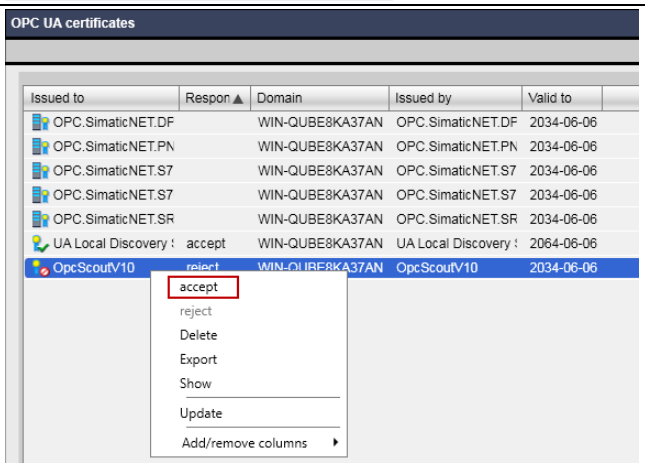
Here are three subfolders with the following content:

- **\certs**
  contains the server's public certificate as well as all trusted certificates of clients. Public certificates from OPC UA clients must be copied into this folder in order for the server to accept them.
- **\Crl**
  contains a file with a list of untrusted certificates, called the "RevocationList"
- **\Private**
  contains the private certificate of the OPC UA Server. This certificate may not be made available to anyone.

The server automatically creates the folder \reject\ below \certs\ and initially stores all unknown client certificates in this "quarantine folder". By simply "moving" the file, the certificate can be made trustworthy.

Configuration of servers with SIMATIC NET DVD from V8.2

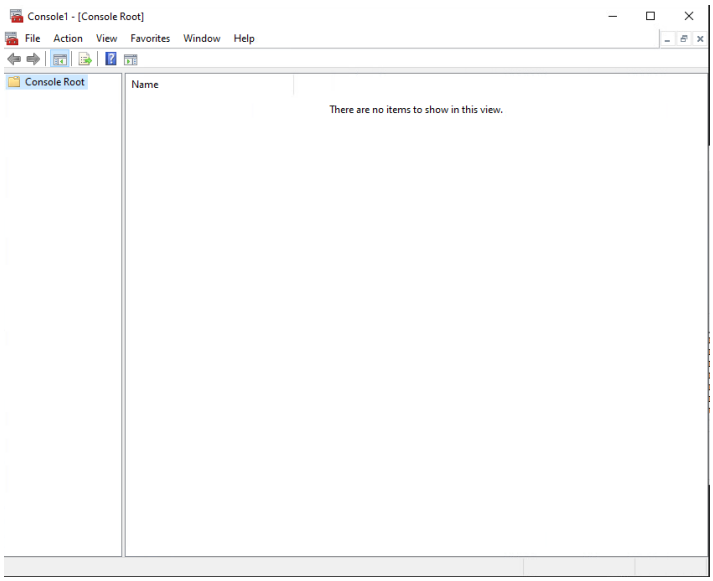
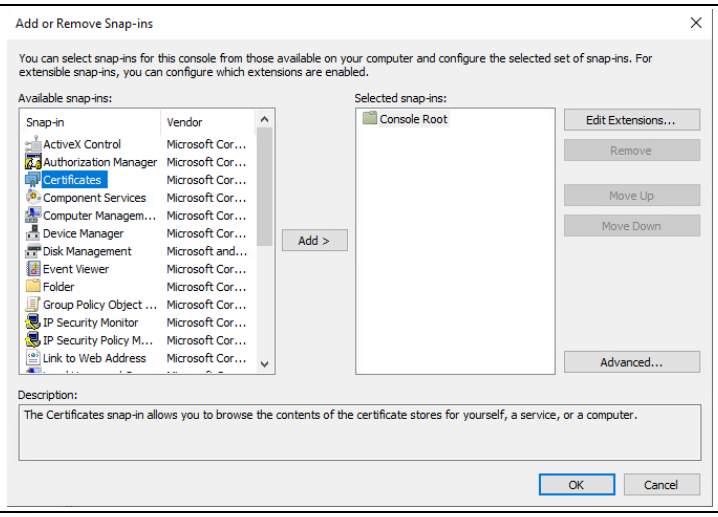
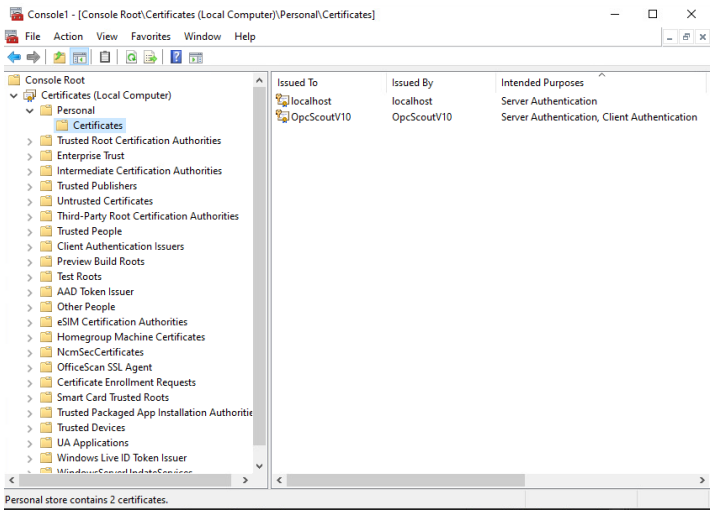
From the SIMATIC NET CD V8.0 onwards, this is possible with the "Siemens Communication Settings" configuration tool.

No.	Action	Note
1.	Start the configuration tool: "Siemens Communication Settings" and select "SIMATIC NET configuration >> OPC settings >> OPC UA certificates" ("SIMATIC NET configuration OPC settings OPC UA certificates"	On the server PC. 
2.	Here you will find all public UA certificates that the server knows. By right-clicking on a selected certificate, you can accept it.	

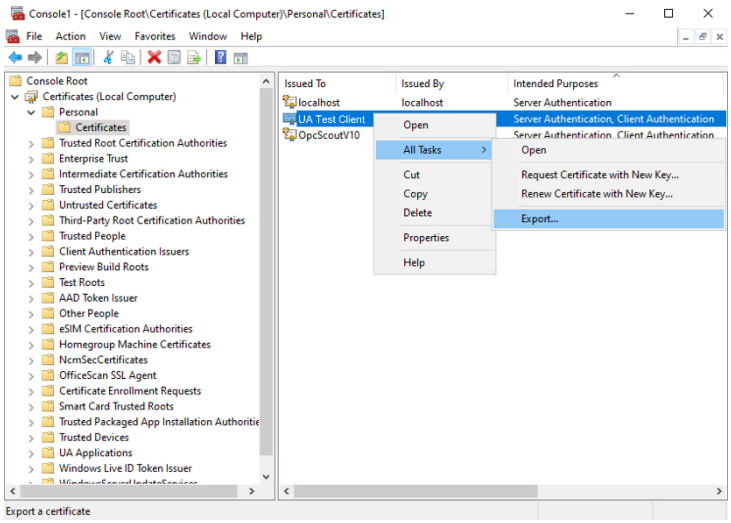
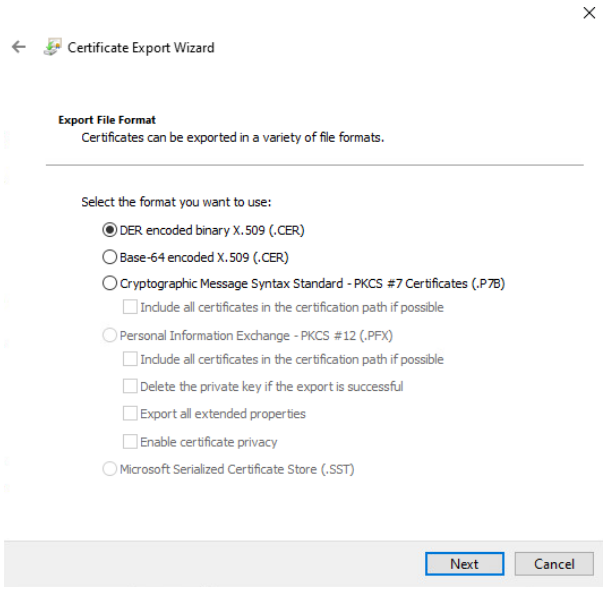
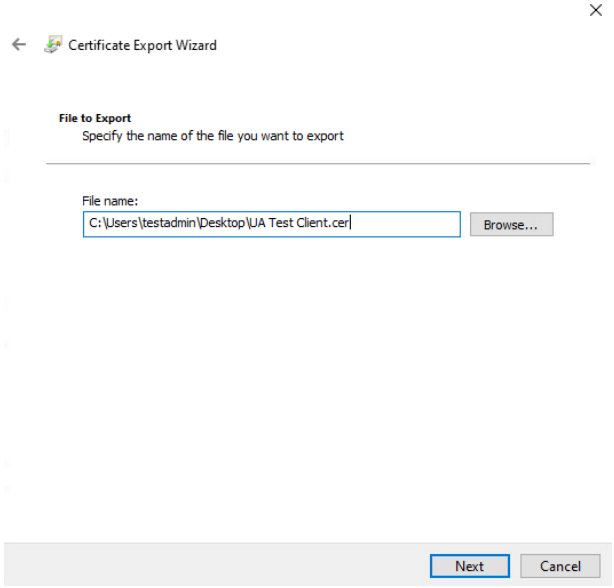
© Siemens 2025

Server Configuration with SIMATIC NET CD 2008 (V7.1)

In the SIMATIC NET CD 2008 (V7.1 or lower), trusted client certificates must be copied manually into the server's certificate directory. To do this, they are first exported from the Windows certificate management on the client PC.

No.	Action	Note
1.	Open Windows Certificate Manager in the Management Console.	On the Client PC "OPC Client Station" Start(Run("mmc" 
2.	From the File menu, choose Add/Remove SnapIn, and then choose Add>.	
3.	Select the certificates for the "computer account" of the local computer.	

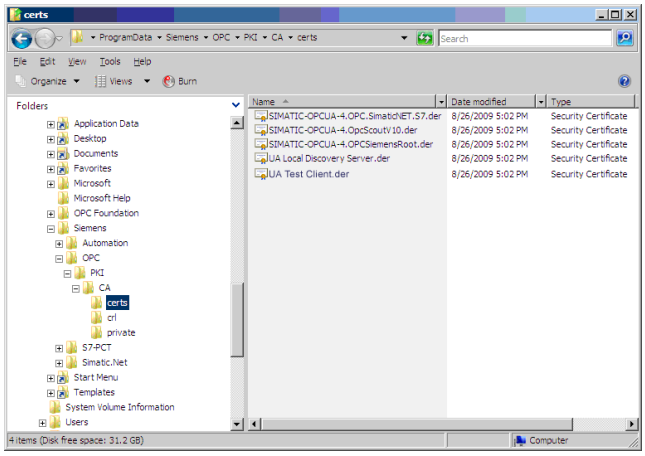
© Siemens 2025

No.	Action	Note
4.	Select the certificate you want to export (right-click(All tasks (export).	In this example, the UA Test Client certificate that was generated by the OPC UA sample clients at the first startup. The Export Wizard will start 
5.	Choose the encoding DER	
6.	Specify a file name and save the certificate file.	

© Siemens 2025

Manual import of the client certificate from the server

In SIMATIC NET CD 2008 (V7.1), trusted client certificates must be copied directly into the server's certificate directory. To do this, the certificate file exported from Windows Certificate Management on the client PC is renamed and copied to the PC server (to the server's certificate directory).

No.	Action	Note
1.	Open the directory in Windows Explorer	On the server PC "OPC Server Station"
2.	Copy the file to the server PC "OPC Server Station" and change the file extension to "DER"	Here in the example: "UA Test Client.der"
3.	In the server's certificate directory, you can find all the public OPC UA certificates that the server trusts. At this point, the client certificate must be copied to be accepted by the server.	The directory is located in the data directory of the SIMATIC NET software. C:\Documents and settings\All Users\Application Data \Siemens\OPC\PKI\CA\ 

5.3.3. Authentication, SecurityPolicy, and MessageSecurityMode

The OPC UA Server supports authentication of clients when establishing a connection. Two types of authentication are supported:

- Anonymous
- UserName / Password

ATTENTION From OPC Server V20 onwards, only nodes can be browsed when using Anonymous authentication, but no read and

After installation, both modes are active to facilitate commissioning. The server can be reconfigured so that anonymous logins are no longer possible. Username and password must be provided by the client and the server checks them against the Windows user management. This means that only clients that have a Windows account on the server machine can connect.

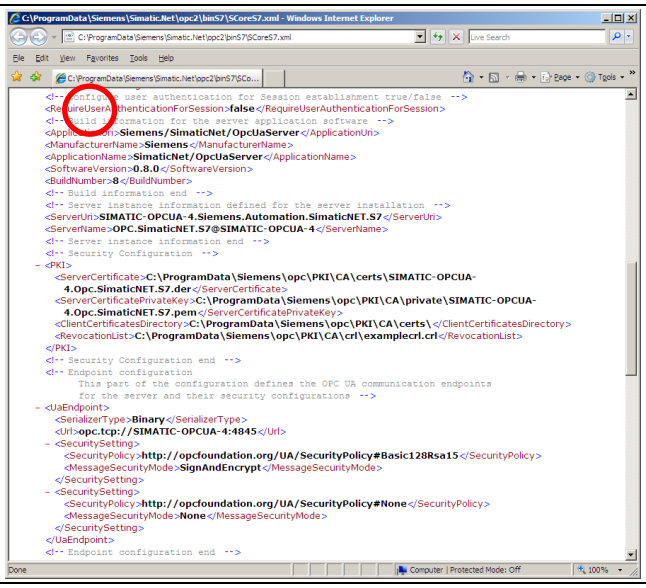
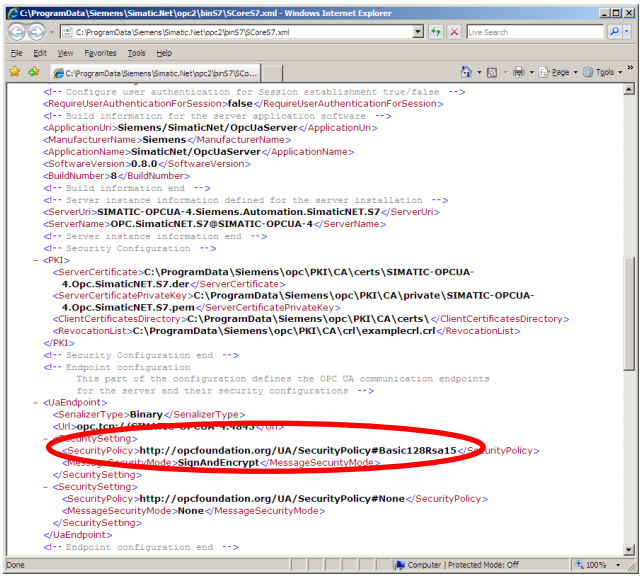
The server has two connection endpoint configurations that can be used by clients. Each of these endpoints represents a different encryption mechanism for data transmission.

- None
- Basic128RSA15

After installation, both SecurityPolicies are active to facilitate commissioning. The server can be reconfigured to use only secure encrypted connections. This means that only clients that are capable of Basic128RSA15 encryption can connect.

If you want to change the security mechanisms of the OPC UA Server, edit the configuration file of the OPC UA Server and then restart the server.

C:\ Documents and settings\All Users\Application Data \Siemens\Simatic.Net\opc2\bins7\ SCoreS7.xml

No.	Action	Note
1.	Opening the OPC UA server configuration file "SCoreS7.xml" in an editor (e.g. Notepad)	The file is located in the data directory of the SIMATIC NET software. C:\Docoments and Settings\All Users\Application Data\Siemens\Simatic.Net\opc2\bins7
2.	Set RequireUserAuthenticationForSession to true if you want each OPC UA client to authenticate to the server with a username and password in order to establish a session.	
3.	Delete the complete SecuritySetting entry for SecurityPolicy "none" and MessageSecurityMode "none" from the UAEndpoint configuration. The server then only allows encrypted transmission of the type "Basic128RSA15" with signing and encryption. Only clients that also support this type of encryption can connect (if their certificate is trusted).	Information Server OPC Unified Architecture 

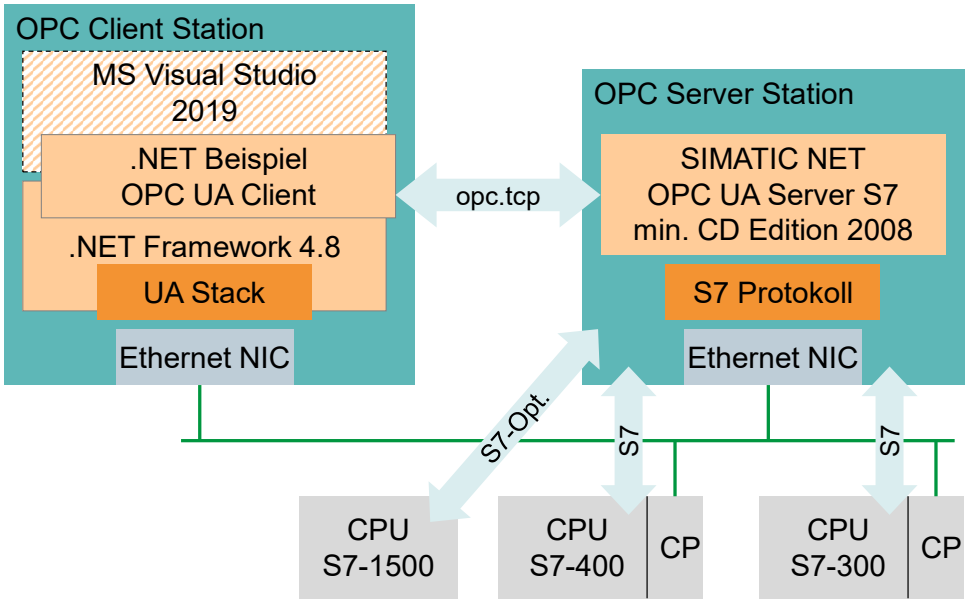
© Siemens 2025

6. Installation

This chapter describes which hardware and software components must be installed. The descriptions, manuals and delivery information that are delivered with the corresponding products must be observed in any case.

Installation of the hardware

The following picture shows the hardware structure of the application as well as the required software components.



Note The design guidelines for Industrial Ethernet networks must be observed in general.

Installing the Software

This chapter describes the steps to install the software required for the application.

No.	Action	Note
1	Install SIMATIC NET DVD V20 on the SIMATIC OPC Server Station.	
2	Install STEP 7 V20 on the Engineering Station (e.g. laptop).	
3	Install the .NET Framework 4.8 on the OPC Client Station.	This step is only necessary if the framework has not yet been installed by any other application.
4	Install Microsoft Visual Studio 2019 on the OPC UA Client Station. Install the C# development environment.	This step is only necessary if the code should be verified or modified.

Installing the Samples

This chapter describes the steps to install the sample code.

No.	Action	Note
1	Install the application software on the OPC UA Client Station. Copy the executables as well as the assemblies (folder: "bin") and also the source code (folder: "scr") into a directory to which you have access rights.	You only need the source code if you want to make modifications or verify individual functions.
2	Install the STEP 7 program. Copy and unzip the STEP 7 project.	Make adjustments to your hardware if necessary.

© Siemens 2025

7. Commissioning of the application

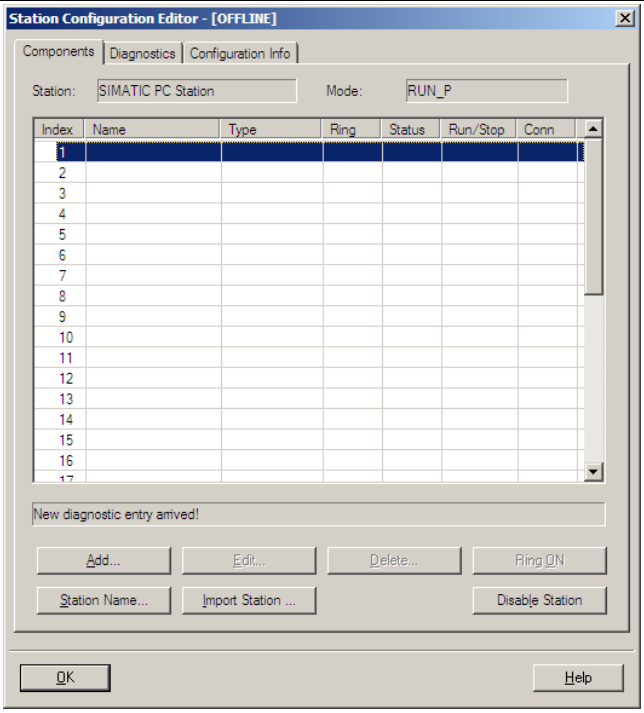
It is assumed that all hardware and software components have been successfully installed and wired.

Note The project file (XDB) delivered with this example contains the fully configured and configured PC station. This file can only be used without customization if the hardware is identical to the configuration.

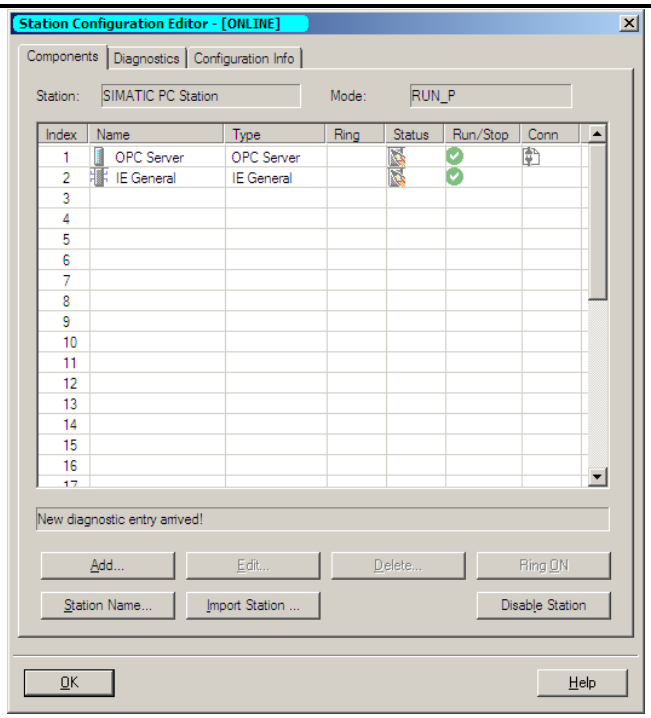
If different hardware is used, the configuration of the PC Station must be adjusted. This is especially necessary if the hardware output levels are different or the Ethernet addresses are different. The supplied STEP 7 project must then be opened and adapted accordingly (see Chapter 6). After saving and compiling, all configuration and configuration information in the XDB file is overwritten.

Commissioning of the PC Station

The following table shows the configuration and configuration of the PC Station by importing an XDB file.

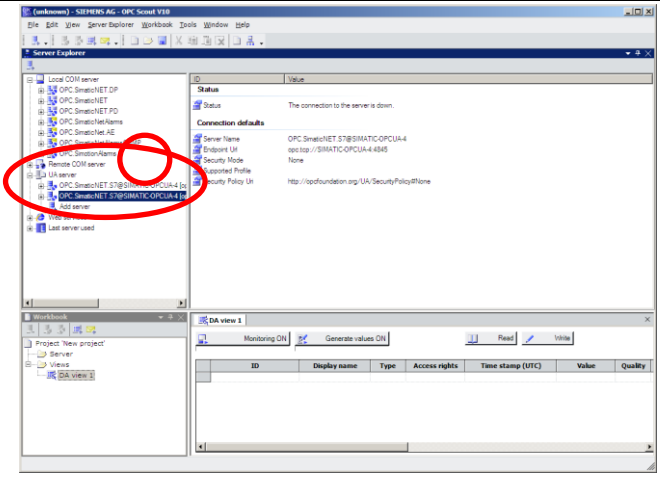
No.	Action	Note
1.	Unarchiving a project	Unzip the file OPC_UA_STEP7_CODE_V2.zip
2.	Open the project file with STEP 7 and configure the CPU stations and load the configuration onto the controllers	Make appropriate adjustments if you are using different hardware. The PC station can also be configured and loaded "online". Alternatively: load the PC station using the XDB file: see the steps below.
3.	Open the Station Configurator by double-clicking on the / icon in the taskbar.	
4.	Click `Import Station`	Confirm the following query with Yes.
5.	XDB File Selection	The XDB file is located in the XDBs subdirectory of the unzipped ZIP file in the directory tree of the STEP7 project.

© Siemens 2025 |

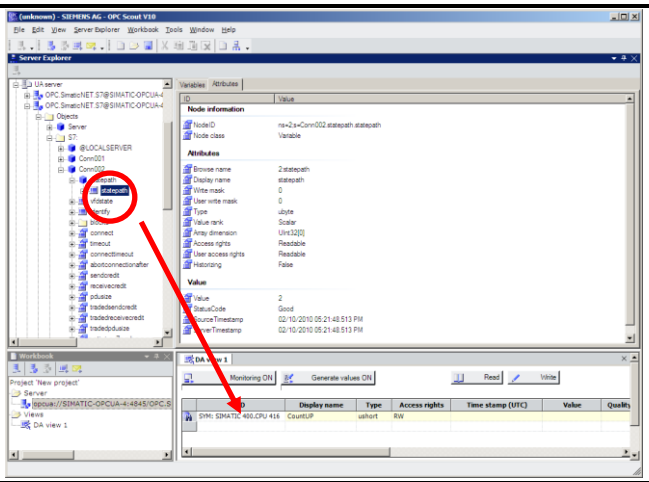
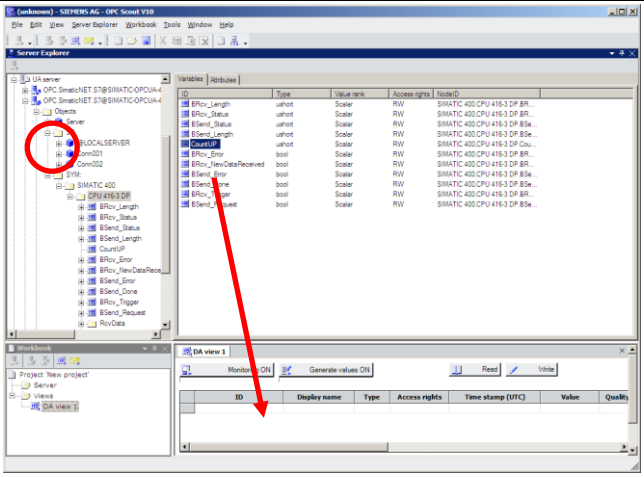
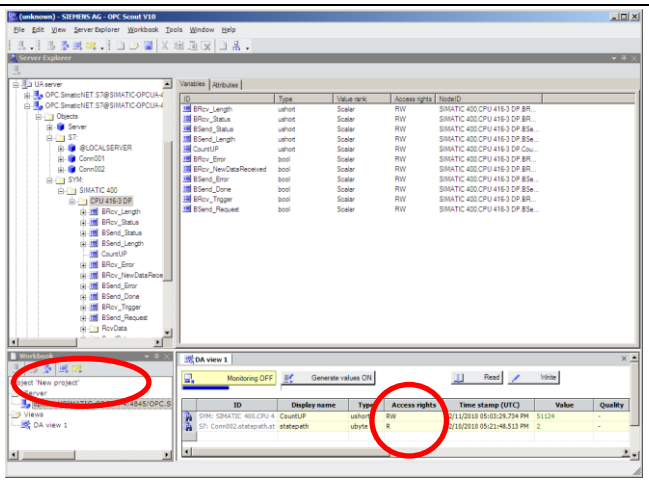
No.	Action	Note
6.	After importing the XDB file, the PC station is configured. Not only is the connection information transferred, but the symbol file is also extracted from the XDB and made available to the server.	

Testing with the OPC Scout V10

The new OPC Scout V10 is a combined OPC client that supports both the classic COM OPC interfaces and the new OPC UA. This client is delivered and installed with the SIMATIC NET CD.

No.	Action	Note
1.	Launch the OPC Scout V10 from the Start menu	On the server PC "OPC Server Station"
2.	Connect to the "insecure" endpoint "none". The connection is established as soon as you open the corresponding entry in the tree view.	

© Siemens 2025

No.	Action	Note
3.	Browse to the subfolder "S7:" and drag and drop the variable "&statepath" from the connection "Conn002" into the lower window.	
4.	Browse to the "SYM:" subfolder and drag and drop the variable "CountUP" into the lower window.	
5.	Turn on monitoring and see if there are any corresponding values in the Value column. The connection status variable must have a value of "2" (built) and the counter should increment.	

Starting the sample application

No.	Action	Note
1.	Start the corresponding application "Client" or "SimpleClient".	The files are located in the "bin" subfolder of the unzipped example.
2.	Operation of the sample application	See Chapter 8.

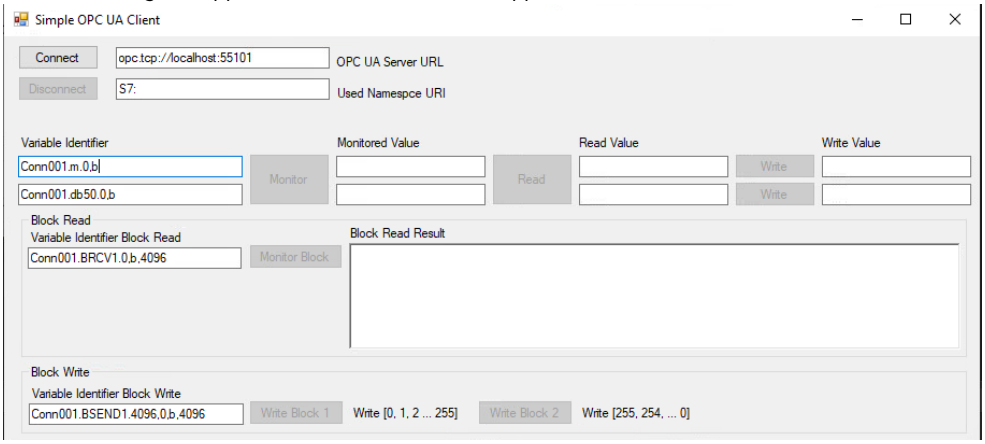
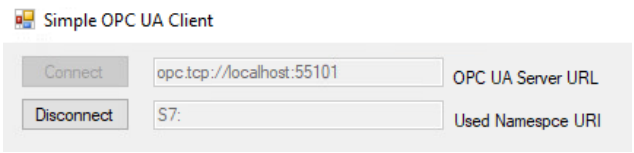
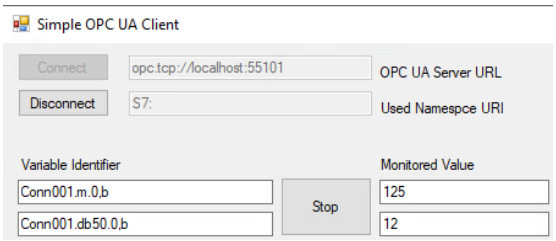
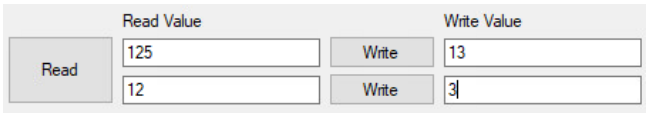
© Siemens 2025 |

8. Operation of the application

8.1. Operation Simple Client

Access with direct addressing

In the first part of the simple example, data from S7 variables is observed, read and written via direct addressing.

No.	Action	Note
1	Launch the Simple Client.	After launching the application, this interface will appear. 
2	Establishing a connection to the server	The server URL must be entered manually. All identifiers for NodeIds in the interface use the same namespace. The default namespace is S7: for direct addressing. The connection to the server can be established via the Connect button.  **Note** In order to be able to access optimized data modules of the S7-1200/S7-1500 CPUs, you must have at least the OPC server V12 configured and access the OPC server with the client via the SimetcNET.S7OPT (port 55105). **Note** With OPC Server V20, anonymous login is disabled by default. This can be activated in the communication settings per namespace. If this is not activated, an authentication mask opens. By default, you log in here with your Windows ID.
3	Monitoring of data	Two variables can be specified for observing data changes, reading and writing. The NodeId is formed from the identifier and the namespace. Observation can be switched on via the Monitor button. The reported data changes are displayed in the text boxes next to the button. 
4	Reading and Writing	The same NodeIds are used for reading and writing as for observing. By pressing the Read button, both variables are read. The two write buttons can be used to write the variables individually. Before writing, you have to read first, because the data conversion for writing is done on the basis of the read values. 

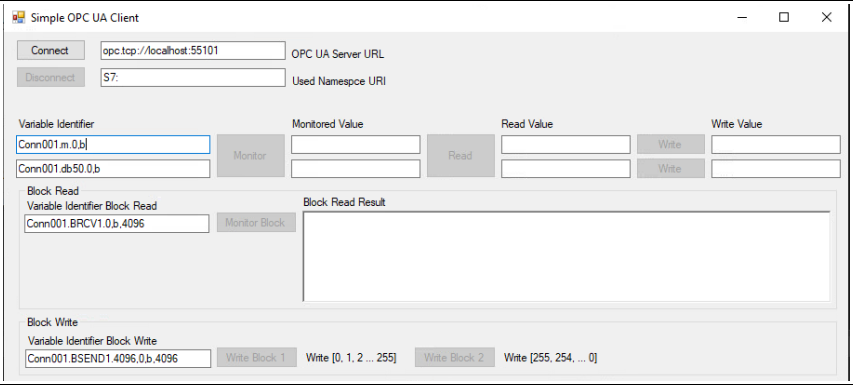
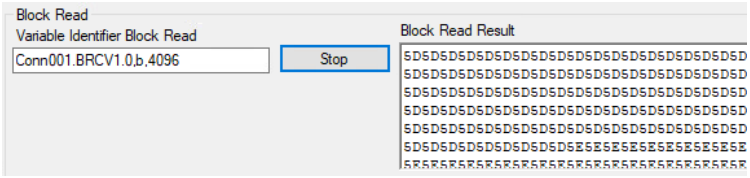
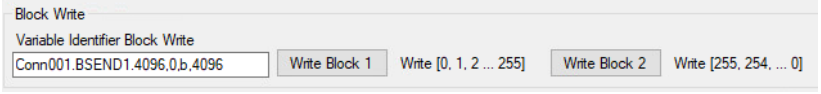
© Siemens 2025 |

Use of Block Services

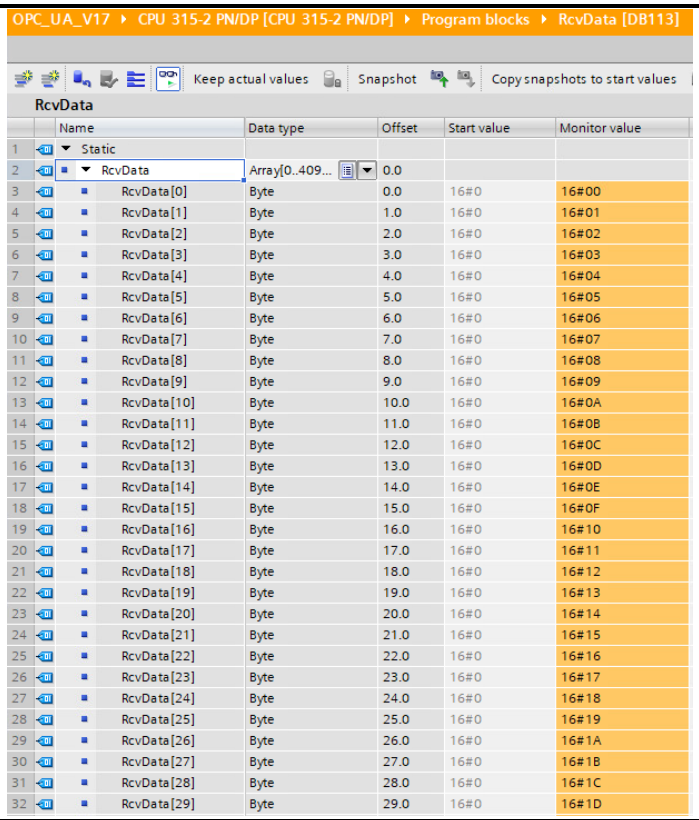
As a clear, simple example, here is a typical use case for block-oriented services: sending recipe data to the S7 or receiving result data from the S7 using the block services BSEND and BRCV.

The S7 controller receives a recipe from the PC station "OPC Client Station". For this purpose, two "recipes" were simulated in the OPC UA Client, which are sent to the OPC UA server by a write request to the NodeID of the BSEND variables after pressing the respective button. The OPC UA server makes a BSEND call to the corresponding controller. The S7, in turn, has to call up a suitable BRCV to receive the data. The BRCV must be available at the right time with the correct RID on the correct connection with the necessary minimum length. Follow the steps below.

Table 81

No.	Action	Note
1.	Start STEP 7 and open the variable table (VAT) of the receive buffer.	Go to the SIMATIC Control -> CPU ->S7-Program -> Modules and open "VAT_ReceivedData". Switch to Online View (Glasses)
2.	Launch the OPC UA Client and connect to the OPC UA Server	
3.	Received by the S7	To receive from the S7, a subscription with Monitored Item must be created. To do this, click the "Monitoring Block" button in the "Block Read" group. In the field next to the button, the result data is displayed as a HEX code: Note the R_ID as part of the NodeId. The S7 must supply the BSEND device with this ID 
4.	Sending the data to the S7	Select the "Recipe" by pressing either "Write Block 1" or "Write Block 2". The first button writes to the data block by incrementing a value for each array entry from 0. In the second case, the array entries are decremented from 255 onwards. Note the R_ID as part of the NodeId. This ID must be used to supply the BRCV module in the S7. 

© Siemens 2025 |

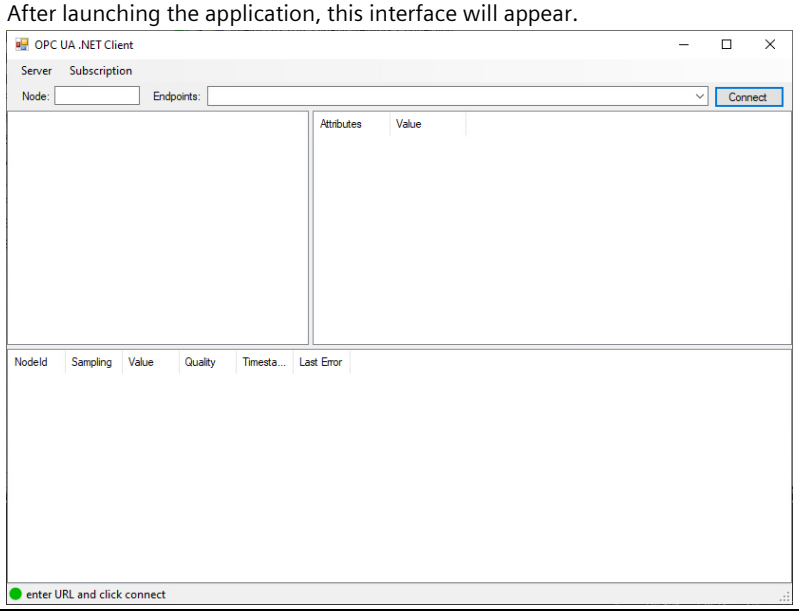
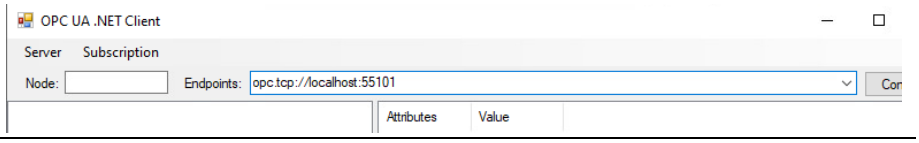
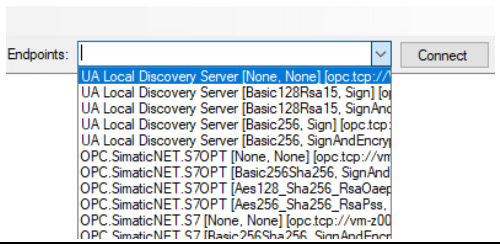
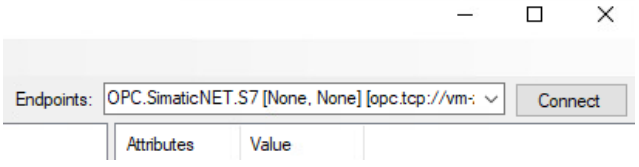
No.	Action	Note
5.	In the data module "RcvData [DB113]", check whether the data has arrived.	
6.	Processing in S7	At this point, the prescription should be processed in the S7 and then the BRCV should be made ready to receive again.
7.	Re-enable Receive	After the data has been processed, "arm" the BRCV again (this is done automatically in the example).
8.	Resend	Now send the second prescription using the OPC UA Client.

	Name	Data type	Offset	Start value	Monitor value
1	Static				
2	RcvData	Array[0..409...	0.0		
3	RcvData[0]	Byte	0.0	16#0	16#00
4	RcvData[1]	Byte	1.0	16#0	16#01
5	RcvData[2]	Byte	2.0	16#0	16#02
6	RcvData[3]	Byte	3.0	16#0	16#03
7	RcvData[4]	Byte	4.0	16#0	16#04
8	RcvData[5]	Byte	5.0	16#0	16#05
9	RcvData[6]	Byte	6.0	16#0	16#06
10	RcvData[7]	Byte	7.0	16#0	16#07
11	RcvData[8]	Byte	8.0	16#0	16#08
12	RcvData[9]	Byte	9.0	16#0	16#09
13	RcvData[10]	Byte	10.0	16#0	16#0A
14	RcvData[11]	Byte	11.0	16#0	16#0B
15	RcvData[12]	Byte	12.0	16#0	16#0C
16	RcvData[13]	Byte	13.0	16#0	16#0D
17	RcvData[14]	Byte	14.0	16#0	16#0E
18	RcvData[15]	Byte	15.0	16#0	16#0F
19	RcvData[16]	Byte	16.0	16#0	16#10
20	RcvData[17]	Byte	17.0	16#0	16#11
21	RcvData[18]	Byte	18.0	16#0	16#12
22	RcvData[19]	Byte	19.0	16#0	16#13
23	RcvData[20]	Byte	20.0	16#0	16#14
24	RcvData[21]	Byte	21.0	16#0	16#15
25	RcvData[22]	Byte	22.0	16#0	16#16
26	RcvData[23]	Byte	23.0	16#0	16#17
27	RcvData[24]	Byte	24.0	16#0	16#18
28	RcvData[25]	Byte	25.0	16#0	16#19
29	RcvData[26]	Byte	26.0	16#0	16#1A
30	RcvData[27]	Byte	27.0	16#0	16#1B
31	RcvData[28]	Byte	28.0	16#0	16#1C
32	RcvData[29]	Byte	29.0	16#0	16#1D

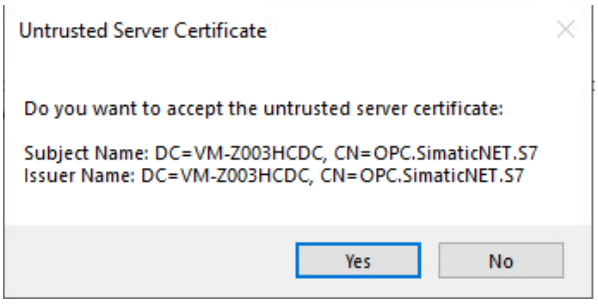
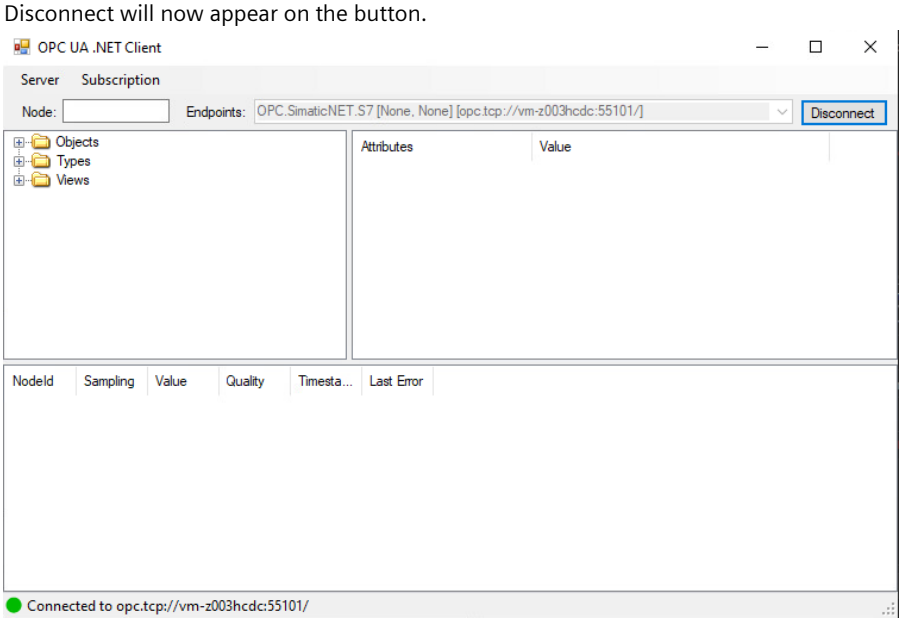
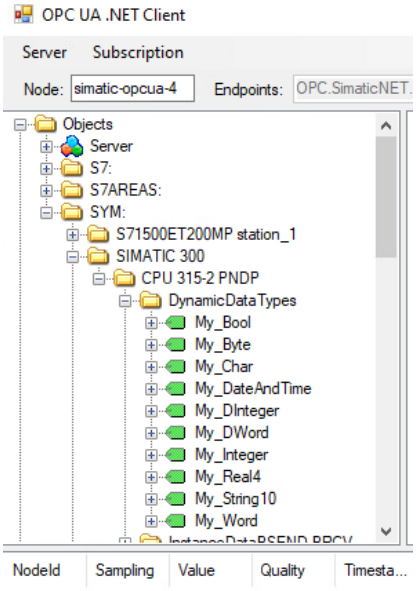
© Siemens 2025 |

8.2. Convenient OPC UA Client operation

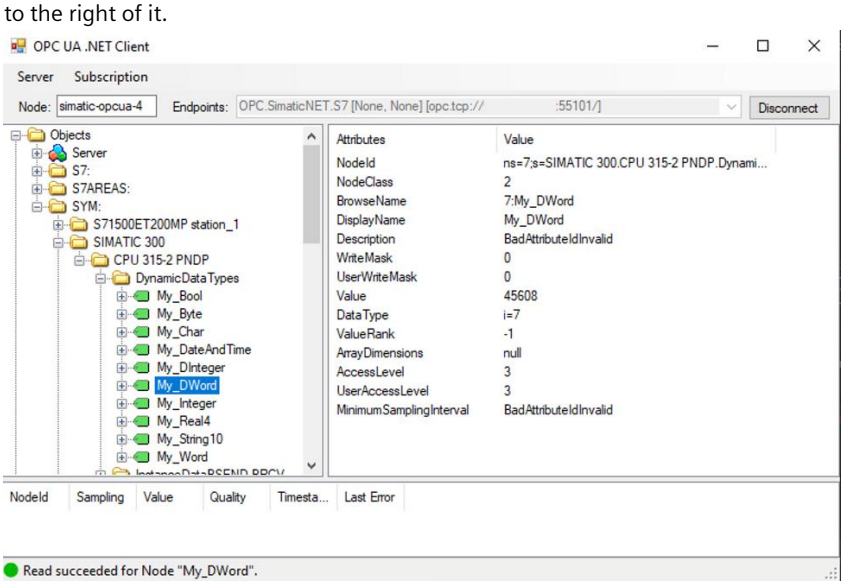
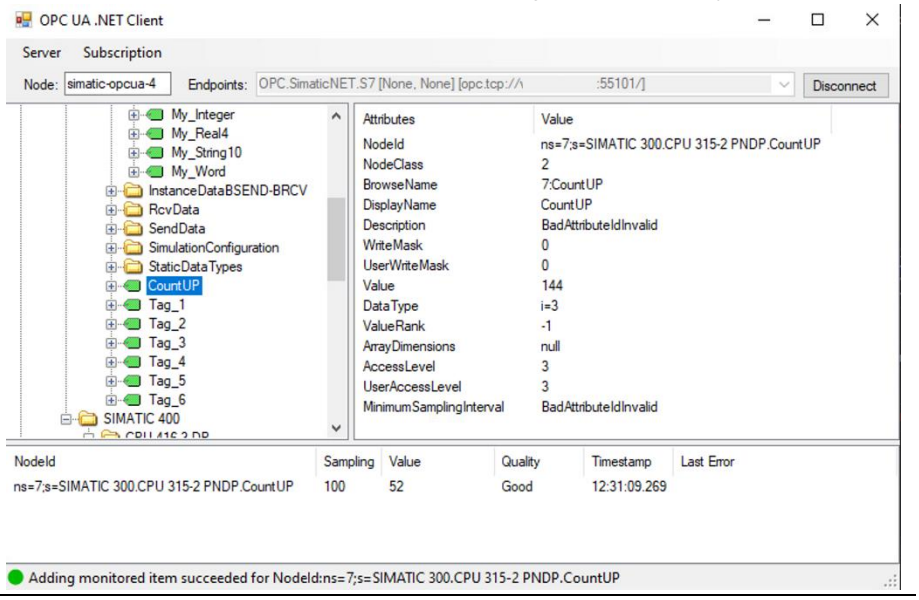
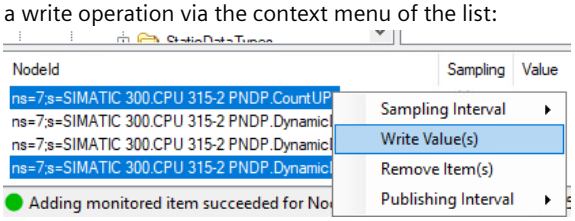
Table 82

No.	Action	Note
1	Starting the OPC UA Client (this example)	After launching the application, this interface will appear. 
2	Manually entering a server URL	To establish a connection to an OPC UA server, the URL of the server can be entered manually in the Endpoints text field. If the SIMATIC NET OPC UA Server is installed on the same computer, the URL opc.tcp://localhost:55101 can be used. Manual entry attempts to establish the connection without security. **Note** In order to be able to access optimized data modules of the S7-1200/S7-1500 CPUs, you must have at least the OPC server V12 configured and access the OPC server with the client via the SimetcNET.S7OPT (port 55101). 
3	List available endpoints	The list of available endpoints can be used to list the list of available endpoints via the Endpoints drop-down list. If the Node text field is empty, the local servers or their endpoints are listed. Entering a remote machine name in the Node text field and then clicking on the Endpoints drop-down list displays the available endpoints on the remote machine. In addition to the server name, the security settings and the URL of the endpoint are displayed. 
4	Establishing a connection to the server	The Connect button establishes the connection to the server. 

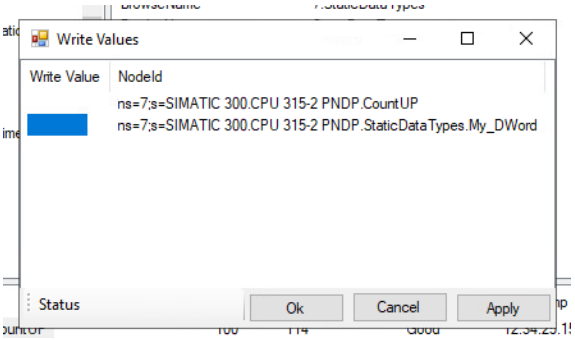
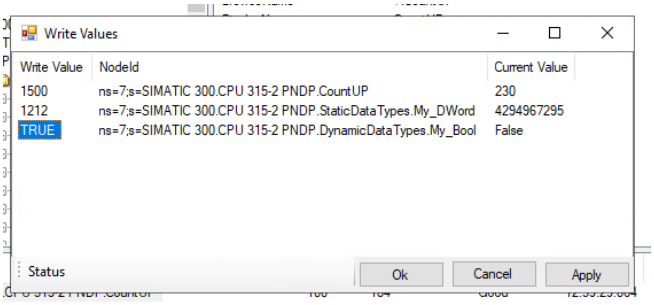
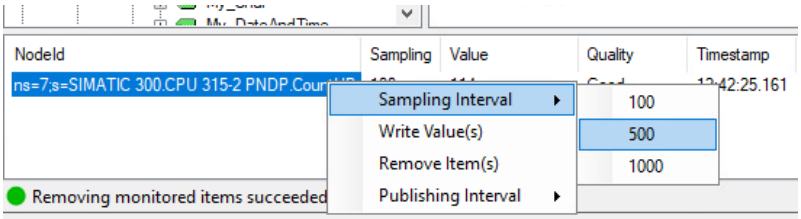
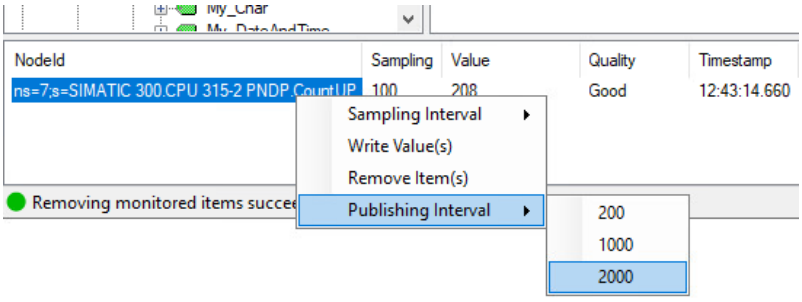
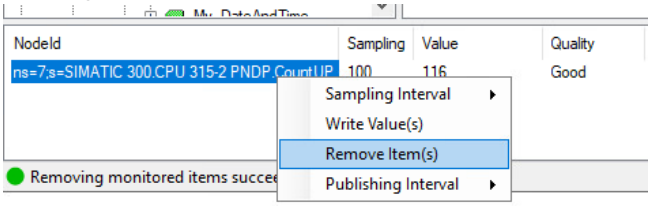
© Siemens 2025 |

No.	Action	Note
		If the server's certificate is not trusted, the user is asked if he or she still wants to accept the certificate. The necessary steps to accept the client certificate from the server are described in chapter 5.3.2.  If the connection was successful, the top level of the address space of the OPC UA server is displayed in the window at the top left: Disconnect will now appear on the button. 
5	Browse	The tree view allows you to navigate in the server's address space. 

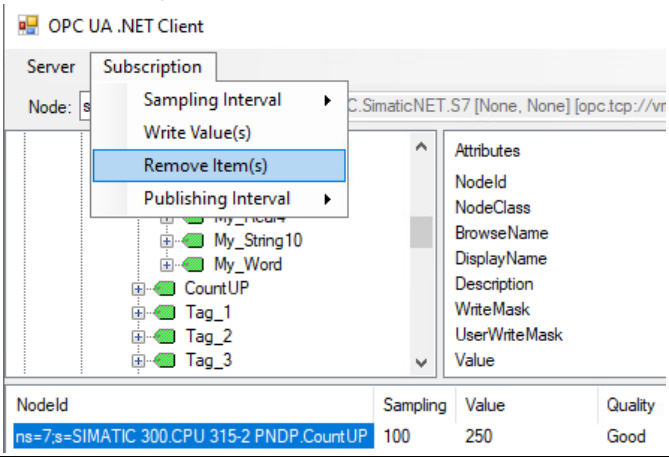
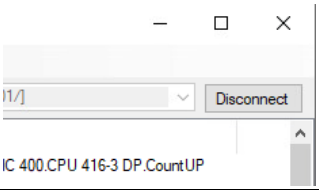
© Siemens 2025 |

No.	Action	Note
6	Read attributes of a node	By selecting a node in the tree view, its attributes and their values are displayed in the list to the right of it. 
7	Monitoring Variables (Monitored Items)	Variables present in the server's address space can be dragged and dropped from the tree view into the lower observation window. Value changes are then displayed there. 
8	Writing Values	If one or more variables have been selected in the observation window, the user can start a write operation via the context menu of the list:  In the dialog box that now opens, the variables previously selected in the list appear with their current values:

© Siemens 2025

No.	Action	Note
		 Values can be entered in the Write Value column:  Pressing the Ok button will write the new values and the dialog will close. If, on the other hand, the Apply button is pressed, the values are written and the dialog remains open for further writing of values.
9	Changing the Sampling Interval	Three different sampling intervals can be set via the context menu. It is possible to select several entries from the list field. 
10	Change Publish Interval	The publish interval of the subscription can also be changed via the context menu of the observation window. 
11	Remove Monitored Items	Selecting the Remove Item(s) menu item removes all variables selected in the list. 

© Siemens 2025

No.	Action	Note
12	Calling Actions from the Menu Bar	Actions 8 through 11 can also be performed from the menu bar: 
13	Disconnect from the server	The connection to the OPC UA server is disconnected via the disconnect button. 
14	Connection error OPC UA server for control	If the connection between OPC UA Server and S7 is lost, the status of the variables in the watchlist changes to Bad.
15	Errors in calls	If errors occur during OPC UA calls, these errors are displayed in the status bar of the application. If the server does not respond, the Client API provides timeout errors.

© Siemens 2025 |

9. Further notes, tips and tricks, etc.

Client API reuse and extension

The Client API is implemented as an independent, reusable assembly DLL. It can be used directly in other applications. An extension to additional OPC UA features such as method calls is easily possible.

Reuse of GUI Controls

The GUI elements for browsing, listing attributes and monitoring variable values are created as controls. For reuse, it makes sense to outsource these controls to a separate assembly DLL.

Storing NodeIds

NodeIds are made up of an identifier and the namespace index. The namespace index does not change as long as the OPC UA server is running, but in principle the index can change when the OPC UA server is restarted. This is not the case with the SIMATIC NET OPC UA server, but an OPC UA client should be prepared for this when it stores the NodeIds.

When saving, the index must not simply be saved, but the namespace URI must be stored. This URI remains constant even if the index changes.

There are two strategies for storing the namespace URI:

- The URI is stored with the identifier for the NodeId instead of the index. This is the simplest option, but has the disadvantage that a lot of redundant information is stored if the namespace URI is the same for all stored NodeIds of the server.
- The index is stored with the identifier, but in parallel with it the matching namespace table with the namespace URIs is stored. This variant is more efficient, but requires an additional location for the table.
- For both variants, after establishing a connection with the server, the namespace table must be read by the server and the namespace URI must be converted back into the current index.

Optimization of NodeIds for Read and Write

NodeIds can contain long texts and are therefore not suitable for use in cyclic calls to read and write, as it creates unnecessary overhead on the network and during processing in the server.

OPC UA provides the special services RegisterNodes and UnregisterNodes to achieve optimization. For a list of original NodeIds, RegisterNodes provides a list of optimized NodeIds with numeric identifiers that can be used like handles. These NodeIds are only four bytes long on the network and can be used in the server for very fast access to the data.

Since the handle is also a NodeId, it can be used in all services instead of the original NodeId. However, the optimized NodeId is only valid within the session.

If registered NodeIds are no longer needed, they should be released with UnregisterNode to free up the resources in the server.

© Siemens 2025 |

10. Appendix

10.1. Service and support

SiePortal

The integrated platform for product selection, purchasing and support - and connection of Industry Mall and Online support. The SiePortal home page replaces the previous home pages of the Industry Mall and the Online Support Portal (SIOS) and combines them.

- Products & Services
  In Products & Services, you can find all our offerings as previously available in Mall Catalog.
- Support
  In Support, you can find all information helpful for resolving technical issues with our products.
- mySieportal
  mySiePortal collects all your personal data and processes, from your account to current orders, service requests and more. You can only see the full range of functions here after you have logged in.

You can access SiePortal via this address: sieportal.siemens.com

Technical Support

The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers – ranging from basic support to individual support contracts.
Please send queries to Technical Support via Web form: support.industry.siemens.com/cs/my/src

SITRAIN – Digital Industry Academy

We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that's tailored to the customer's specific needs.
For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page: siemens.com/sitrain

Industry Online Support app

You will receive optimum support wherever you are with the "Industry Online Support" app. The app is available for iOS and Android:

10.2. Links and Literature

Bibliographic references

No.	Topic
\1\	Automating with STEP7 in AWL and SCL Hans Berger Publicis MCD Verlag ISBN 3-89578-113-4
\2\	OPC Unified Archticture Mahnke, Leitner, Damm Springer Verlag ISBN 978-3-540-68898-3

Internet links

No.	Topic
\1\	SiePortal https://sieportal.siemens.com/en-ww
\2\	Link to the contribution page of the application example https://support.industry.siemens.com/cs/ww/en/view/42014088
\3\	OPC Data Access Custom Interface Version 3.0 & OPC Unified Architecture Specification on the OPC Foundation website for download for OPC members www.opcfoundation.org
\4\	SIMATIC NET PC Software Industrial Communication with PG/PC Volume 1 - Basics of the System Manual Description or information about: • General information about PC tools. • Features of NCM PC. https://support.industry.siemens.com/cs/document/77376110
\5\	SIMATIC NET PC Software Industrial Communication with PG/PC Volume 2 - Interfaces Programming Manual Manual for Industrial Communication on PG/PC with SIMATIC NET. https://support.industry.siemens.com/cs/document/77378184
\6\	Unified Automation http://www.unified-automation.com/downloads/opc-ua-development.html

© Siemens 2025 |

10.3. Change documentation

Version	Date	Alteration
V2.0	11/2025	Complete revision • Migrating to Step 7 V20 • Latest Version UA Client Helper API • OPC UA 1.05 • Revision of the documentation • Simple client adjustments regarding BSEND and BRCV functionality
V1.2	12/2017	Revised version: Migration to STEP 7 V14, SIMATIC NET V14, OPC UA 1.03
V1.1	06/2014	Revised version: Migration to STEP 7 V13
V1.0	04/2010	First edition

© Siemens 2025 |