OPC UA .NET Client for the SIMATIC S7-1500 OPC UA Server

S7-1500 / OPC UA / .NET / C#

https://support.industry.siemens.com/cs/ww/en/view/109737901

Siemens Industry Online Support

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. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.
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. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

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. 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://support.industry.siemens.com) shall also apply.

Security information

Siemens provides products and solutions with Industrial Security 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 security 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 security measures that may be implemented, please visit https://www.siemens.com/industrialsecurity.

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 Security RSS Feed at: https://www.siemens.com/industrialsecurity.

Table of contents

1 Introduction

1.1 Overview

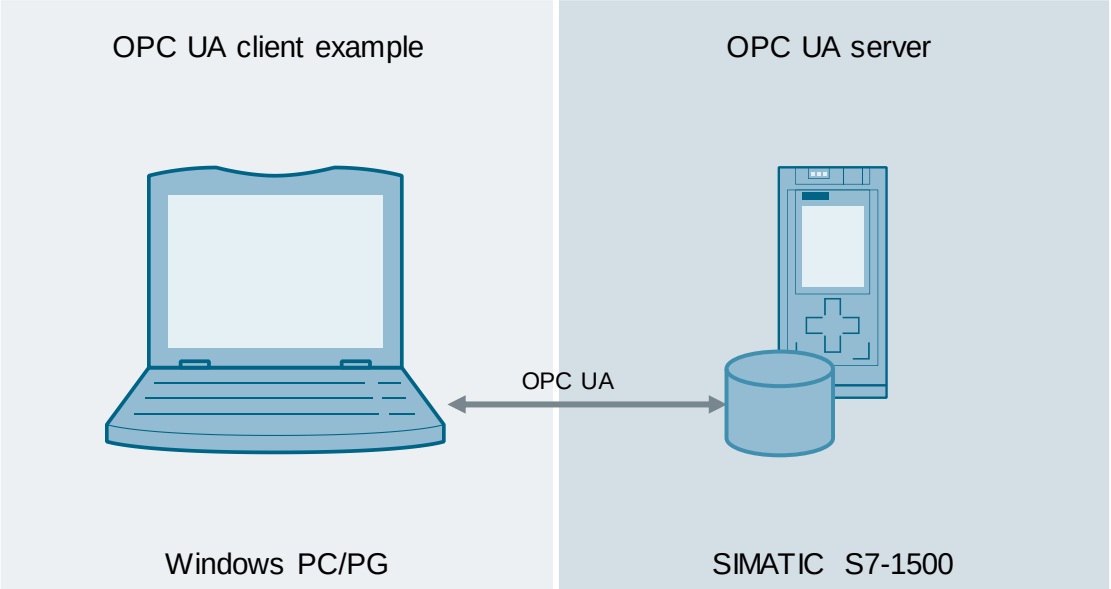
OPC UA (Open Platform Communications Unified Architecture) is an M2M communication protocol adopted in 2009 that was specified by the OPC foundation. The OPC specification has been developed to create an interoperable, secure and reliable communication protocol. Based on these properties OPC UA increasingly prevails as standard in the industrial environment.

With the current firmware of SIMATIC S7-1500 an integrated OPC UA Server has been added to the control system. This enables an additional option of direct process data exchange of the SIMATIC S7-1500 with a wide variety of other systems that support OPC UA.

Content of this application example

In order to exchange data with the server of the SIMATIC S7-1500 via OPC UA, this application example will show you how to create a simple client in .Net. Moreover, it explains step by step how to configure the OPC UA server of the SIMATIC S7-1500.

Figure 1-1



Advantages of the application example

This application example offers you the following advantages:

- Expandable TIA project with preconfigured OPC UA server for a SIMATIC S7-1500
- A simple and expandable OPC UA client, created in C# for .NET
- A commented C# class that summarizes the OPC UA client basic functions and guarantees easy implementation

Assumed knowledge

The following basic knowledge is required by the user:

- Basics of programming in C#/.NET
- Basics of configuring in the TIA Portal
- Basics of OPC
- Basics in software security and certificate handling

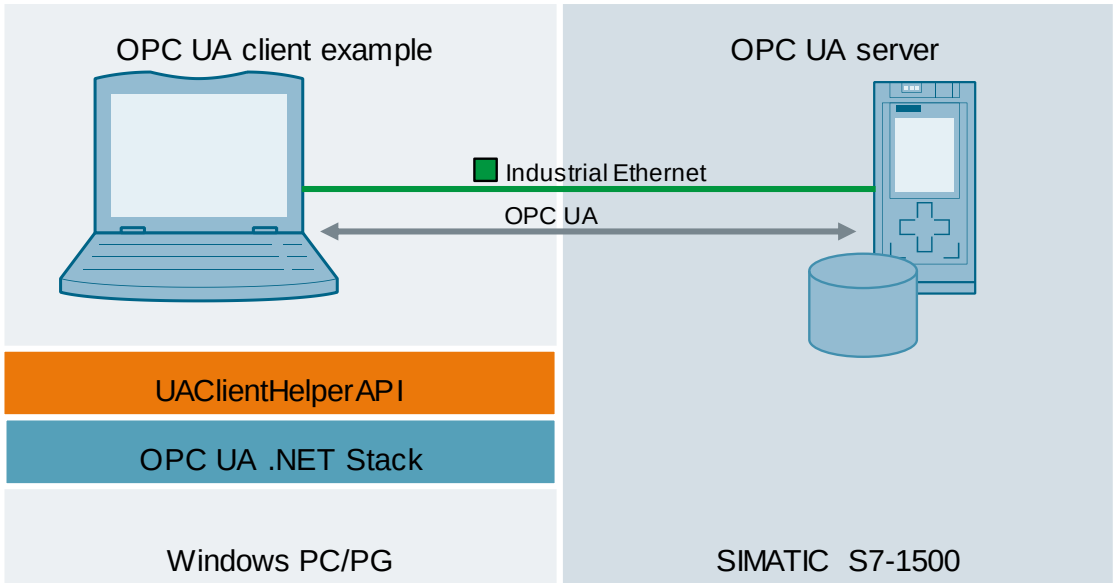
1.2 Mode of operation

Below, you will find an explanation of what components, functions and mode of operations are used in the application example.

General function description

The following figure shows the most important components of this application example:

Figure 1-2



A simple OPC UA .NET client for Windows PCs/PGs communicates with the OPC UA server of a SIMATIC S7-1500.

The client supports the following OPC UA service sets:

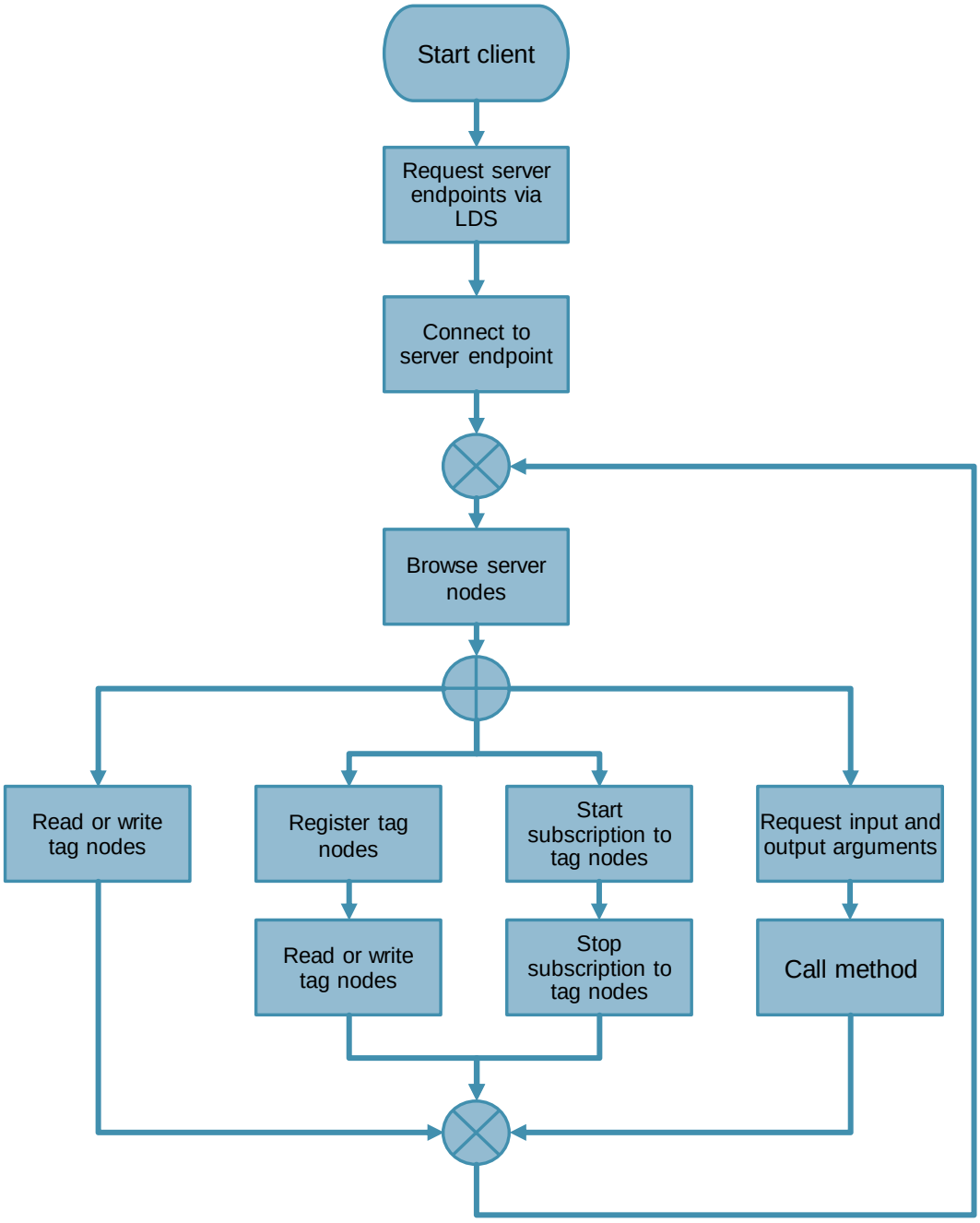
- Searching and finding servers: Discovery Service Set (FindServers, GetEndpoints)
- Creating and ending sessions: Sessions Service Set (CreateSession, CloseSession)
- Navigating in the address space: View Service Set (Browse, RegisterNodes, UnregisterNodes)
- Reading and writing tags and attributes: Attribute Service Set (Read, Write)
- Subscribing to tags: Subscription Service Set (CreateSubscription, DeleteSubscription); MonitoredItem Service Set (CreateMonitoredItem, DeleteMonitoredItem)
- Calling methods: Method Service Set (GetMethodArguments, CallMethod)

The SIMATIC S7-1500 OPC UA server is planned and configured via the TIA Portal. The OPC UA Client is created in C#/.NET and internally uses the freely accessible OPC UA .NET stack of the OPC Foundation. For easier individual implementations of a .NET client, the "UAClientHelperAPI" C# class is included. This class summarizes the basic functions of the .NET stack of the OPC Foundation and considerably facilitates the use of the basic functions for you. Client and server are connected via Ethernet and communicate through OPC UA via TCP/IP.

Functional sequence

Once the OPC UA server has been planned, configured (with client certificate) and loaded into the CPU, the following functional sequence is the result for the client of this example:

Figure 1-3



NOTE

In order to request server endpoints via a LDS (Local Discovery Server) or GDS (Global Discovery Server) a LDS has to be installed on the PC/PG or a GDS has to be available in the network.

1.3 Components used

This application example was created with the following components:

Table 1-1

Component	Qty.	Article number	Note
S7-1500 CPU 1511TF-1 PN	1	6ES7 511-1UK01-0AB0	Any S7-1500-CPU compatible, Firmware 2.9 or higher
STEP 7 Professional	1	6ES7822-1..05-..	TIA Portal V15 or higher
Visual Studio 2020	1	-	Community version also possible.
OPC UA .Net stack	1	-	V1.04.369.30 Download: Links & Literature in item \2\.

NOTE

Requires .NET runtime environment.

This application example contains the following components:

Table 1-2

Component	File name	Note
Documentation	109737901_OPC_UA_Client_S7-1500_DOKU_V1_5_en.pdf	This document.
Example project	109737901_OPC_UA_Client_S7-1500_CODE_V1_5.zip	ZIP archiv contains Visual Studio and TIA Portal project.

NOTE

The TIA Portal project is protected. For signing on you need the following credentials:

User: User

Password: Siemens.1

2 Engineering

2.1 Configuring the OPC UA server of the S7-1500

The following step-by-step instructions show you how to plan and configure the SIMATIC S7-1500 OPC UA server via the TIA Portal.

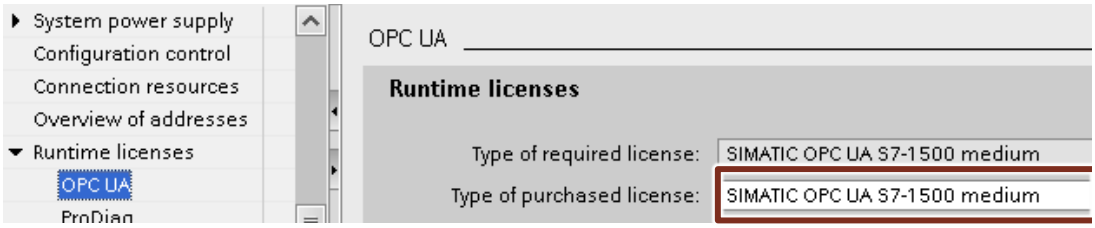
Prerequisites

- Create a new TIA Portal project (V15 or higher).
- Configure a SIMATIC S7-1500 with firmware 2.9 or higher.

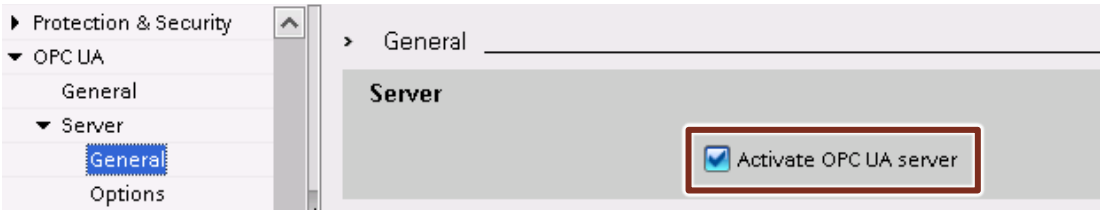
2.1.1 Enabling the OPC UA server

The OPC UA server of the S7-1500 is disabled by default. The instructions below show you how to enable the required steps on the server:

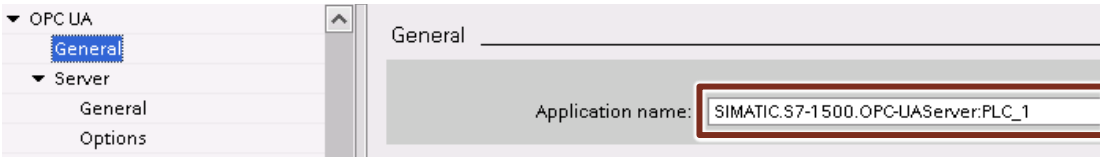
1. Navigate to the “Properties” of the configured S7-1500 CPU in the TIA Portal.
2. Navigate to “Runtime licenses > OPC UA” in the inspector window and select the required license there.



3. Navigate to “OPC UA > Server > General” in the inspector window and enable the “Activate OPC UA server” check box there.

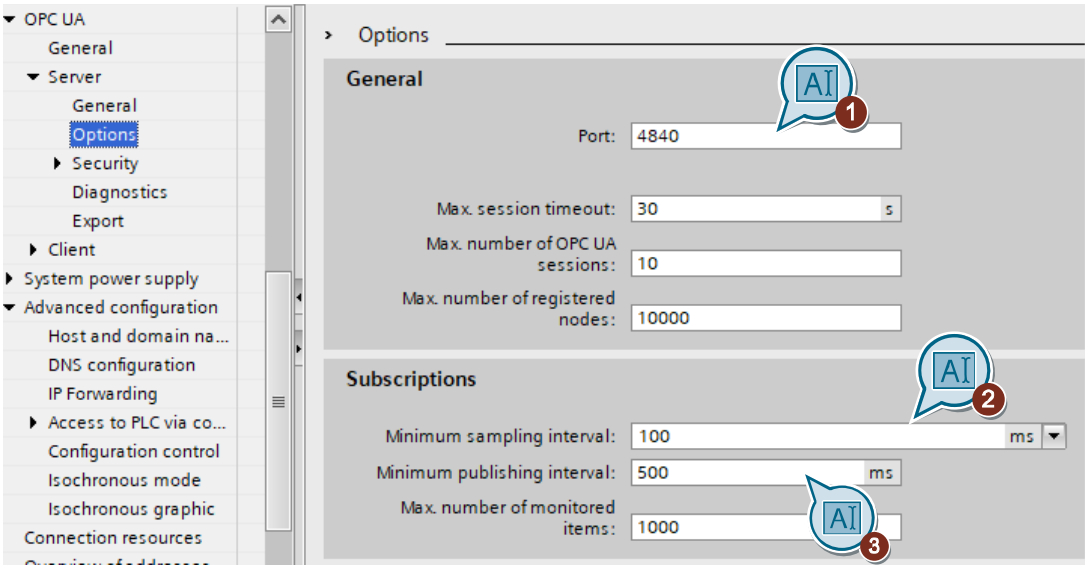


4. Navigate to “OPC UA > General” in the inspector window and assign a suitable name for your OPC UA server in the “Application name” field. The S7-1500 UA server identifies itself to the UA clients via this name.



NOTE This setting is sufficient to enable the OPC UA server of the CPU and to guarantee basic operation. Please note that the server in its standard configuration allows the connection of any client.

5. Navigate to “OPC UA > Server” > Options” in the inspector window and assign your desired port address for the OPC UA server of the CPU. Furthermore, assign a “Minimum publishing interval” and a “Minimum sampling interval” for the OPC UA server.



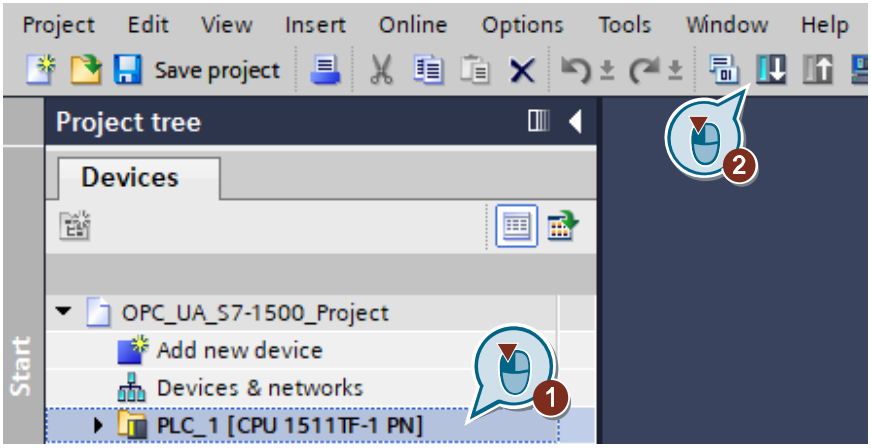
NOTE

"Minimum publishing interval":
This value determines at what minimal intervals the OPC UA server is allowed to send data to a client via OPC UA subscriptions.

“Minimum sampling interval”:
This value determines at what minimum intervals the OPC UA server is allowed to request data changes of the CPU data management.

These values have an influence on the communication and CPU load and should therefore be considered. The minimum value depends on the CPU type.

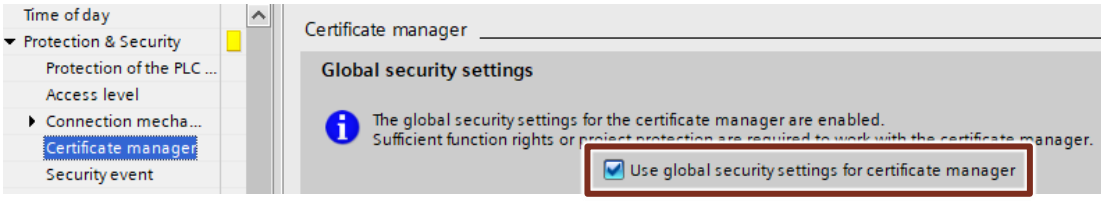
6. Select the CPU in your project tree and load the project into the controller.



2.1.2 Enabling global security settings

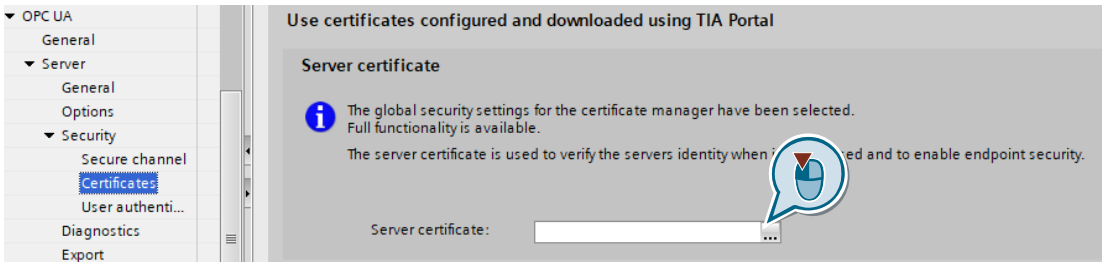
In order to manage the software certificates for the OPC UA server, the global security settings of the TIA project have to be enabled. The instructions below show you the required steps:

1. Navigate to the “Properties” of the configured S7-1500 CPU in the TIA Portal.
2. Navigate to “Protection & Security > Certificate manager” and enable the “Use global security settings for certificate manager” check box.

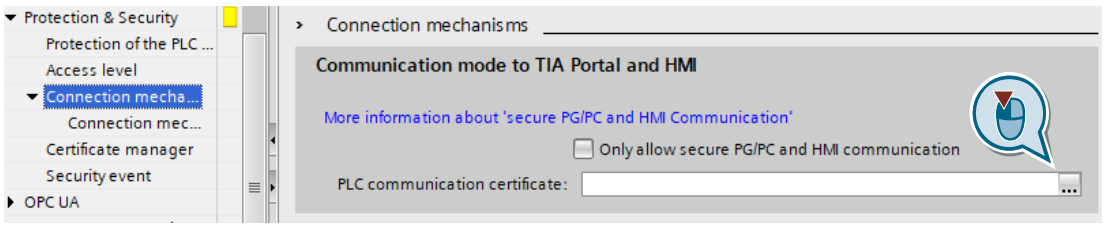


NOTE By checking the check box all device certificates are deleted.

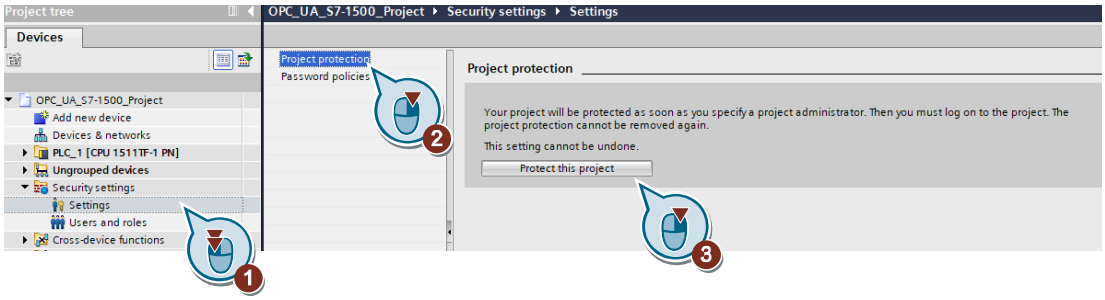
3. Navigate to “OPC UA > Server > Security > Certificates" in the project navigation and create a new server certificate.

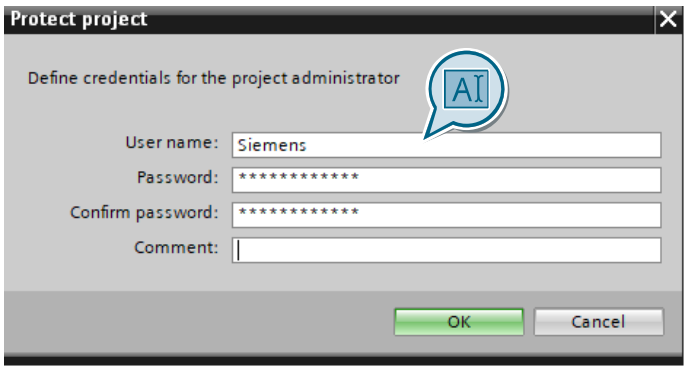


4. Navigate to “Protection & Security > Connection mechanism” and select the created server certificate as "PLC communication certificate".



5. Navigate to “Security setting > Settings > Project protection” in the project navigation and click on the button "Protect this project". Assign a "User name" and a "Password", in order to be able to make security settings in your project.



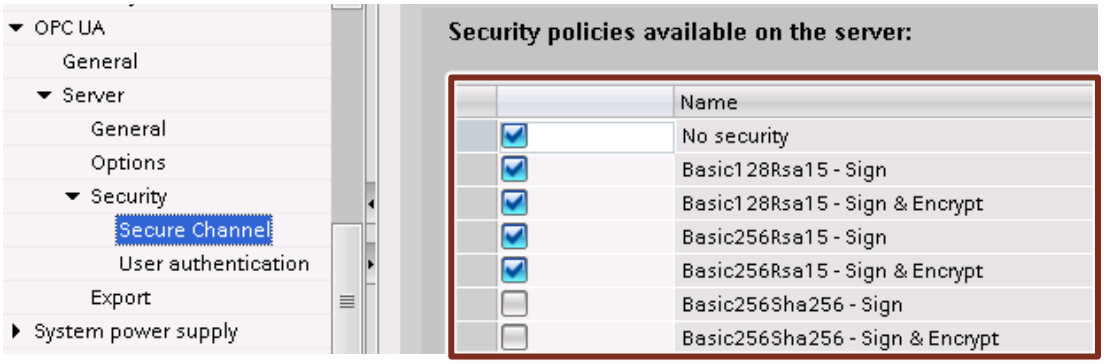


6. Via the assigned user name and the password you can log onto the TIA project to access the certificate manager and other security functions.

2.1.3 Configuring OPC UA security policies (server endpoints)

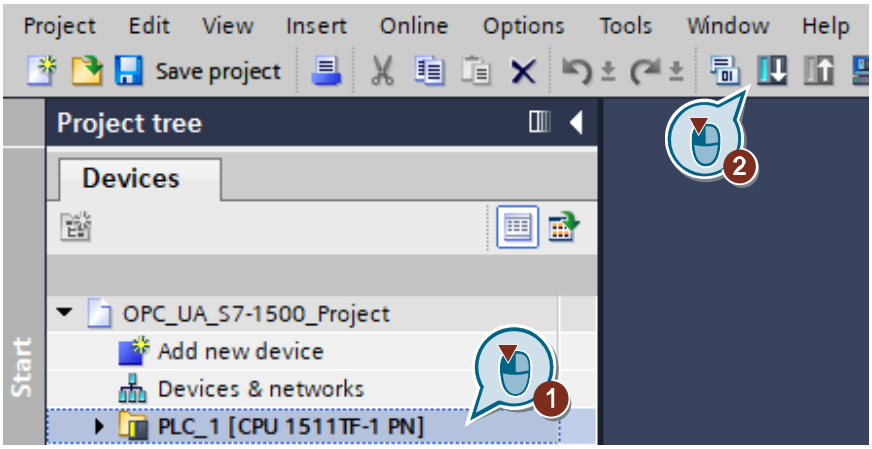
You can configure the way of the encryption and authentication between OPC UA client and server via the security policies of the OPC UA server. The following instruction shows you the required steps to enable the existing security policies:

1. Navigate to the “Properties” of the configured S7-1500 CPU in the TIA Portal.
2. Navigate to “OPC UA > Server > Security > Secure Channel” in the inspector window and select your desired security policies in “Security policies available on the server”. The server creates a separate endpoint for each selected policy to which a client can connect.



NOTE For an OPC UA Client to be able to connect to the endpoints of the OPC UA server it has to support the selected policies.

3. Select the CPU in your project tree and load the project into the controller.



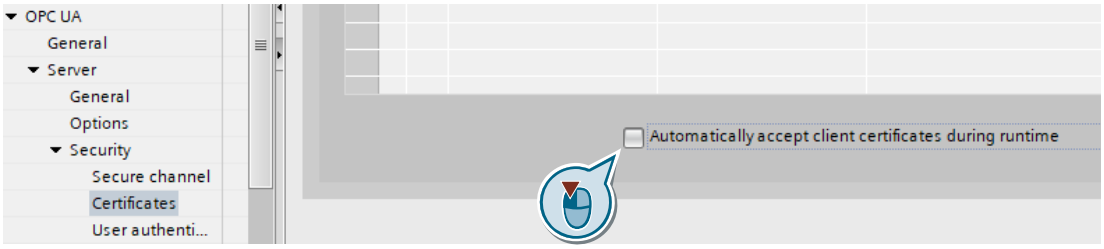
2.1.4 Security via certificate management (optional)

The following prerequisites have to be fulfilled for these settings:

- Global security settings are enabled
- You are logged in to the global security settings
- Client certificates are available.

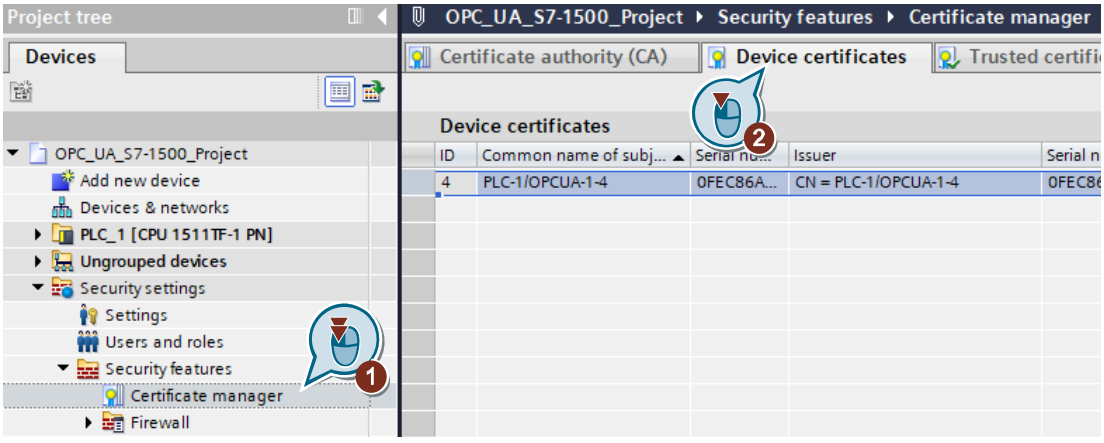
The following instruction shows you what you have to configure, in order to only allow OPC UA clients with defined software certificates to connect to the OPC UA server:

1. Navigate to the “Properties” of the configured S7-1500 CPU in the TIA Portal. Navigate to the “OPC UA > Server > Security > Certificates” inspector window and disable the “Automatically accept all client certificates during runtime” check box in “Trusted clients”.

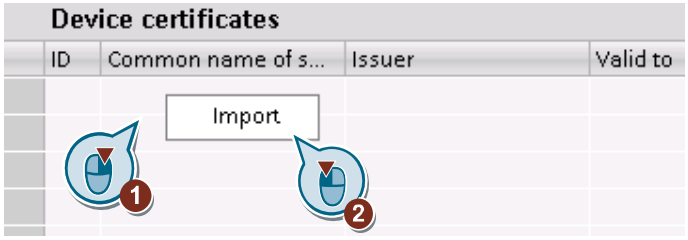


NOTE However, when you enable the security policy “none”, any client can still connect via the appropriate endpoint even without accepted certificate.

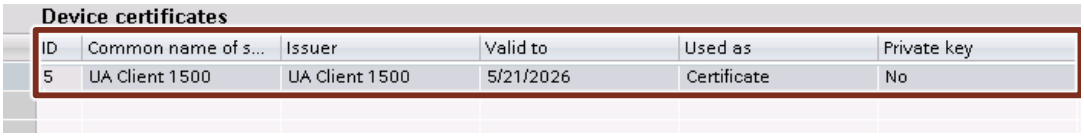
2. Navigate to “Global security settings” in the project navigation and open the “Certificate Manager”. Go to the “Device certificates” tab.



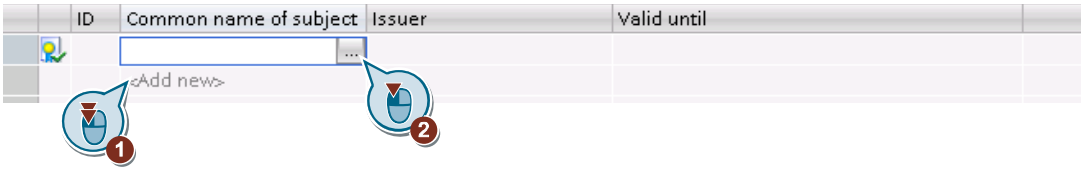
3. Right-click in the work area and then left-click “Import”.



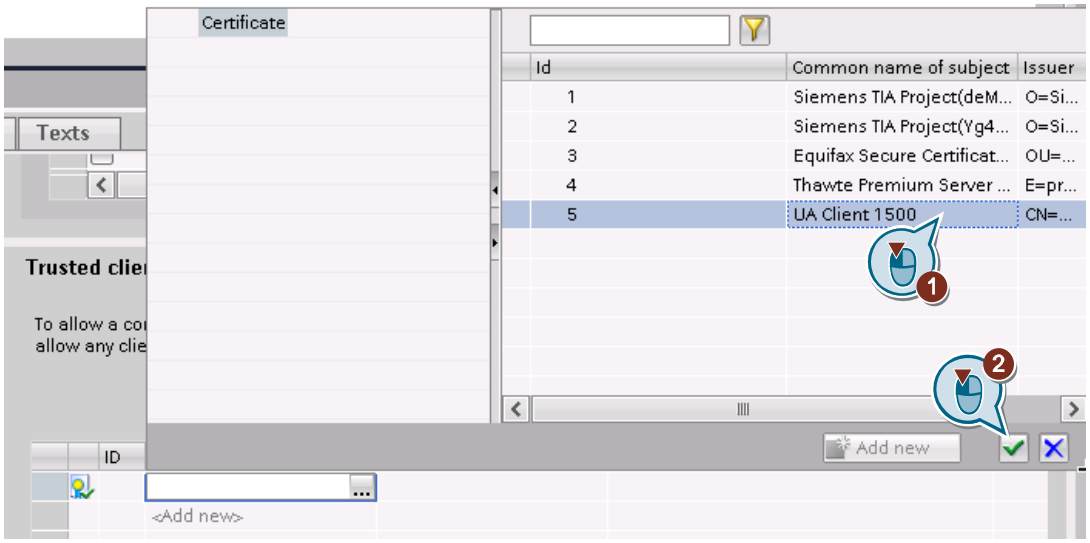
4. Select the software certificate of your OPC UA client via the opened file browser and confirm with “Open”. The imported certificates can then be viewed in the work area.



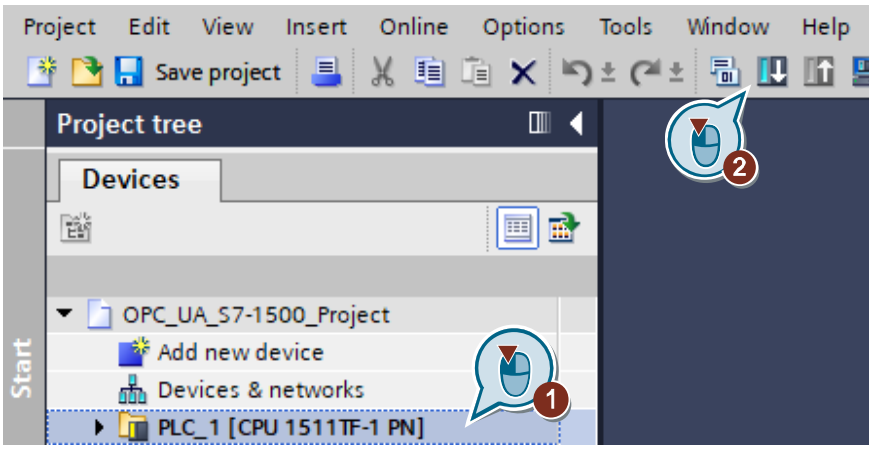
5. Navigate to the “OPC UA > Server > Security > Certificates” inspector window and go to the “Trusted clients” area. Double-click on “<Add new>” in the list and then click the “…” icon.



6. In the dialog that is now open, select the previously imported software certificate of the certificate manager that your OPC UA server is to trust and confirm it with the green tick.



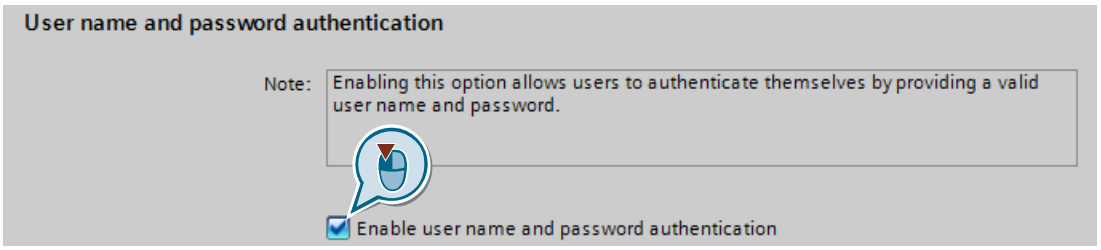
7. Select the CPU in your project tree and load the project into the controller.



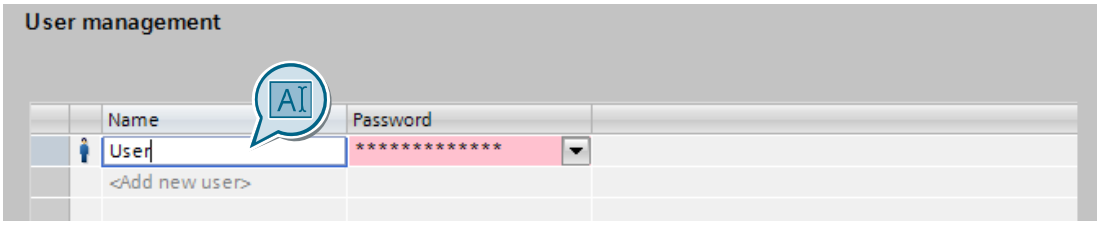
2.1.5 Creating user for the OPC UA server

If you want to create a user on the server for authentication, proceed as follows:

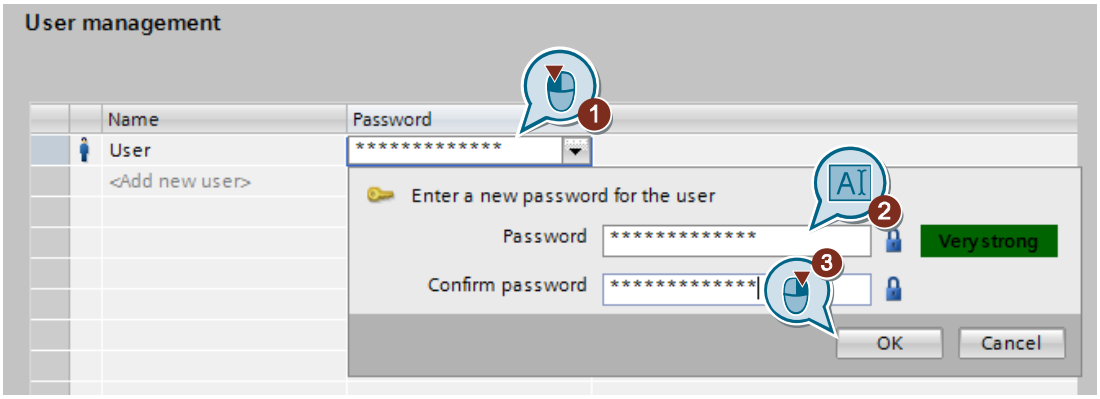
1. Navigate to the “Properties” of the configured S7-1500 CPU in the TIA Portal.
2. Navigate to the “OPC UA > Server > Security > User Authentication” inspector window.
3. Enable the check box "Enable user name and password authentication".



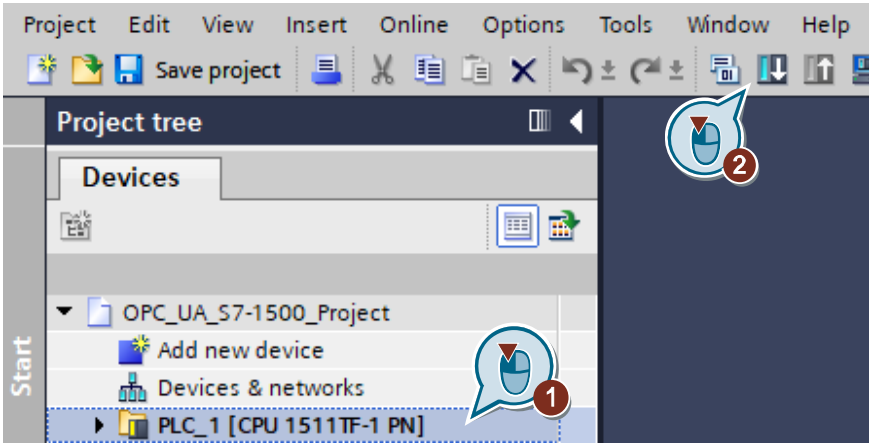
4. Click in the field "<Add new user>" and assign a user name.



5. Then click in the cell of the column "Password" and assign a password for the user in the dialog that follows. Repeat the password and confirm with “OK”.



6. Select the CPU in your project tree and load the project into the controller.

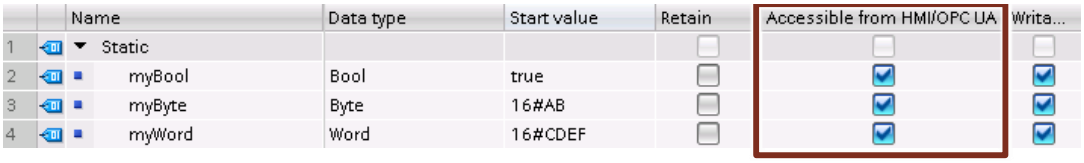


NOTE OPC UA Clients can be authenticated on the server of the S7-1500 via the just created user ID.

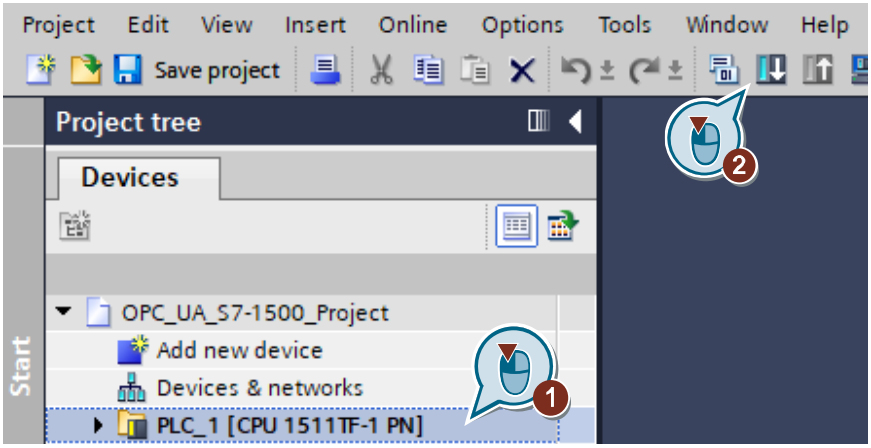
2.1.6 Enabling tags for the OPC UA communication

For each tag (apart from temporary ones) in the S7 user program you can specify individually whether they are to be enabled for the OPC UA communication. The following instruction explains you what you have to do.

1. In your TIA project navigate to the tags you want to have in a FB, DB or the PLC tags.
2. Enable the “Accessible from HMI/OPC UA” check box in the tag declarations.



3. Select the CPU in your project tree and load the project into the controller.



4. The tags modified by you are now writable or readable via OPC UA clients.

2.2 Programming the OPC UA client example

The following descriptions explain the functions and principles of the OPC UA client example program.

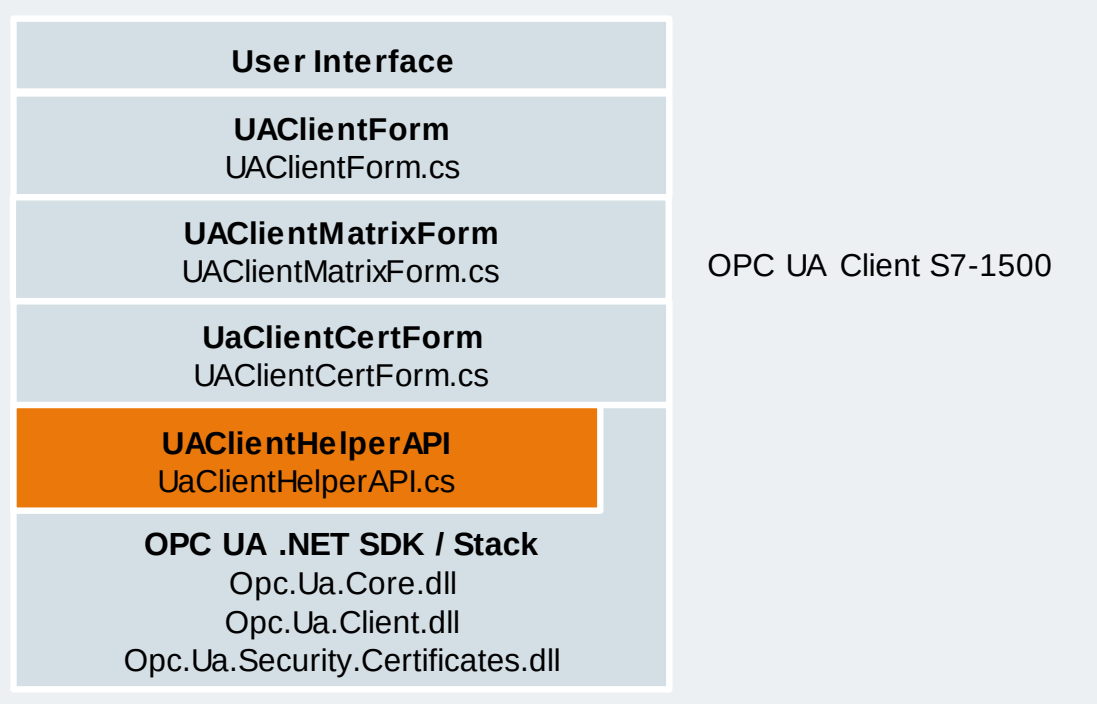
2.2.1 OPC UA Client S7-1500

The OPC UA client example program “OPC UA Client S7-1500” has been created in .NET and requires .NET Framework 4.8.

Structural configuration

The following figure shows the structure of the OPC UA client example of this application example:

Figure 2-1



The “UAClientForm” class is derived from the Windows.Forms system class and includes the form constructor as well as the EventHandlers of the program interface. The methods of the “UAClientHelperAPI” class are accessed in the EventHandlers.

The ”UAClientMatrixForm” is derived form the Windows.Forms. system class and includes the form constructor as well as the EventHandler of the program interface. In the EventHandler the passing of the written values to the “UAClientHelperAPI” class is performed.

The “UAClientCertForm” class is derived from the Windows.Forms system class and includes the form constructor as well as the EventHandlers of the program interface. The certificate management is performed in the EventHandlers.

The “UAClientHelperAPI” class is a user-specific class that summarizes the most important calls of the OPC UA .NET stack. Additionally, private methods are included, in order to create and fill required objects for the OPC UA .NET stack. This class can be expanded and reused as desired and can be used by developers, in order to create simple separate OPC UA clients.

The “OPC UA .NET stack” of the OPC Foundation includes the actual classes/objects that execute and manage the OPC UA communication. The stack consists of a multitude of libraries (DLLs). This application example is only realized via the methods and objects of Opc.Ua.Core.dll, Opc.Ua.Client.dll and Opc.Ua.Security.Certificates.dll. The download of the complete .NET stack as well as its documentation can be found in the links and literature in item \2\.

Using UAClientHelperAPI in the example

In the following table lists the functions in which the public methods of the UAClientHelperAPI are used:

Table 2-1

UAClientHelperAPI	Used within UAClientForm.cs in the method…
FindServers	EndpointButton_Click
GetEndpoints	EndpointButton_Click
Connect	ConnectServerButton_Click EpConnectButton_Click
Disconnect	ConnectServerButton_Click EpConnectButton_Click ClientForm_FormClosing
GetNamespaceUri	-
GetNamespaceIndex	-
GetNamespaceArray	-
BrowseRoot	BrowsePage_Enter
BrowseNode	NodeTreeView_BeforeExpand
BrowseNodeByReferenceType	-
Subscribe	SubscribeButton_Click
AddMonitoredItem	SubscribeButton_Click
AddEventMonitoredItem	-
RemoveMonitoredItem	SubscribeButton_Click
RemoveSubscription	UnsubscribeButton_Click
ReadNode	NodeTreeView_BeforeSelect
ReadNodeAttributes	StructReadButton_Click
WriteValues	WriteValButton_Click RgWriteValButton_Click
ReadValues	ReadValButton_Click RgReadButton_Click
ReadStructUdt	StructReadButton_Click
WriteStructUdt	StructWriteButton_Click
RegisterNodeIds	RegisterButton_Click
UnregisterNodeIds	UnregisterButton_Click
GetMethodArguments	MethodInfoButton_Click
CallMethod	CallButton_Click

The following table lists the EventHandlers in which the public events of the UAClientHelperAPI are to be processed:

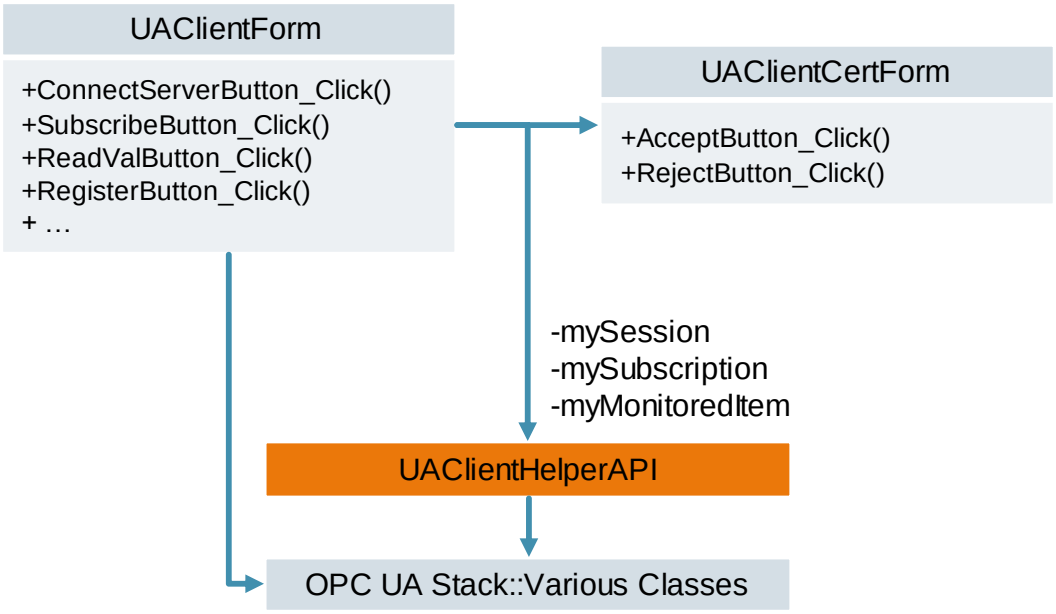
Table 2-2

UAClientHelperAPI	Used within UAClientForm.cs in event handler…
Notification_CertificateValidation	Validator_CertificateValidation
Notification_MonitoredItem	Notification_MonitoredItem
Notification_KeepAlive	Notification_KeppAlive
Notification_MonitoredEventItem	-
Server_ReconnectComplete	-

Class diagram

The following class diagram shows you the classes of the OPC UA client example. The functions of the program interface are implemented by the classes used.

Figure 2-2



2.2.2 UAClientHelperAPI

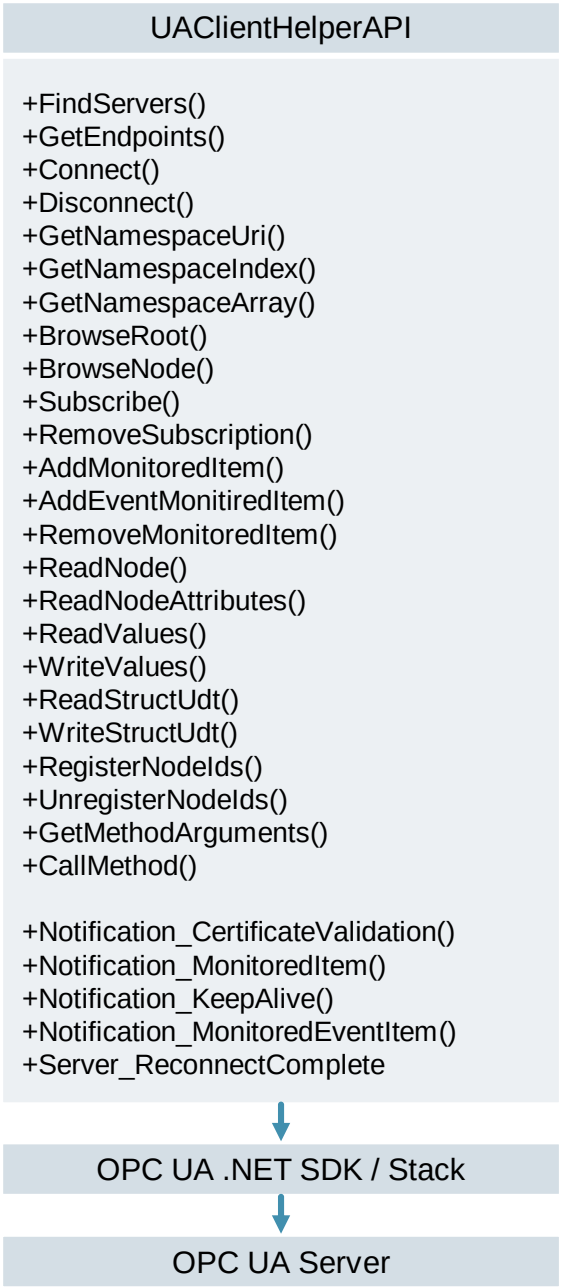
The following explanations describe the reusable “UAClientHelperAPI” class that illustrates the main functionalities of the OPC UA client example.

Class diagram

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

The UAClientHelperAPI accesses the .NET Assembly’s Opc.UA.Client.dll, Opc.UA.Core.dll and Opc.Ua.Security.Certificates.dll of the OPC Foundation.

Figure 2-3



Method description

The following table explains the functions of the public methods within the “UAClientHelperAPI” class, via which the OPC UA client functionalities are realized:

Table 2-3

Method	Explanation
FindServers	Searches for OPC UA servers in the network. Requirement: A LDS (Local Discovery Server) or GDS (Global Discovery Server) has to be available.
GetEndpoints	Determines the available endpoints on a server via which a connection can be established.
Connect	Establishes a connection to a server and creates a secure channel and a session to the server.
Disconnet	Ends an existing session and disconnects the connection to 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 hierarchically found in a specific node.
BrowseByReferenceType	Returns a collection of nodes that can be found in a specific node by a specific reference type.
GetNamespaceUri	Returns the Namespace Uri related to an submitted Namespace Index.
GetNamespaceIndex	Returns the Namespace Index related to an submitted 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 for monitoring an existing subscription.
AddEventMonitoredItem	Adds a (Event)MonitoredItem for monitoring an existing subscription.
RemoveMonitoredItem	Deletes an existing MonitoredItem of a subscription.
ReadNode	Reads the metadata of a specific node.
ReadNodeAttributes	Reads a list of the most important attributes of a specific node.
ReadValues	Reads the values of a variables node.
WriteValues	Writes values to node variables.
ReadStructUdt	Reads values of user-defined structures and UDTs with the help of the "TypeDictionary" of the server.
WriteStructUdt	Writes values to user-defined structures and UDTs that were read before.
RegisterNodeIds	Registers node IDs at the server for an optimized access to the nodes.
UnregisterNodeIds	Deletes the registration of already registered node IDs.
GetMethodArguments	Determines the available input and output arguments of a method.
CallMethod	Calls a method on a server.
Notification_CertificateValidation	Event that is fired when a server certificate cannot be accepted.
Notification_MonitoredItem	Event that is fired when the value of a MonitoredItem is changed.
Notification_KeepAlive	Event that is fired when the value of a KeepAliveNotification arrives.
Notification_MonitoredEventItem	Event that is fired when a OPC UA-Event arrived.
Server_ReconnectComplete	Event that is fired when a reconnect event arrived.

NOTE

In this example, not all available PLC data types can be converted. This restriction affects the methods "ReadValues", "WriteValues", “ReadStructUdt", "WriteStructUdt" and Subscription.

Adjust the source code to your needs.

2.2.3 Sequence diagram for the client example

The following sequence diagrams show the program sequences of the OPC UA example client for various functions of the example.

Establishing and ending connection to the OPC UA server

The following sequence diagram shows the procedures, in order to establish and disconnect the connection to the OPC UA server.

Figure 2-4

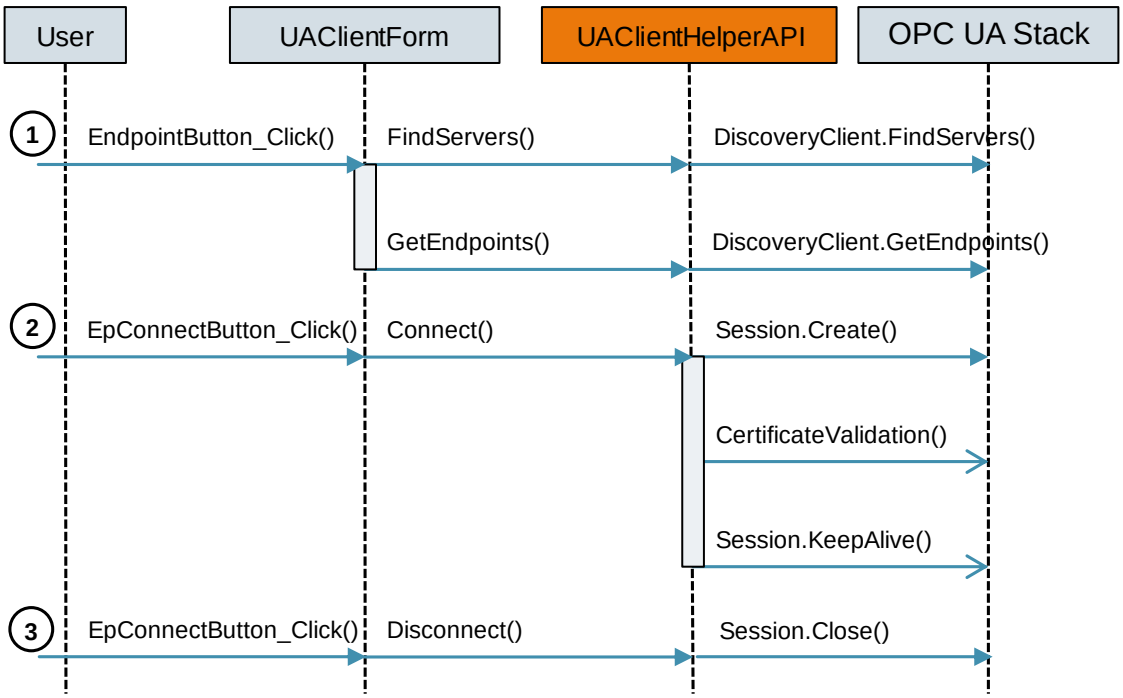


Table 2-4

No.	Description
1.	Via the “Get Endpoints” button, the EndpointButton_Click() method of the interface is called, which in turn calls the FindServers() and GetEndpoints() methods of the UAClientHelperAPI. As a result, the user receives all available endpoints in the network (if a LDS or GDS is available) or of a server.
2.	Once the user has selected an endpoint from the list and has clicked the “Connect to server” button, the EpConnectButton_Click() UI method is called. This calls the Connect() method of the UAClientHelperAPI. A session object and another object for the Session.Create() method is created and filled. Once the Session.Create() is executed, it is responded to an arriving CertificateValidation event via which the server certificate can be validated.
3.	When the button is clicked again the EpConnectButton_Click() method is executed again. If a connection already exists, the Disconnect() method of the UAClientHelperAPI is executed. This ends the existing session via the OPC UA stack Session.Close() method.

Browsing on the OPC UA server

The following sequence diagram shows the procedures, in order to browse nodes in the OPC UA address space of the server.

Figure 2-5

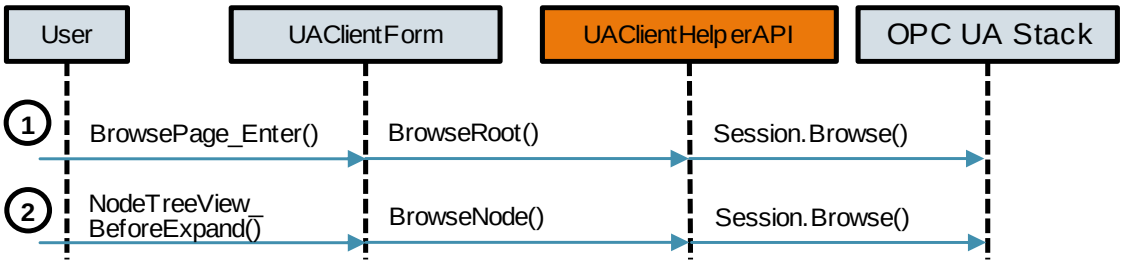


Table 2-5

No.	Description
1.	As soon as the user in the client example goes to the “Browse Nodes” tab, the BrowsePage_Enter() UI method is called. This calls the BrowseRoot() method of the UAClientHelperAPI. In this, the Session.Browse() method of the OPC UA stack is called with the suitable transfer parameters, in order to browse the root node of the server.
2.	When a node of the tree view of the address space is be expanded, the NodeTreeView_BeforeExpand() UI method is called. This calls the BrowseNode() method of the UAClientHelperAPI. In this, the Session.Browse() method of the OPC UA stack is called with the suitable transfer parameters, in order to browse any node of the server.

Reading and writing tags

The following sequence diagram shows the procedures, in order to read or write tags:

Figure 2-6

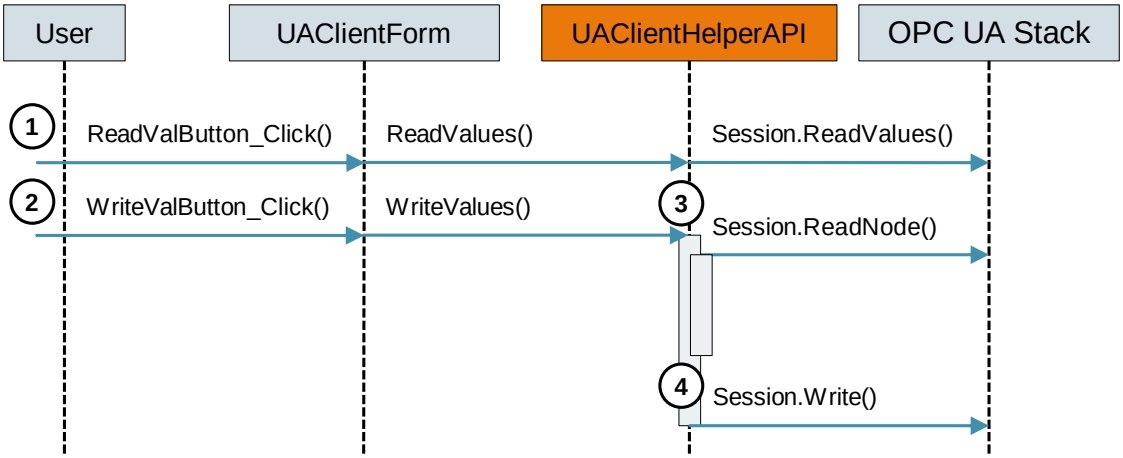


Table 2-6

No.	Description
1.	When you click the “Read” button, the ReadValButton_Click() UI method is called. It transfers the ReadValues() method a list of node ID strings to the UAClientHelperAPI. From the transmitted string list, a node ID list is created and transferred to the Session.ReadValues() OPC UA stack method. This method reads all values of the node IDs of the list and returns them.
2.	When you click the “Write” button, the WriteValButton_Click() UI method is called. This transfers a dictionary of value pairs of node IDs and their related values as string to the WriteValues() method of the UAClientHelperAPI.

No.	Description
3.	Node IDs are created from the transmitted strings. These are read via the Session.ReadNode() method, in order to determine their data types.
4.	The Session.Write() method writes the values to the server via the determined data types and node IDs.

Reading and writing structures/UDTs

The following sequence diagram shows the procedures, in order to read or write structures/UDTs:

Figure 2-7

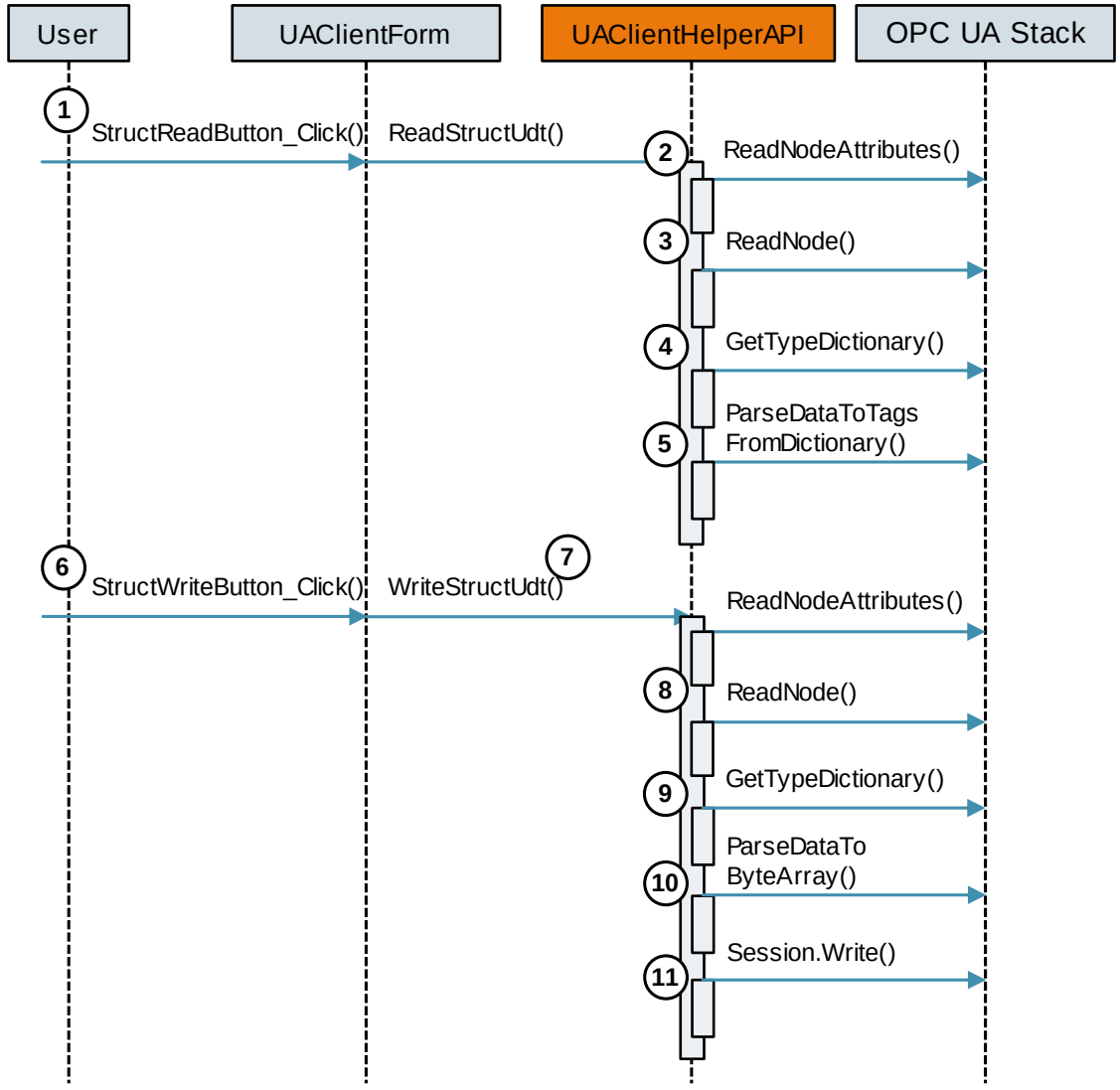


Table 2-7

No.	Description
1.	When you click the “Read Struct/UDT” button, the StructReadButton_Click() UI method is called. This transfers a list of node ID strings to the ReadStructUdt() method of the UAClientHelperAPI.
2.	The "ReadNodeAttributes" method reads out the attributes of the node ID based on the transferred node ID. Of these, the "Value" and "DataType" attributes are used.
3.	Based on the determined attribute for the data type, the data type node is read out by the "ReadNode" method. You will receive a node ID of the type "DataTypeNode".

No.	Description
4.	Based on the data type Node the method GetTypeDictionary() reads the TypeDictionary of the server. The dictionary is created as follows: 1. The previously read data Type Node contains in its body the information about the structure with the class "StructureDefinition". 2. By using the object of the class "StructureDefinition" a "StructureFieldCollection" is obtained, which contains all information about the structure fields 3. By checking the "NamespaceIndex" of the individual structure fields in the "StructureFieldCollection" it is determined whether further structures/UDTs are contained in the parent structure/UDT. If this is the case, the node ID of the corresponding structure field is read and the function "GetTypeDictionary()" is called recursively. 4. Each structure field of the base data type is added to a list of string arrays with the information about the data type and the array dimensions. The method returns a list with all structure fields, the "TypeDictionary", as well as their data types and array dimensions.
5.	Based on the determined "TypeDictionary" and the attribute "Value", which is read in a structure as a byte array, the method "ParseDataToTagsFromDictionary" deserializes the byte array according to the individual fields and their data type from the "TypeDictionary". The method returns a list of string arrays, these contain the information, array index, tag name, tag value and tag data type.
6.	When you click the “Write Struct/UDT” button, the StructWriteButton_Click() UI method is called. It transfers the entered process values and a node ID to the WriteStructUdt() method of the UAClientHelperAPI.
7.	The "ReadNodeAttributes" method reads out the attributes of the node ID based on the transferred node ID. Of these, the "Value" and "DataType" attributes are used in particular.
8.	Based on the determined attribute for the data type, the data type node is read out by the "ReadNode" method. You will receive a node ID of the type "DataTypeNode".
9.	The method GetTypeDictionary() reads the TypeDictionary of the server from the data type node. The method returns a list with all structure fields, the "TypeDictionary" and their data types and array dimensions.
10.	Based on the determined "TypeDictionary" and the list of the UI entered process values the method "ParseDataToByteArray" encodes the string values of the list from the UI according to the actual data types from the "TypeDictionary" into a byte array. The method returns a byte array.
11.	The Session.Write() method writes the values to the server via the Node Id.

Registering node IDs

The following sequence diagram shows the procedures, in order to register node IDs on the OPC UA server. Optimized read and write operations can be executed via these registered IDs.

Figure 2-8

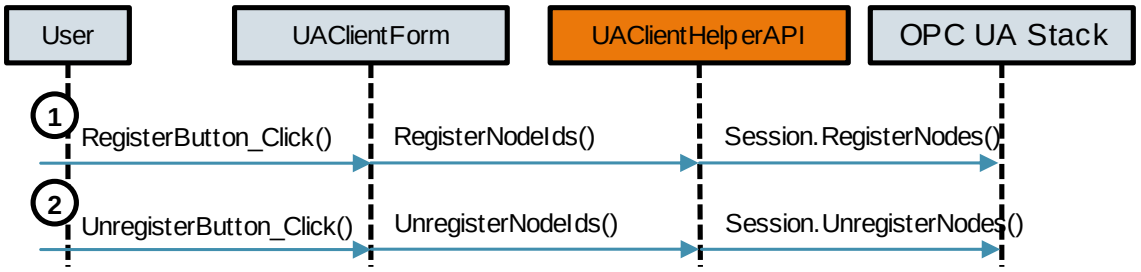


Table 2-8

No.	Description
1.	When you click the “Register” button, the RegisterButton_Click() UI method is called. This transfers the RegisterNodeIds() method a list of node ID strings to the UAClientHelperAPI. From the transmitted string list, a node ID list is created and transferred to the Session.RegisterNodes() OPC UA stack method. This method registers all node IDs of the list.
2.	When you click the “Unregister” button, the UnregisterButton_Click() UI method is called. This transfers the UnregisterNodeIds() method a list of node ID strings to the UAClientHelperAPI. From the transmitted string list, a node ID list is created and transferred to the Session.UnregisterNodes() OPC UA stack method. This method cancels the registration of all transferred node IDs.

Subscribing/ending subscriptions

The following sequence diagram shows the procedures, in order to subscribe or end subscriptions to certain tags:

Figure 2-9

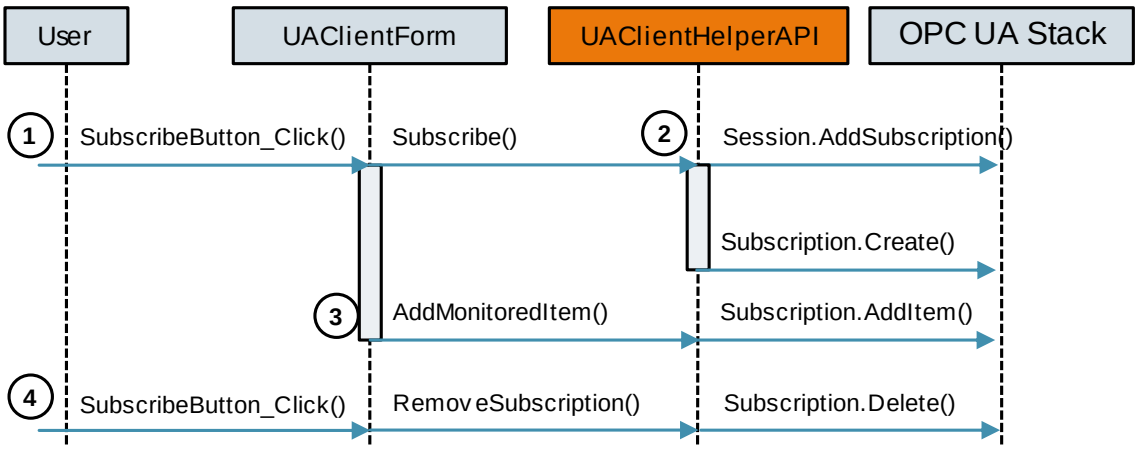


Table 2-9

No.	Description
1.	When you click the “Subscribe” button, the SubscribeButton_Click() UI method is called. This calls the Subscribe() method of the UAClientHelperAPI.
2.	Within the Subscribe() method, a subscription is created, transferred to the session (Session.AddSubscription) and enabled (Subscription.Create()).
3.	Within the SubscribeButton_Click() UI method a monitored item is created once a subscription has been created and transferred to the AddMonitoredItem() method of the UAClientHelperAPI. This adds the transferred MonitoredItem via the Subscription.AddItem() method to the subscription.
4.	When you click the “Subscribe” button again, the UnregisterButton_Click() UI method is called. This calls the RemoveSubscription() method of the UAClientHelperAPI. Within this method, the Subscription.Delete() UA stack method is called that ends the subscription on the server.

Calling methods

The following sequence diagram shows the procedures in order to call methods on the server.

Figure 2-10

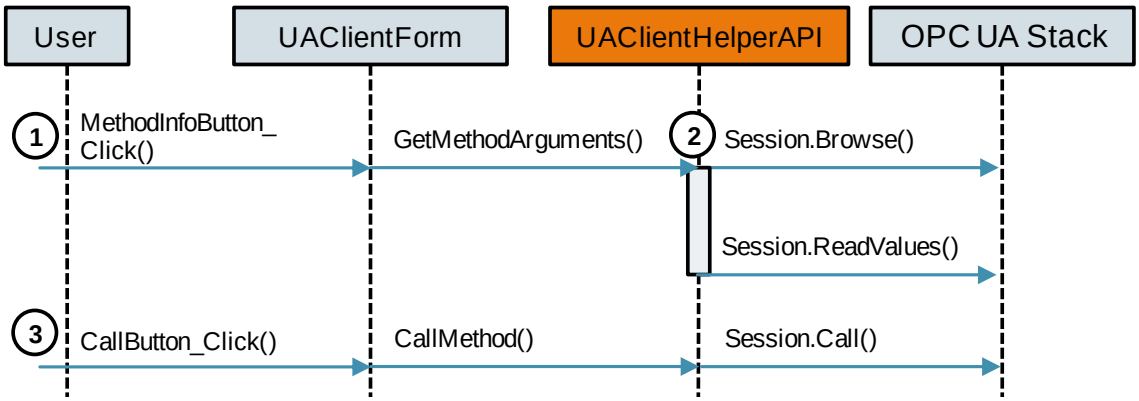


Table 2-10

No.	Description
1.	When you click the "Get Method Info" button, the MethodInfoButton_Click() UI method is called. This calls the GetMethodArguments() method of the UAClientHelperAPI.
2.	Within the GetMethodArguments() method, Session.Browse() is used for browsing to the nodes of the input and output arguments. Subsequently, the arguments are read via Session.ReadValues(). In doing so, the names and data types of the arguments are determined.
3.	When you click the “Call Method” button, the CallButton_Click() UI method is called. This calls the CallMethod() method of the UAClientHelperAPI and transfers the previously determined input arguments as well as the values that you have assigned for the arguments on the user interface. After the call, the result of the method is converted in accordance with the determined output arguments and returned.

2.3 Operation

The following step-by-step instructions show you how you can commission the application example and how you can operate it.

2.3.1 Description of the user interface

The user interface of the “OPC UA Client S7-1500” example client is divided in five tabs:

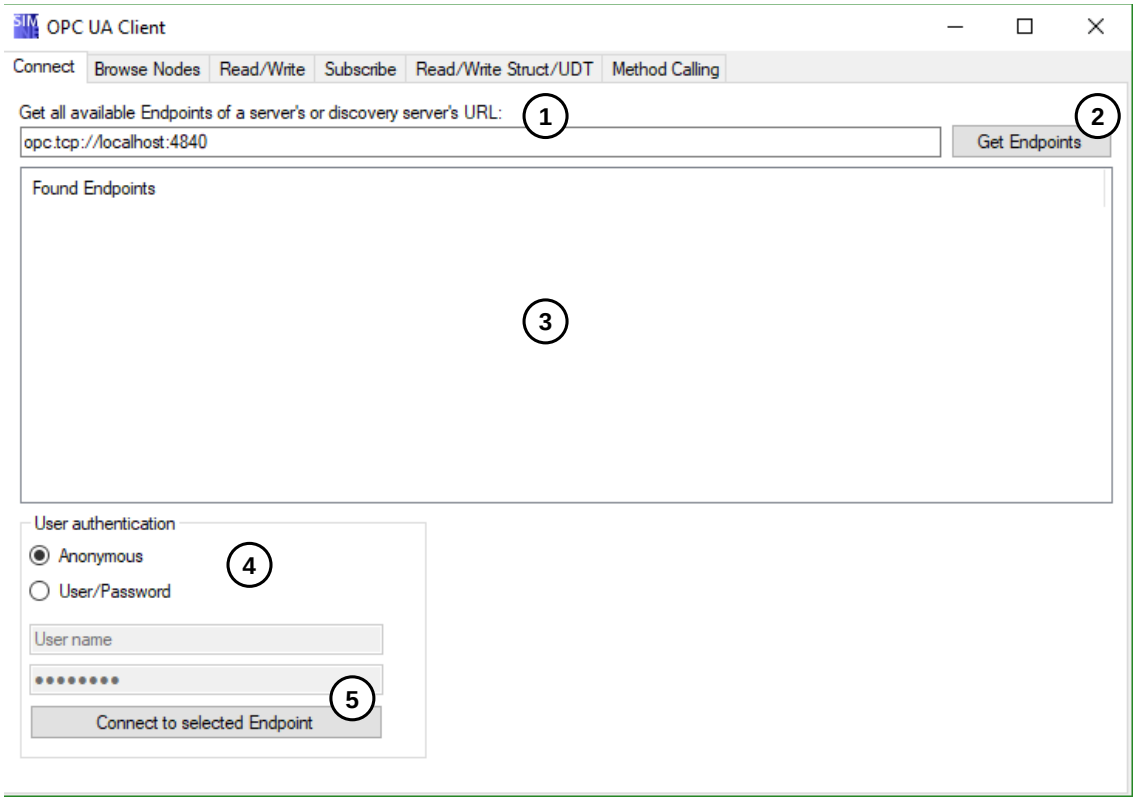
- "Connect"
- "Browse Nodes"
- "Read/Write"
- "Subscribe"
- "Read/Write Struct/UDT"
- "Method Calling"

The descriptions below explain the individual tabs in more detail:

"Connect"

The following figure shows the interface of the “Connect” tab:

Figure 2-11



The following table describes the functions of the interface of the previous figure:

Table 2-11

No.	Description
1.	Text field to enter an OPC UA (Discovery) server URL.
2.	Button to search OPC UA endpoints with the URL from the text field (1).
3.	List of the OPC UA endpoints found.
4.	Settings for user authentication.

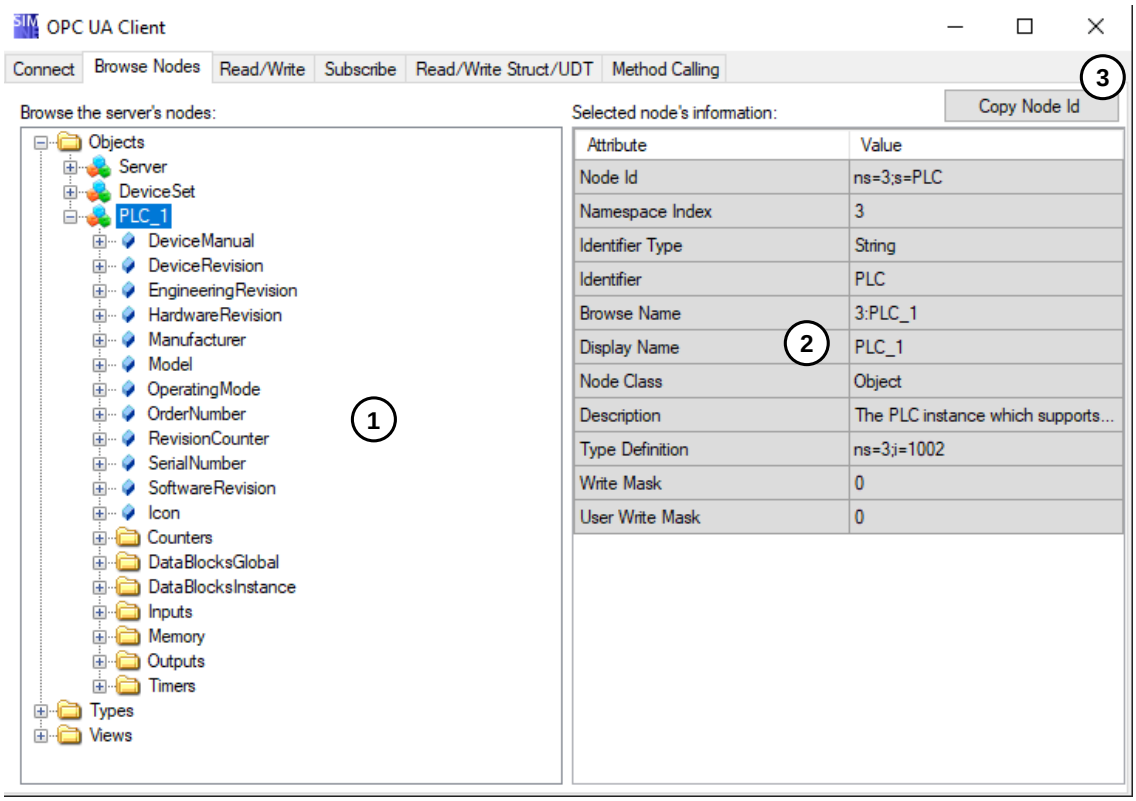
No.	Description
5.	Button to establish a connection to a selected endpoint of the list (3).

NOTE In order to access the “Browse Nodes”, “Read/Write” and “Subscribe” tabs, you have to be connected with an OPC UA server.

"Browse Nodes"

The following figure shows the interface of the “Browse Nodes” tab:

Figure 2-12



The following table describes the functions of the interface of the previous figure:

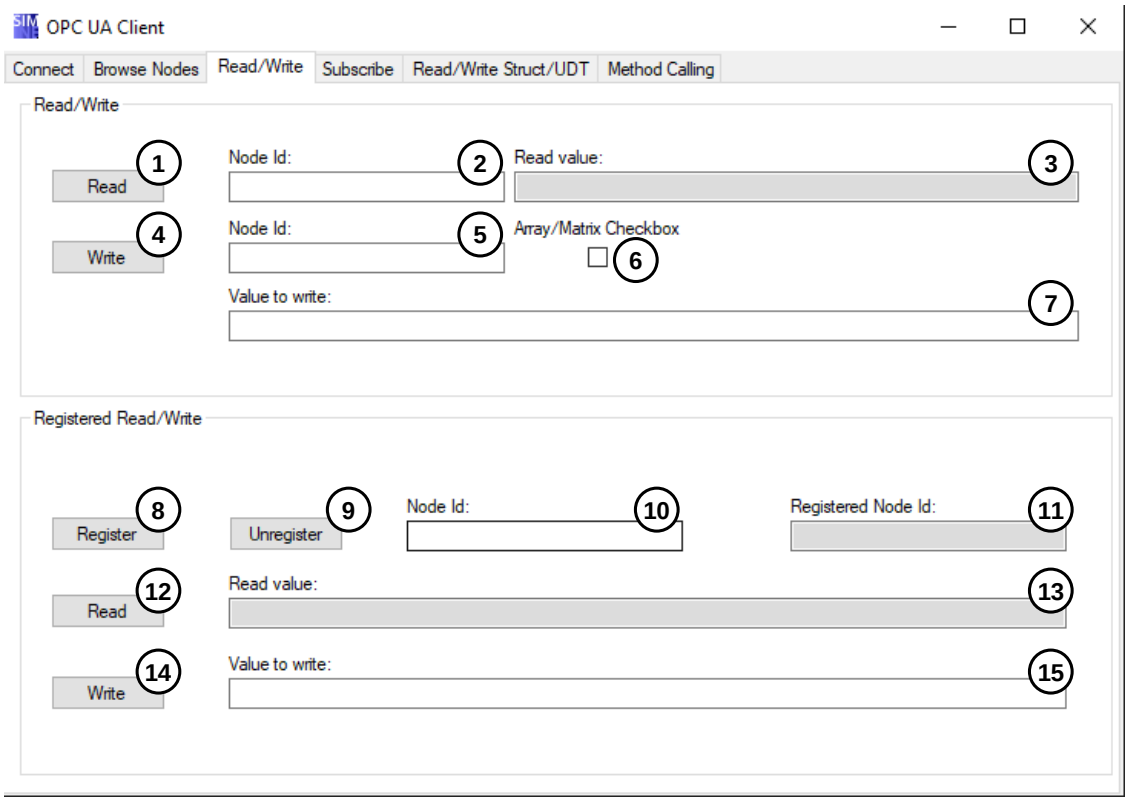
Table 2-12

No.	Description
1.	Tree view of the available nodes on the OPC UA server.
2.	Data view of the attributes of a selected node from the tree view (1).
3.	Button to copy the node ID of the data view (2) to the clipboard.

"Read/Write"

The following figure shows the interface of the “Read/Write” tab.

Figure 2-13



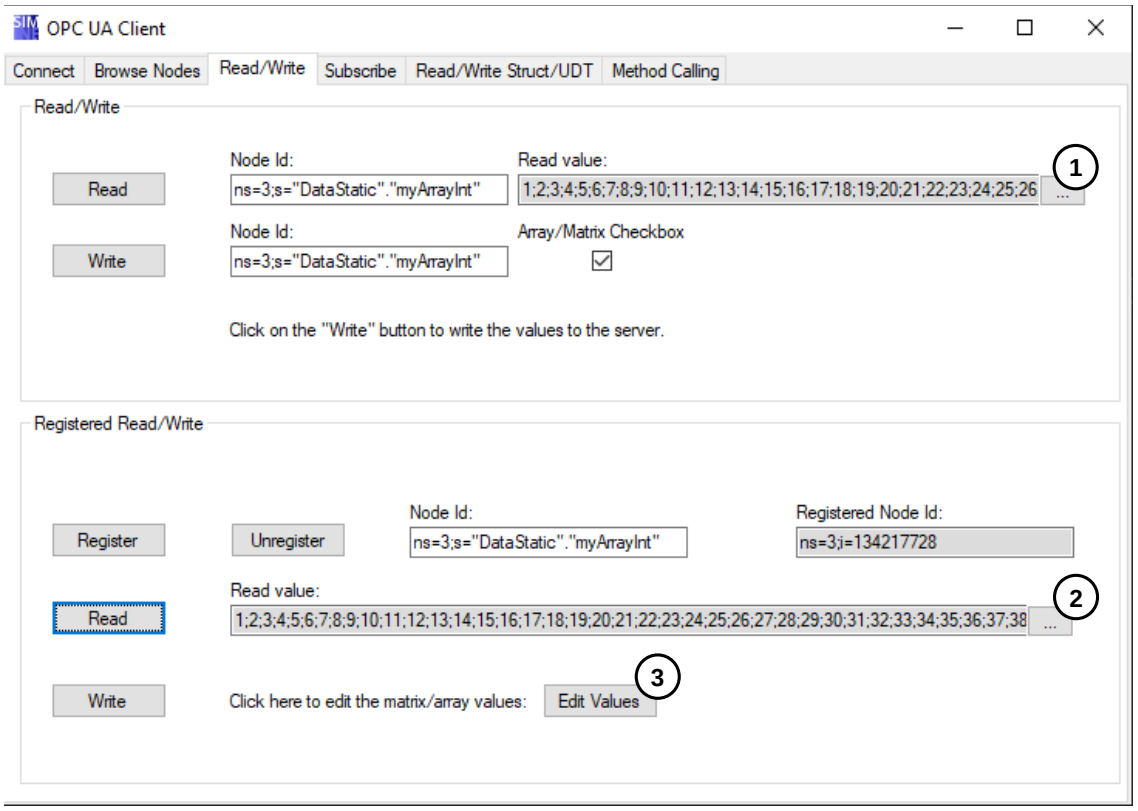
The following table describes the functions of the interface of the previous figure:

Table 2-13

No.	Description
1.	Button to read an entered node ID from text field (2).
2.	Text field to enter a node ID to be read.
3.	Text field to output the value of the read node ID from text field (2).
4.	Button to write an entered node ID from text field (5).
5.	Text field to enter a node ID to be written.
6.	Checkbox to select whether the node ID is an array/matrix or not.
7.	Text field to enter the value of the node ID to be written from text field (5).
8.	Button to register an entered node ID from text field (9).
9.	Button to cancel the registration of an entered node ID from text field (9).
10.	Text field to enter a node ID to be registered.
11.	Text field to output a registered node ID.
12.	Button to read a registered node ID.
13.	Text field to output the value of the read, registered node ID from text field.
14.	Button to write on a registered node ID.
15.	Text field to enter the value of the node ID to be written.

The following figure shows the interface of the “Read/Write” tab by entering a node ID of the type array/matrix.

Figure 2-14



Buttons 1 - 3 allow you to display or enter the array/matrix values.

The following figure shows the interface of the “Read/Write” tab for entering array/matrix values:

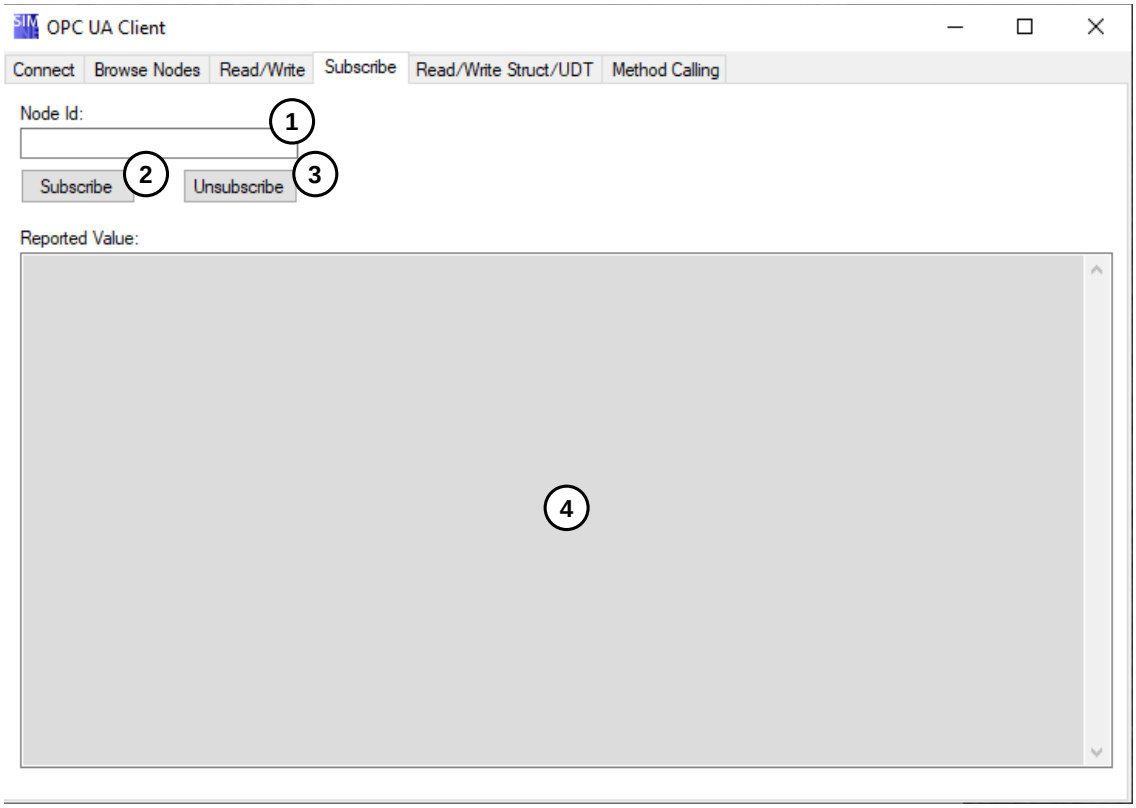
Figure 2-15

The button enables the entering of array/matrix values and closes the view.

"Subscribe"

The following figure shows the interface of the “Subscribe” tab.

Figure 2-16



The following table describes the functions of the interface of the previous figure:

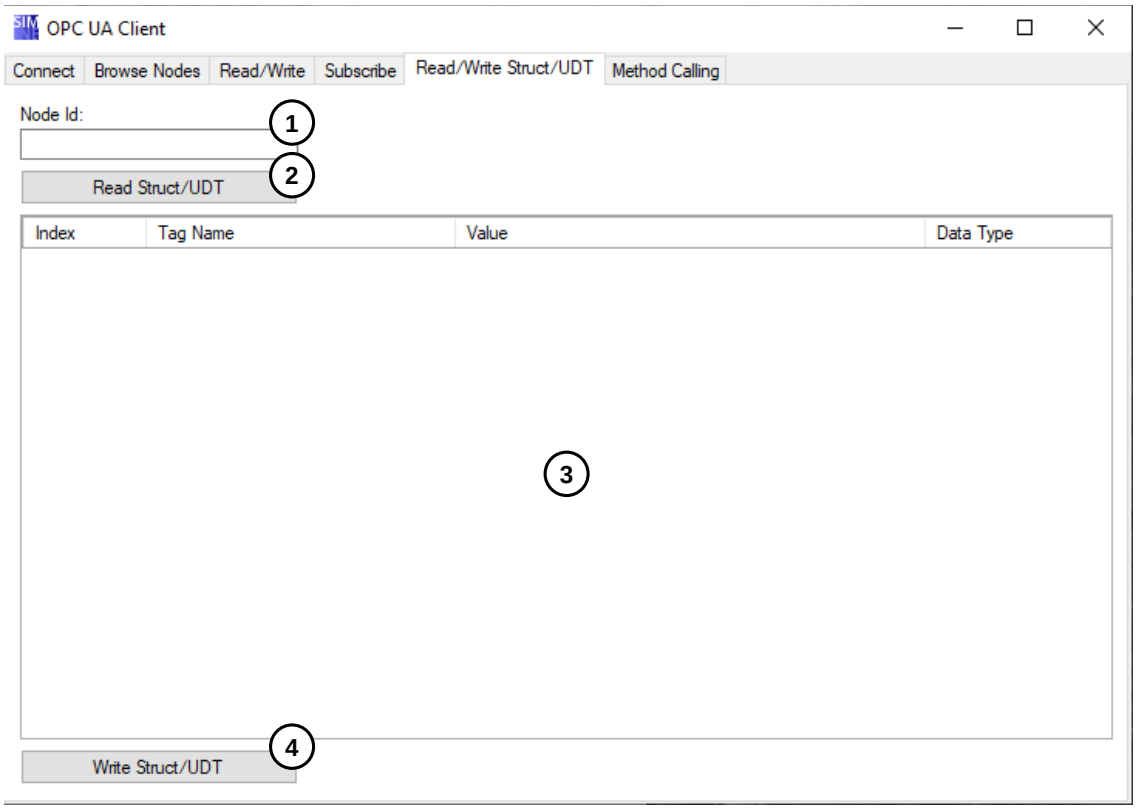
Table 2-14

No.	Description
1.	Text field to enter a node ID that is added to the subscription as MonitoredItem.
2.	Button to start a subscription on an OPC UA server and to create a MonitoredItem via the node ID from text field (2).
3.	Button to end a subscription and to delete the MonitoredItem.
4.	Text field to output the value of the MonitoredItem of the subscription with time stamp and status.

"Read/Write Struct/UDT"

The following figure shows the interface of the “Read/Write Struct/UDT” tab.

Figure 2-17



The following table describes the functions of the interface of the previous figure:

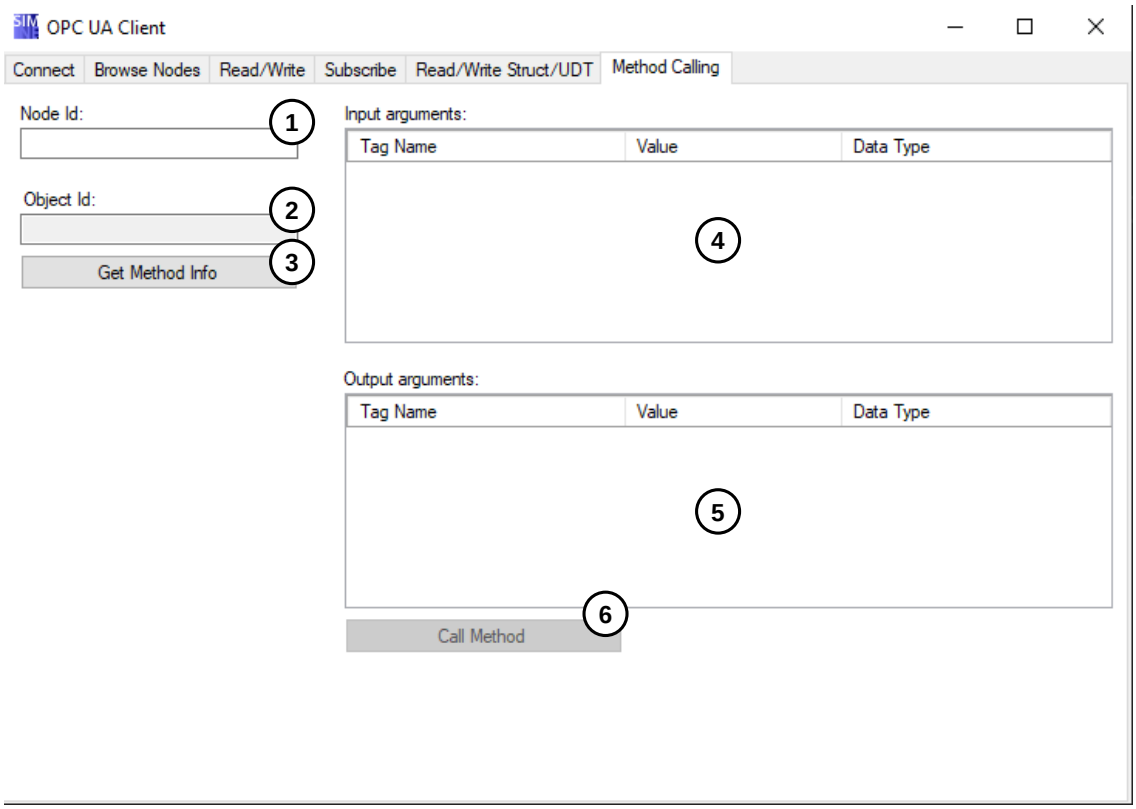
Table 2-15

No.	Description
1.	Text field to enter a node ID which references to a read/write structure/UDT.
2.	Button to read an entered node ID from text field (1).
3.	Data view of the read structure/UDT of an entered node from the text field (1).
4.	Button to write on the read node ID from text field (1).

"Method Calling"

The following figure shows the interface of the “Method Calling” tab.

Figure 2-18



The following table describes the functions of the interface of the previous figure:

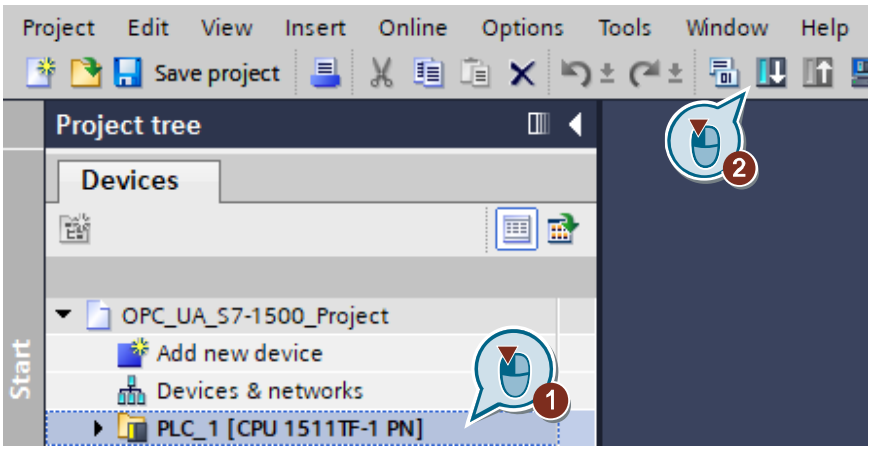
Table 2-16

No.	Description
1.	Text field to enter a node ID which references to a method.
2.	Text field to show the node ID which references to an object including a method.
3.	Button to determine the input and output arguments of the method.
4.	List of input arguments.
5.	List of output arguments.
6.	Button to call the method.

2.3.2 Commissioning the OPC UA server of the S7-1500

Carry out the configuration steps in chapter 2.1 "Configuring the OPC UA server of the S7-1500" or download the pre-prepared TIA Portal project into your controller. Proceed as follows:

1. Download the "109737901_OPC_UA_Client_S7-1500_CODE_V13.zip" project onto your hard drive. The download can be found on the HTML page of this entry (\1\).
2. Unzip the project.
3. Navigate to “OPC_UA_Server_1500” in the unzipped folder. The TIA Portal project is located in this folder.
4. Open the project by double-clicking the "UA Server 1500.ap17” file.
   - User name: "User"
   - Password: "Siemens.1"
5. Select the CPU in the project tree and load the project into the controller.



2.3.3 Commissioning the OPC UA Client S7-1500

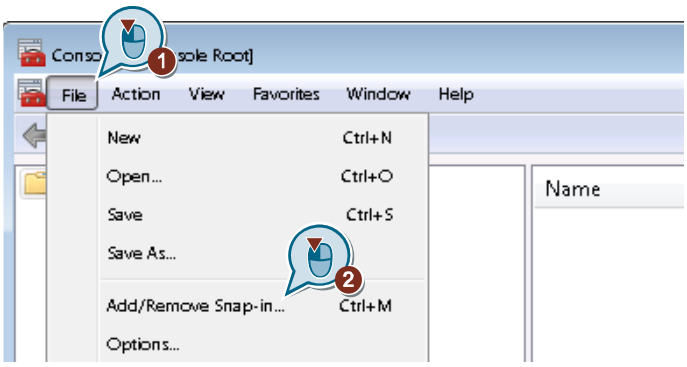
The example client is provided via a Visual Studio project.

1. Download the "109737901_OPC_UA_Client_S7-1500_CODE_V15.zip" project onto your hard drive. The download can be found on the HTML page of this entry (\1\).
2. Unzip the project.
3. Navigate to “OPC_UA_Client_1500 > UA_Client_1500 > Application” in the unzipped folder. The compiled executable .EXE file of the OPC UA example client is located in this folder.
4. Start the program by double-clicking the “UA Client 1500.exe” file.

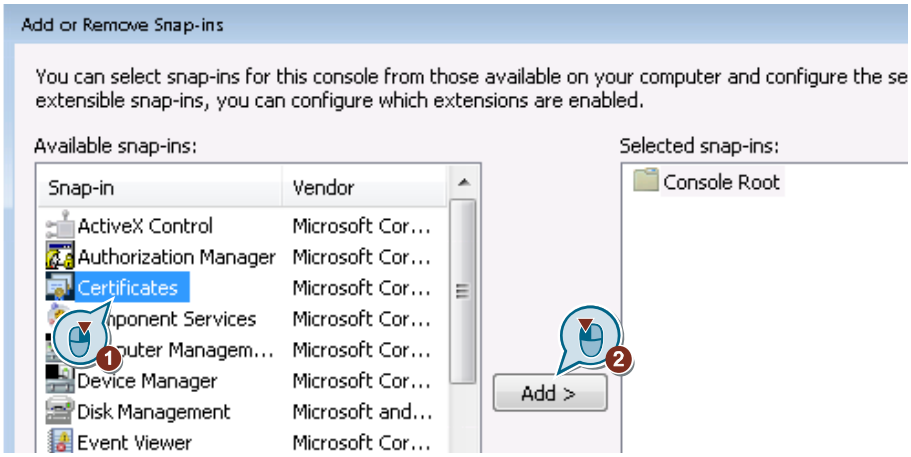
2.3.4 Creating, exporting and loading client certificate into the S7-1500 (optional)

If you want to increase the security of your application via the certificate management, please follow the following steps:

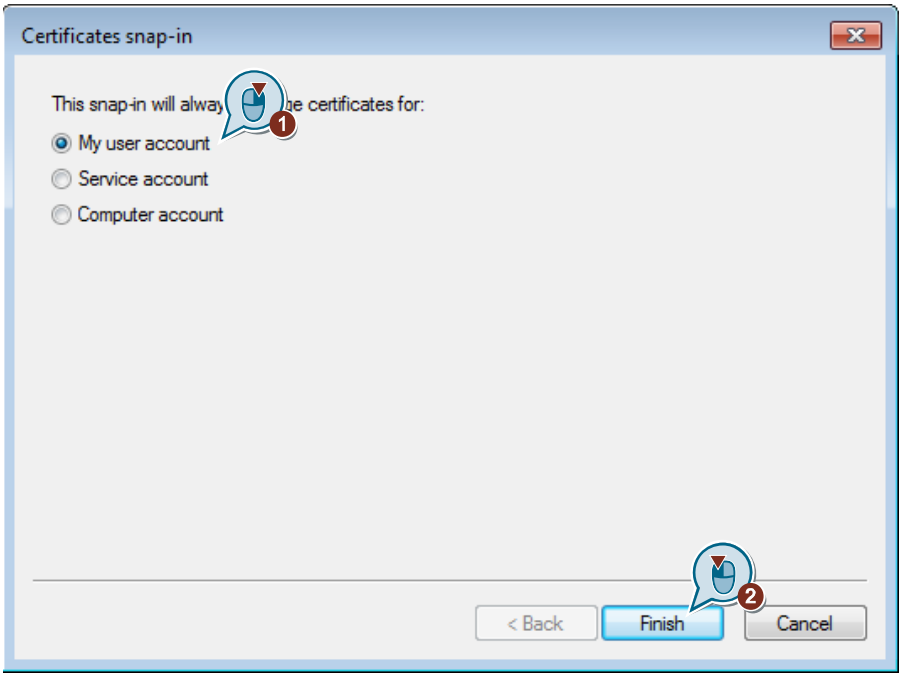
1. Start the OPC UA Client S7-1500.
2. During the first program start, a self-signed software certificate of the client program is created and stored in the Windows Certificate Store. This certificate has to be known by the OPC UA server if you want to communicate signed and encrypted with a server.
3. Click on “Start > Run” in Windows and enter “mmc". Confirm with the “Enter” button.
4. In the now opened certificate store, click “File > Add/Remove Snap-in…”.



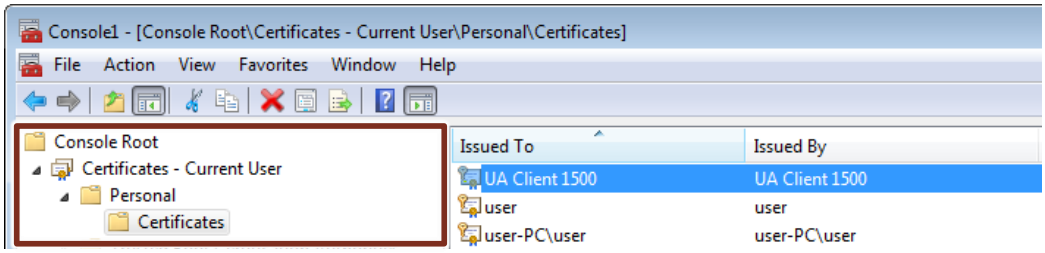
5. Search for “Certificates” in the dialog that appears, select it and click “Add >”.



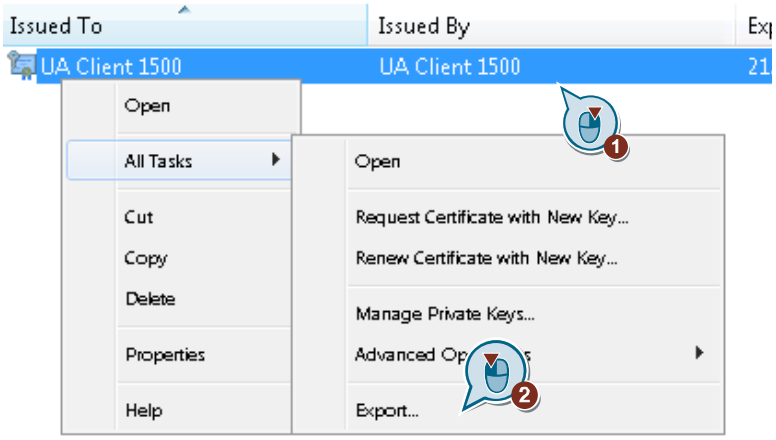
6. Select the “My user account” check box and confirm with “Finish”.



7. Then click “OK” to confirm.
8. Navigate from the “Console Root” to "Certificates - Current User > Personal > Certificates".

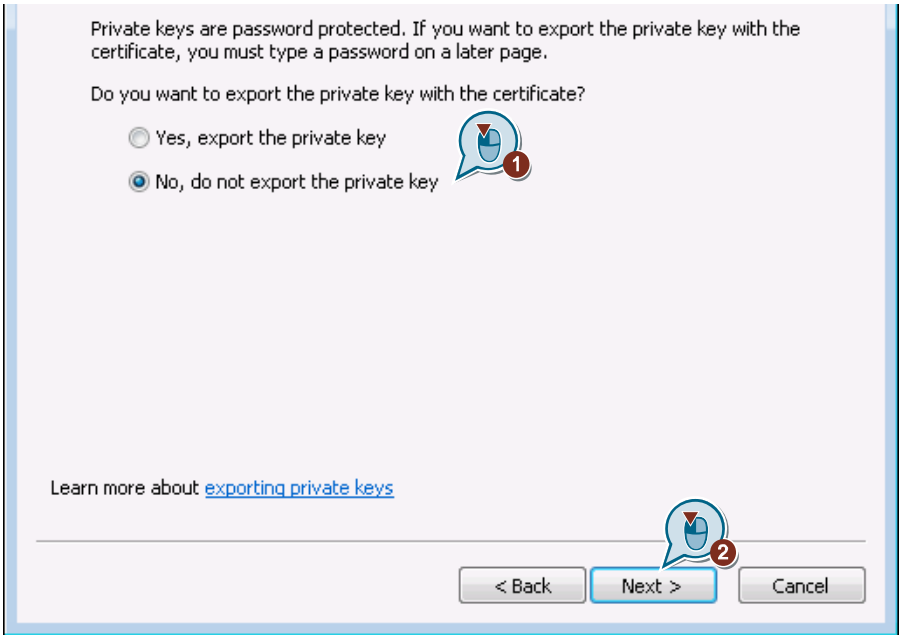


9. Right-click the “UA Client 1500” certificate that has been created by the example client and navigate to “All Tasks” in the context menu. Then click on “Export…”.

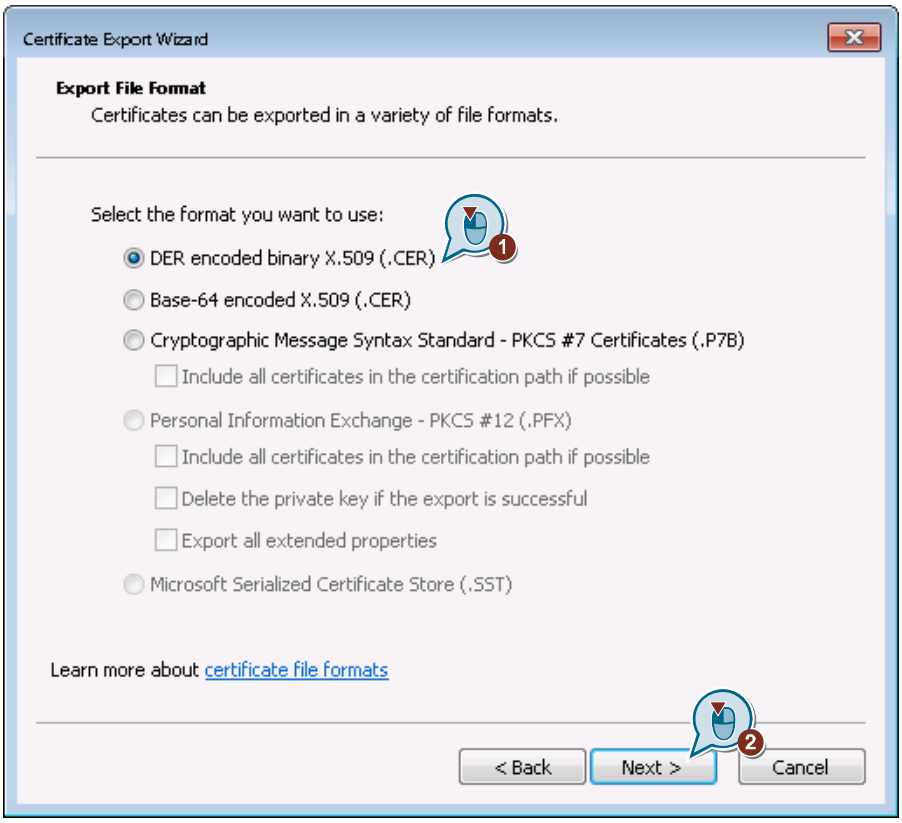


10. In the dialog that appears then, click “Next >”.

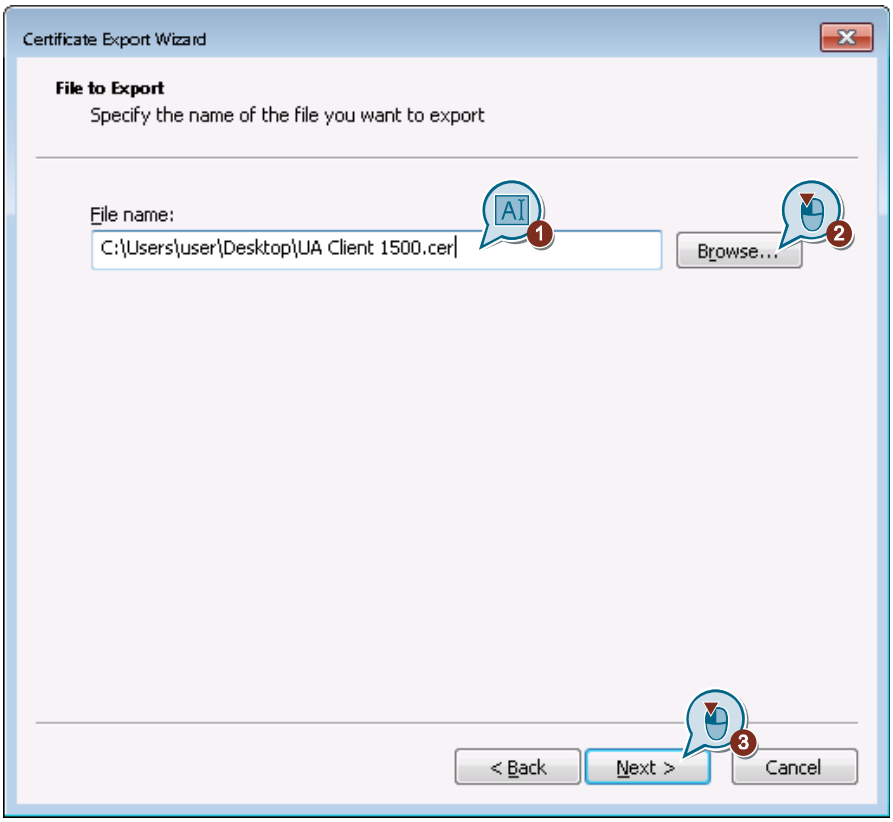
11. Select the “No, do not export private key” check box and click on “Next >”.



12. Select the “DER encoded binary X.509 (.CER)” check box and then click “Next >”.



13. Select a suitable storage location for the certificate via “Browse…” and assign a file name. Then click on “Next >”.



14. Click “Finish” in the dialog that follows.
15. The certificate is now stored in the selected storage location and can be imported into the TIA Portal or into other OPC UA servers from there.
16. (Optional) Follow the configuration instructions in chapter 2.1.4 “Security via certificate management (optional)”.

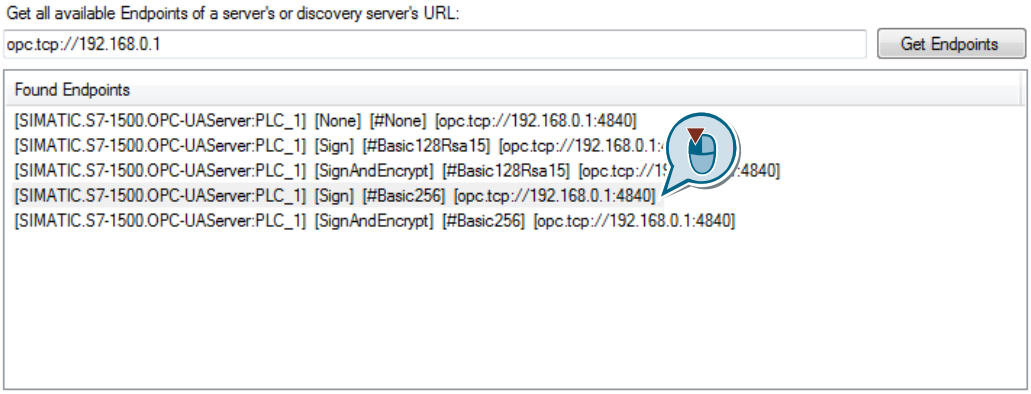
2.3.5 Establishing a connection to the OPC UA server

Connect to the example client with an OPC UA server via the endpoint browser. Proceed as follows:

1. Enter the URL of an OPC UA server, LDS or GDS into the text field. Click the “Get Endpoints” button.

NOTE If you do not specify a port, the default port "4840" is used. If you use another port, you must append it with ":" and the port number.

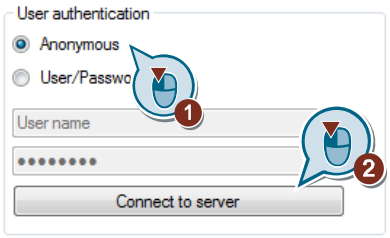
2. Select a detected endpoint to which you want to establish a connection.



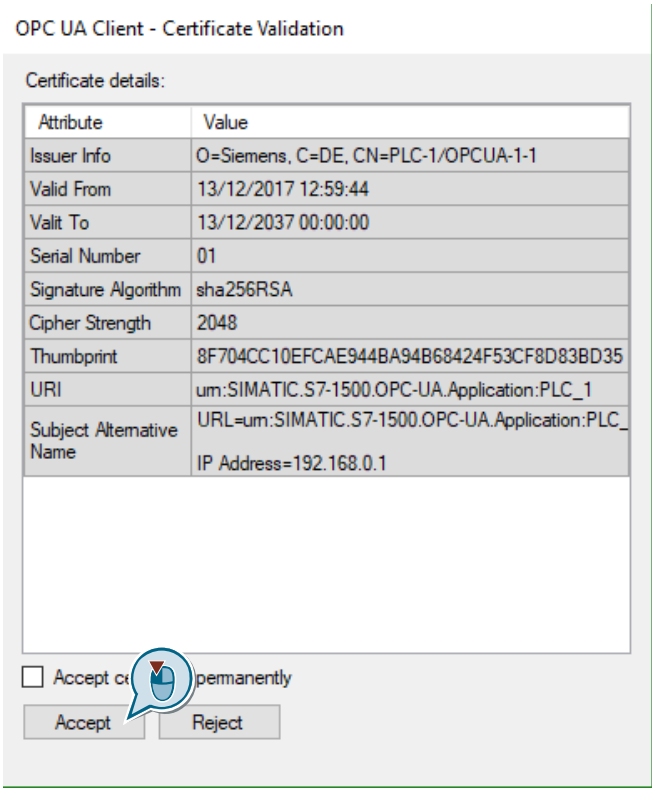
NOTE

If you want to connect to “Sign” or “SignAndEncrypt” endpoints, the client certificate of the example client has to be known to the server to be able to accept it (Chapter 2.1.4).

3. Select whether you want to log on anonymously or as certain user on the server. Then click “Connect to server” to establish the connection.



4. In the dialog that follows, accept the server certificate. To do this, click the "Accept" button. In order to permanently accept the server certificate, enable the "Accept certificate permanently" check box before clicking the "Accept" button.



NOTE

If you accept the certificate permanently, the certificate will be stored in the Windows Certificate Store. For all other connections to the server with this certificate, the "Certificate Validation" dialog will no longer appear.

In order to reject the certificate later on again, you have to remove it from the Windows Certificate Store.

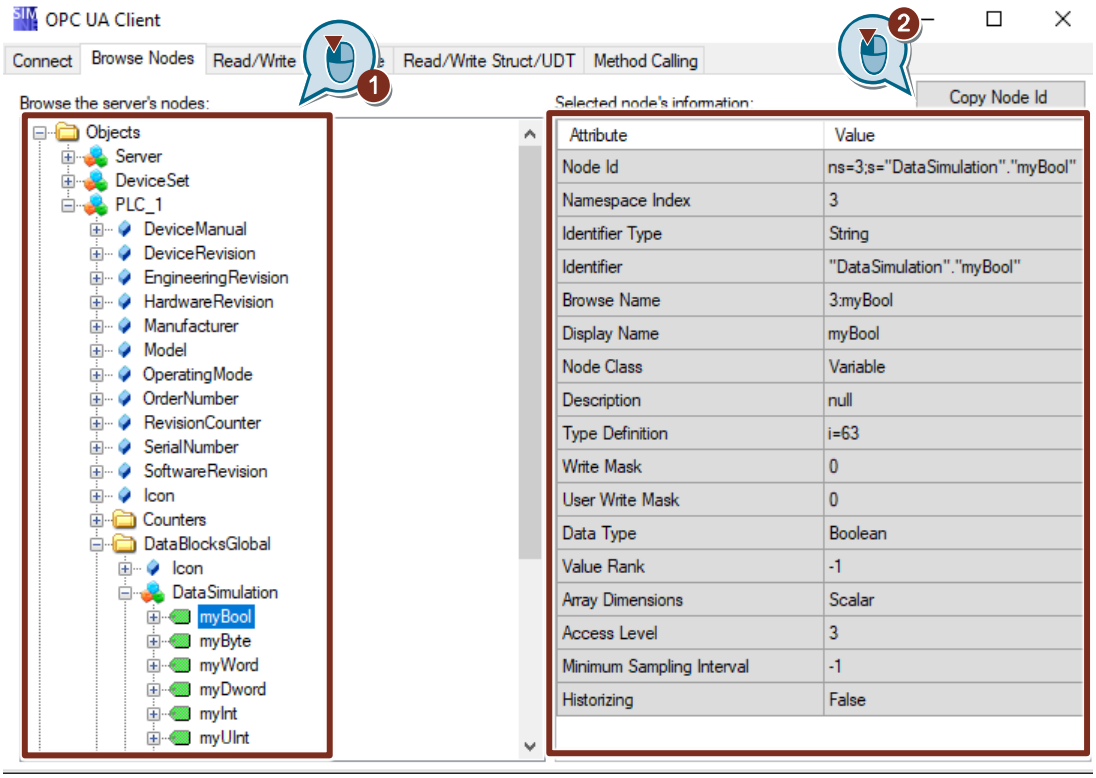
Within the store there is a permanently accepted certificate in "Console Root > Certificates - Current User > Trusted Root Certification Authorities > Certificates".

5. The text with the button "Connect to server" changes to "Disconnect from Server", once you are successfully connected with a server.
6. You can disconnect the session and the connection to the OPC UA server again via the “Disconnect from Server” button.

2.3.6 Browsing the address space of the OPC UA server

You can navigate via the “Browse Nodes” tab within the address space of the OPC UA server.

1. Connect with an OPC UA server.
2. Go to the "Browse Nodes” tab.
3. In the tree view you can browse through the individual nodes in the address space of the OPC UA server. When you are clicking on a node in the tree, you will receive specific information on the selected node on the right side in the data view. Click on the button “Copy Node Id” to copy the node ID of the selected node to the tree view.



NOTE

The information of the data view displayed, depends on the node class of the selected node (object, tag und data type).

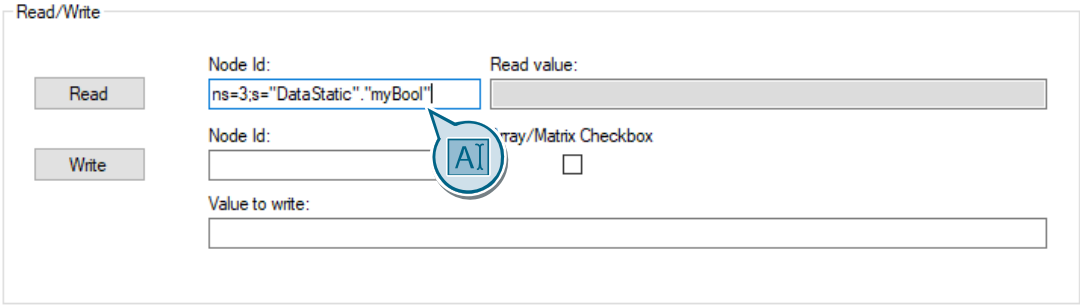
2.3.7 Reading/writing tags

Data access to tags of an OPC UA server is realized and shown to you in the “Read/Write” tab.

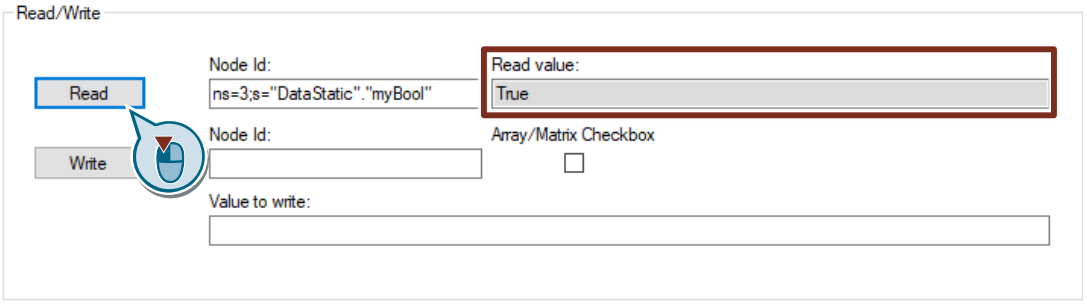
1. Connect with an OPC UA server.
2. Go to the “Browse Nodes” tab and navigate to a tag node that you want to read or write.
3. Click on the node of the tag, click on the button “Copy Node Id” to copy the value of the “Node Id” to the clipboard or copy the value of the “Node Id” field from the data view with the <CTRL+C> button combination.
4. Go to the “Read/Write” tab.

Reading scalar tag nodes

1. Add the previously copied node ID with the CTRL-V key combination into the upper field “Node Id:” of the “Read/Write” area.

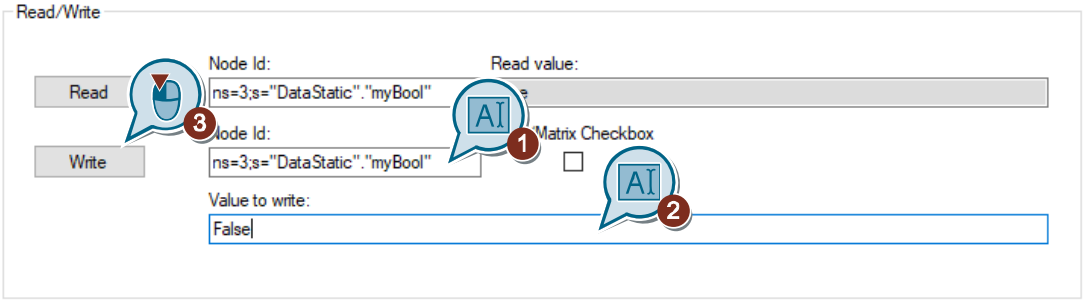


2. Click on the “Read” button. The read value is output in the “Read value:” text field.



Writing scalar tag nodes

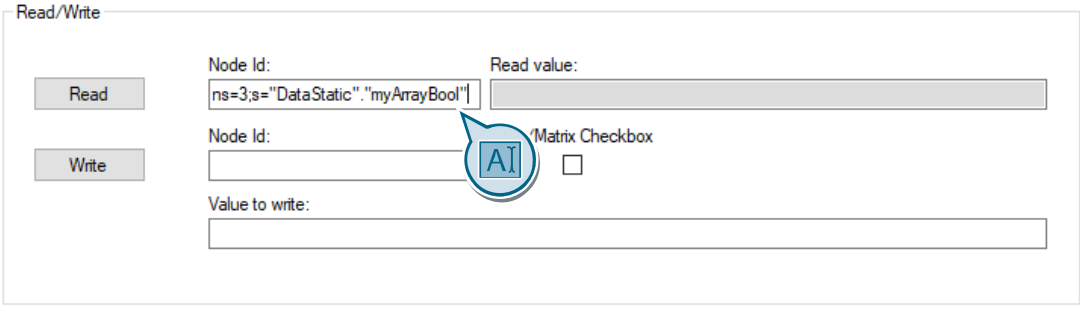
1. Add the previously copied node ID with the CTRL-V key combination into the bottom field “Node Id:” of the “Read/Write” area. Enter the value to be written into the “Values to write:” field (in this example the Boolean value “False”). Click on the “Write” button.



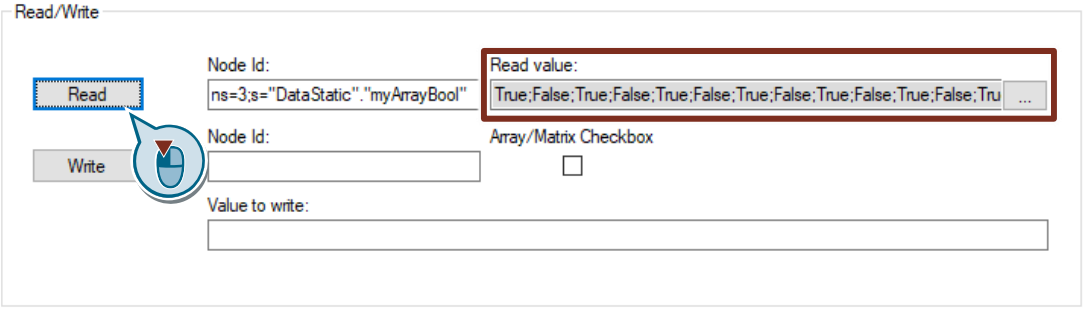
2. The write operation was successful when a message-box with the text “Values written successfully.” appears.
3. You can read the tag again to check it.

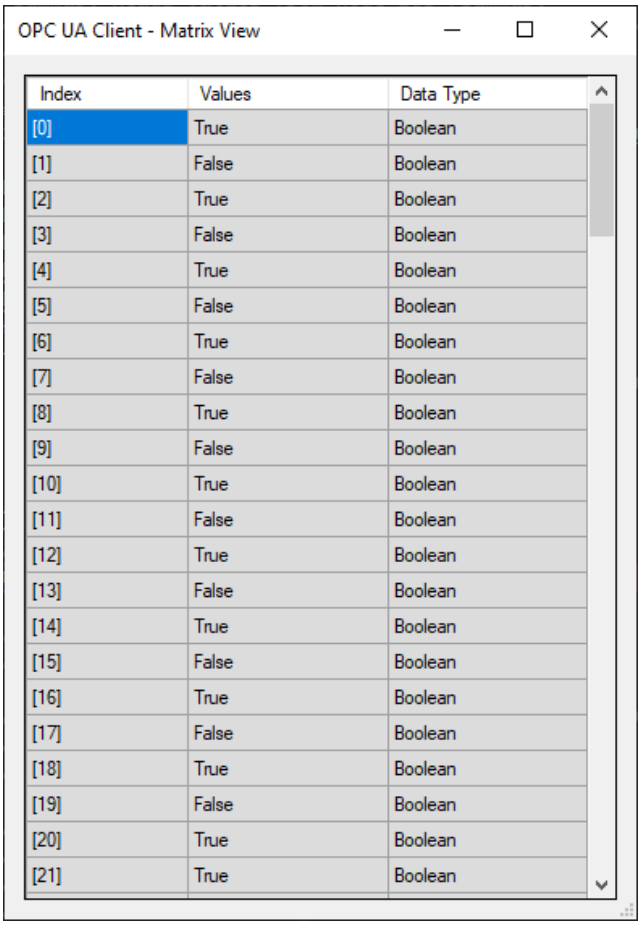
Reading array/matrix tag nodes

1. Add the previously copied node ID with the CTRL-V key combination into the upper field “Node Id:” of the “Read/Write” area.



2. Click on the “Read” button. The read value is output in the “Read value:” text field. If the read values are an array or a matrix, the single values are separated via a semicolon. Click on the button “…” to show the values in tabular form.

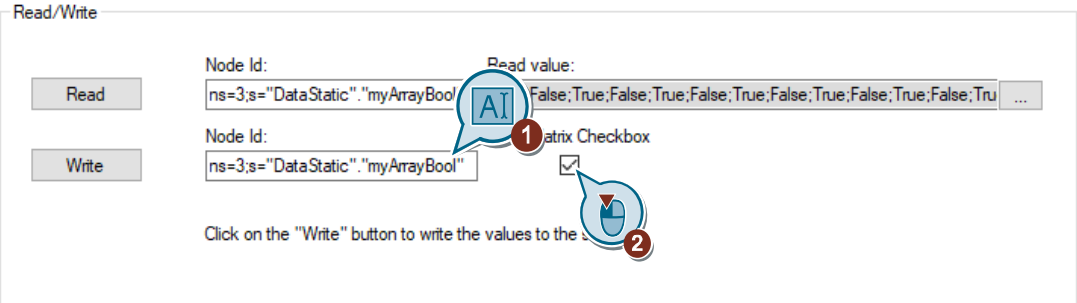




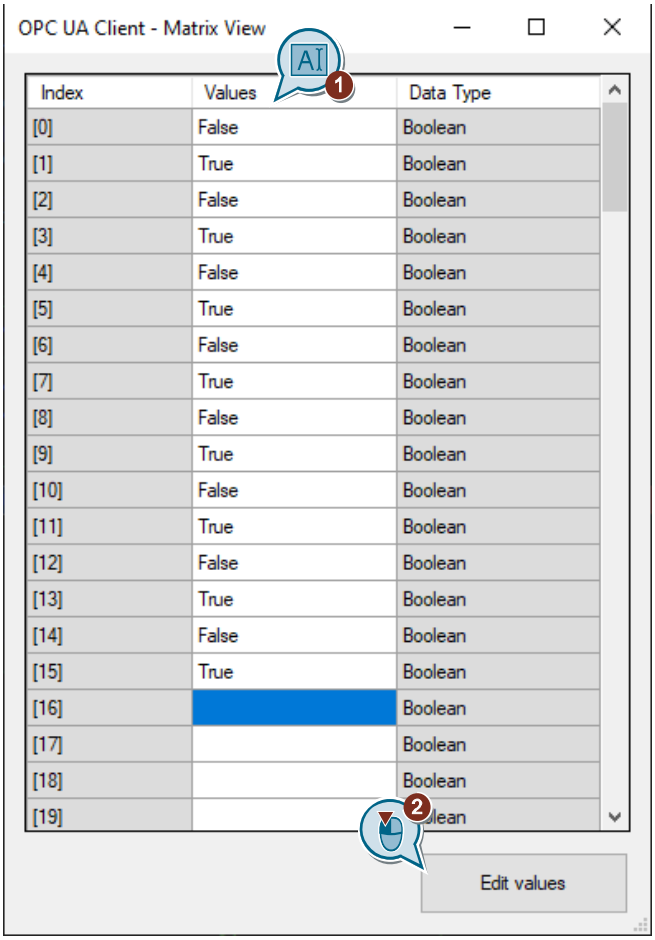
Index	Values	Data Type
[0]	True	Boolean
[1]	False	Boolean
[2]	True	Boolean
[3]	False	Boolean
[4]	True	Boolean
[5]	False	Boolean
[6]	True	Boolean
[7]	False	Boolean
[8]	True	Boolean
[9]	False	Boolean
[10]	True	Boolean
[11]	False	Boolean
[12]	True	Boolean
[13]	False	Boolean
[14]	True	Boolean
[15]	False	Boolean
[16]	True	Boolean
[17]	False	Boolean
[18]	True	Boolean
[19]	False	Boolean
[20]	True	Boolean
[21]	True	Boolean

Writing array/matrix tag nodes

1. Add the previously copied node ID with the CTRL-V key combination into the bottom field “Node Id:” of the “Read/Write” area. Click on the check box “Array/Matrix Checkbox” to open a window form to enter the values in tabular form.

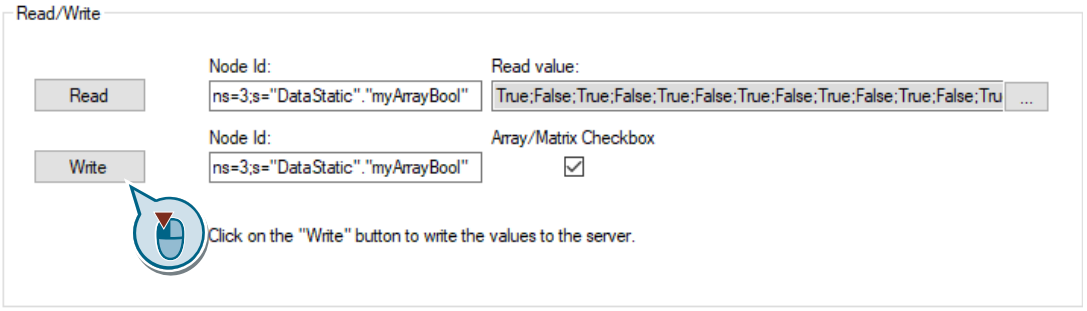


2. Enter the values to be written in their according fields and clock on the button “Edit Values” to close the window form.



NOTE All fields are always written. If a field does not contain an entry, it is written with the default value (usually "0" or "null").

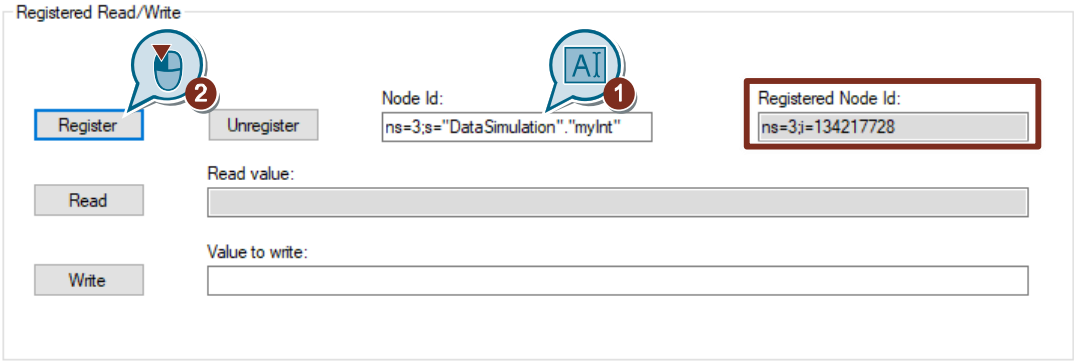
3. Click on the “Write” button.



4. The write operation was successful when a message-box with the text “Values written successfully.” appears.
5. You can read the tag again to check it.

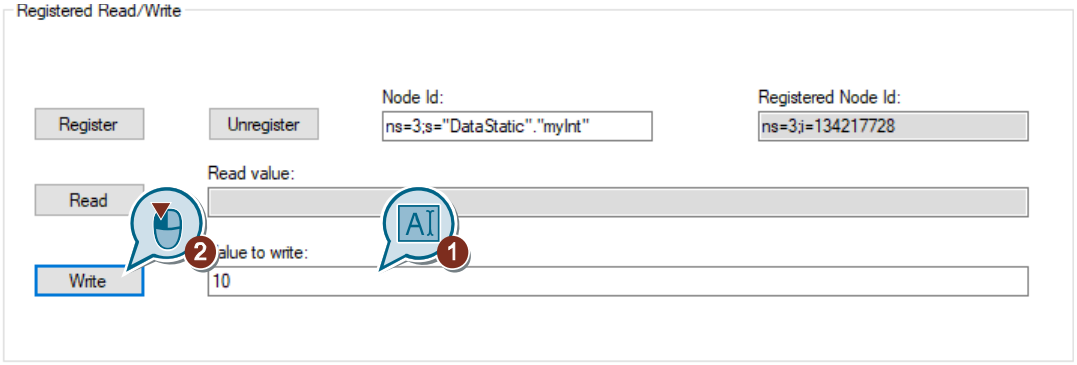
Registered read/write scalar node tags

1. Add the previously copied node ID with the CTRL-V key combination into the “Node Id:” field of the “Registered Read/Write” area and click on the “Register” button. The registered or optimized node ID is displayed in the “Registered Node Id:” field.



NOTE The registered node ID does not necessarily differ from the original.

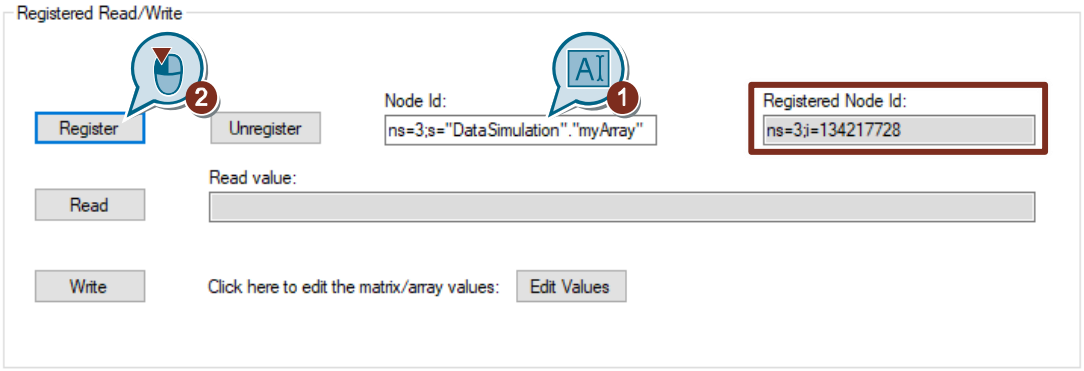
2. Click on the “Read” button, in order to read the registered node ID. The read value is shown in the “Read value:” field.
3. Enter a value in the “Values to write:” field. Click on the “Write” button, in order to write the previously entered value to a registered node.



4. The write operation was successful when a message-box with the text “Values written successfully.” appears.
5. You can read the tag again to check it.
6. Click on the “Unregister” button, in order to release a registered node ID.

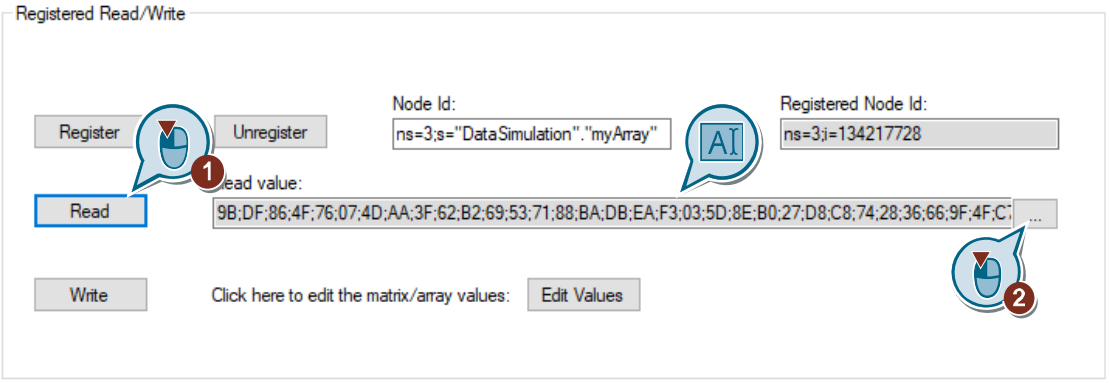
Registered read/write array/matrix node tags

1. Add the previously copied node ID with the CTRL-V key combination into the “Node Id:” field of the “Registered Read/Write” area and click on the “Register” button. The registered or optimized node ID is displayed in the “Registered Node Id:” field.

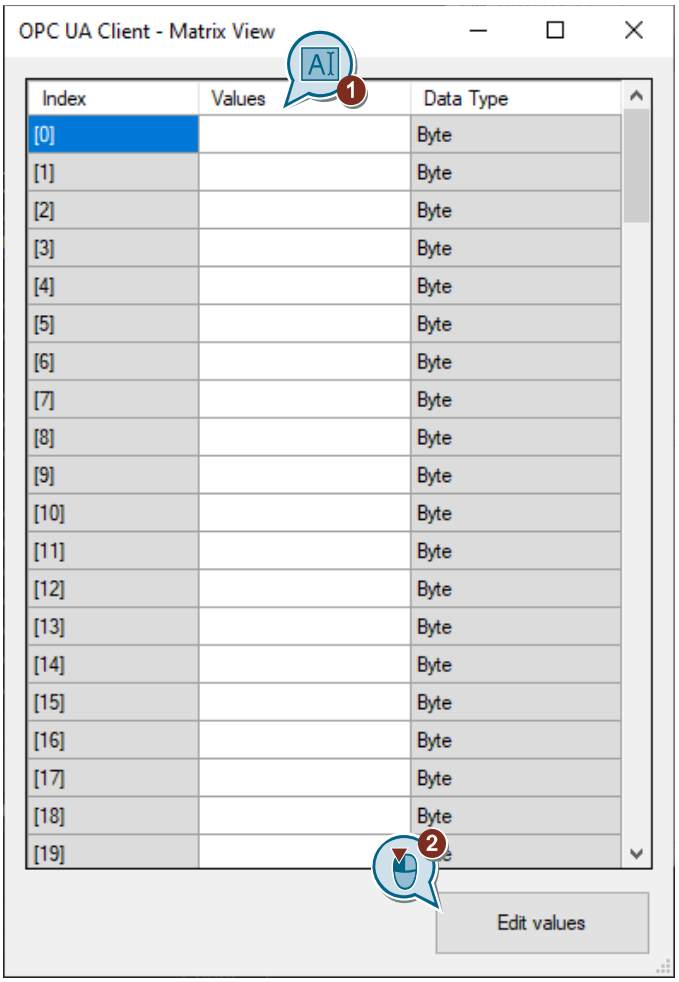


NOTE The registered node ID does not necessarily differ from the original.

2. Click on the “Read” button. The read value is output in the “Read value:” text field. If the read values are an array or a matrix, the single values are separated via a semicolon. Click on the button “…” to show the values in tabular form.



3. Click on the button “Edit Values” to open a window form for entering array/matrix values. Enter the values to be written in the corresponding fields and confirm your entry by clicking on the button “Edit Values”.



NOTE

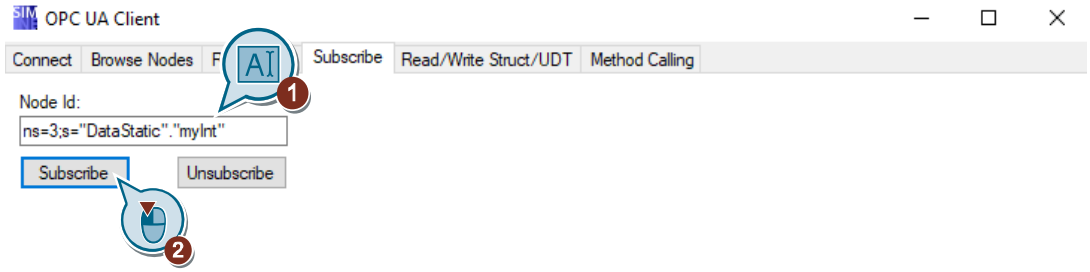
All fields are always written. If a field does not contain an entry, it is written with the default value (usually "0" or "null").

4. Click on the “Write” button to write the confirmed values to the node id.
5. The write operation was successful when a message-box with the text “Values written successfully.” appears.
6. You can read the tag again to check it.
7. Click on the “Unregister” button, in order to release a registered node ID.

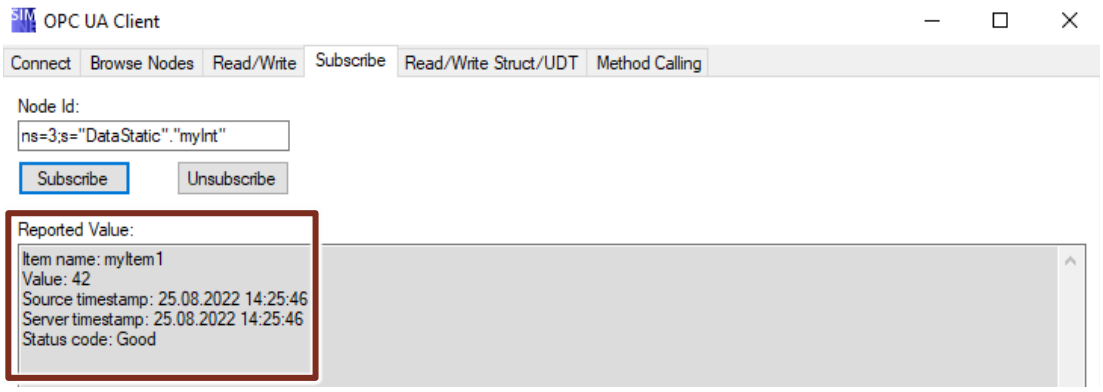
2.3.8 Subscriptions

You get value changes or updates of tags via a subscription, without ordering cyclic reading.

1. Connect with an OPC UA server.
2. Go to the “Browse Nodes” tab and navigate to a tag node that you want to subscribe to.
3. Click on the node of the tag, click on the button “Copy Node Id” to copy the value of the “Node Id” to the clipboard or copy the value of the “Node Id” field from the data view with the <CTRL+C> button combination.
4. Go to the “Subscribe” tab.
5. Add the previously copied node ID with the CTRL-V key combination in the “Node Id:” field. Click the "Subscribe" button.



6. In the “Reported Value:” text field, value, status, source and server time stamp of the MonitoredItems are displayed.



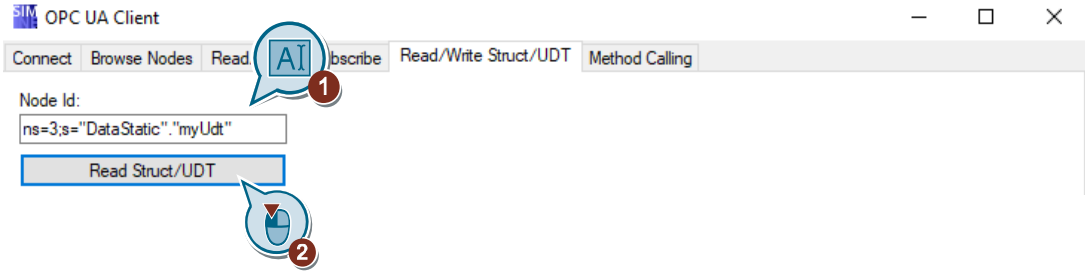
NOTE In this example, the publishing interval of the subscription is set to 1000ms and the sampling interval of the MonitoredItems is set to 1ms.

7. Click the “Unsubscribe" button to end the subscription.

2.3.9 Reading and writing structures/UDTs

Data access to tags that consist of a structure or an UDT will be shown to you in the “Read/Write Struct/UDT” tab.

1. Connect with an OPC UA server.
2. Go to the “Browse Nodes” tab and navigate to a tag node that you want to read or write (the node has to be entered a node ID in "Type Definition". This will show that the node consists of a user-defined type or a structure/a UDT).
3. Click on the node of the tag, click on the button “Copy Node Id” to copy the value of the “Node Id” to the clipboard or copy the value of the “Node Id” field from the data view with the <CTRL+C> button combination.
4. Go to the "Read/Write Struct/UDT" tab.
5. Add the previously copied node ID with the CTRL-V key combination in the “Node Id:” field. Click the "Read Struct/UDT" button.

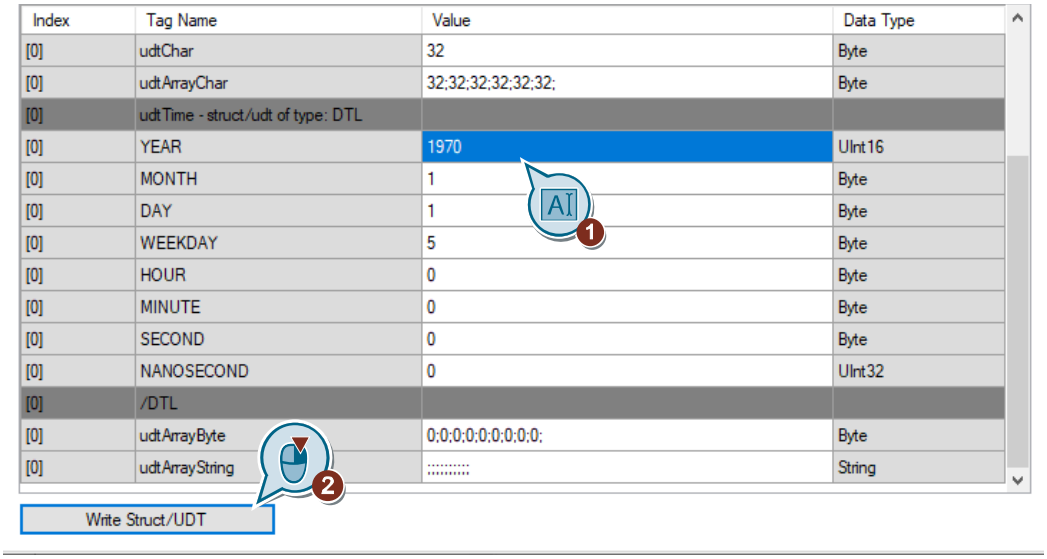


6. The tag names, values and data types of the read structure/UDT will be shown in the data view:

Index	Tag Name	Value	Data Type
[0]	udtInt	42	Int16
[0]	udtBool	True	Boolean
[0]	udtReal	38,5	Float
[0]	udtString	SiemensAG	String
[0]	udtWord	65440	UInt16
[0]	udtByte	16	Byte
[0]	udtChar	32	Byte
[0]	udtArrayChar	32;32;32;32;32;32;	Byte
[0]	udtTime - struct/udt of type: DTL		
[0]	YEAR	1970	UInt16
[0]	MONTH	1	Byte
[0]	DAY	1	Byte
[0]	WEEKDAY	5	Byte
[0]	HOUR	0	Byte

NOTE Dark grey colored fields mark the begin and the end of a struct/udt in the parent struct/udt.

7. Change one or several values in the "Value" column and click the "Write Struct/UDT" button.

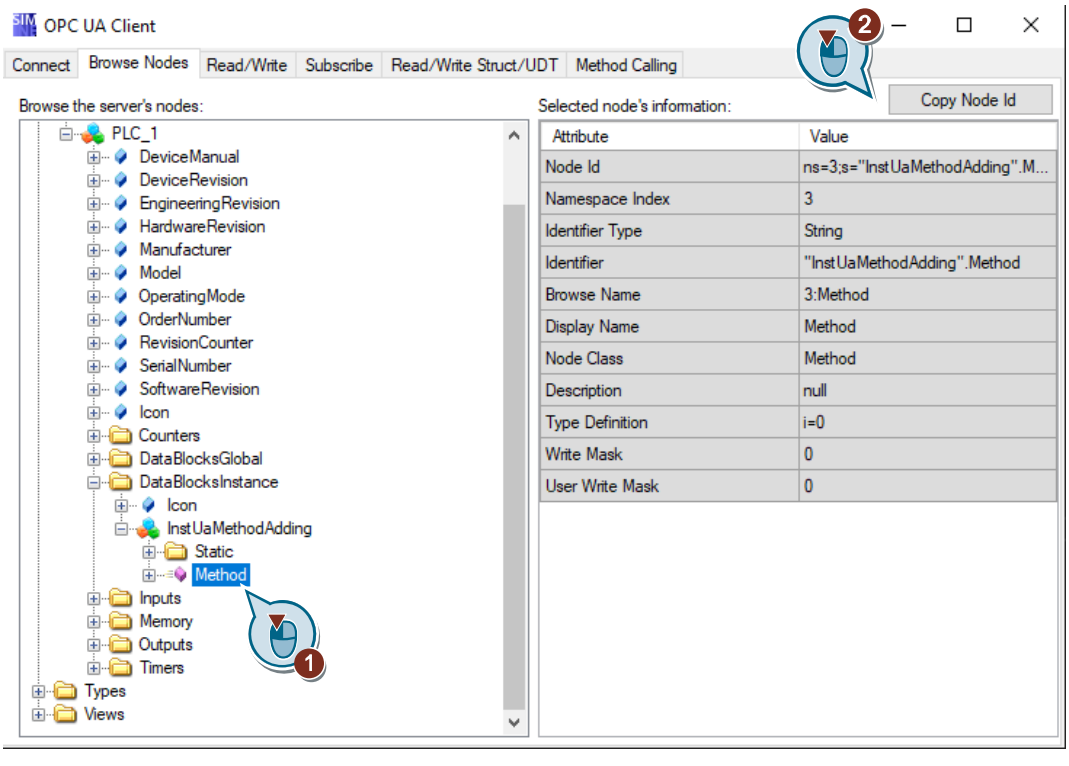


8. The write operation was successful when a message-box with the text “Values written successfully.” appears.
9. You can reread the structure/UDT again to check it.

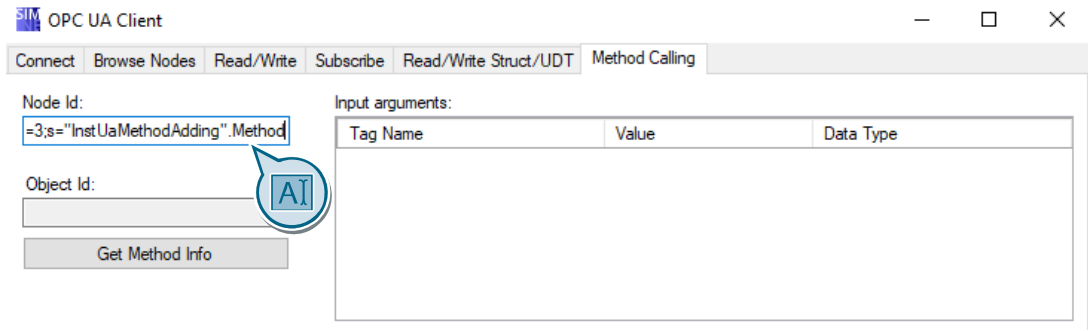
2.3.10 Calling methods

You can call OPC UA methods in the "Method Calling" tab.

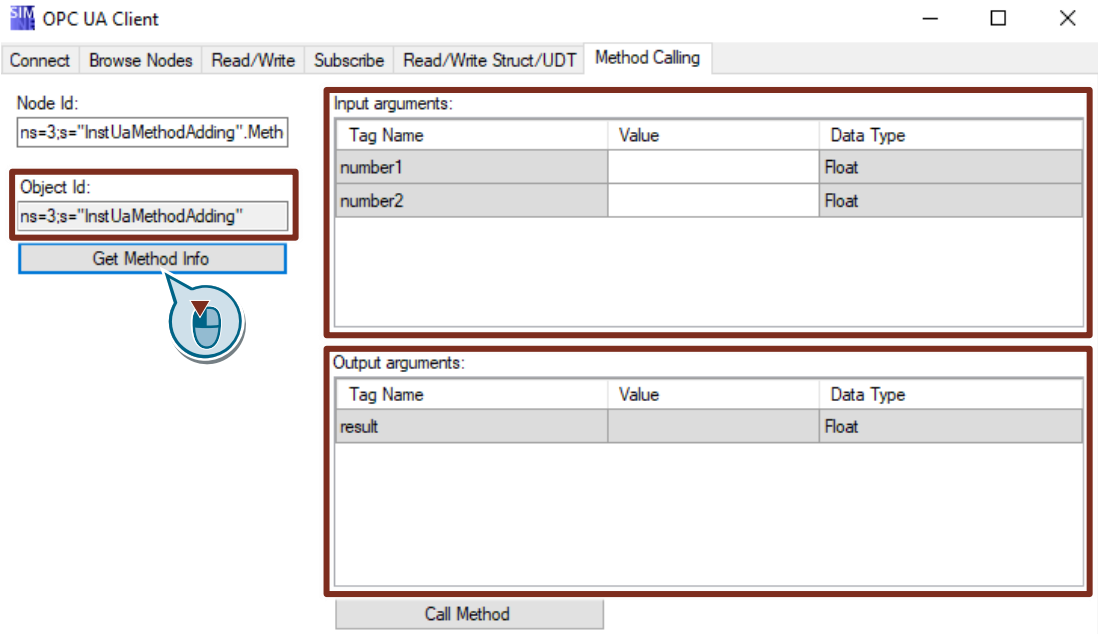
1. Connect to an OPC UA server
2. Go to the "Browse Nodes" tab and navigate to a method node you want to call. Click on the node of the tag, click on the button “Copy Node Id” to copy the value of the “Node Id” to the clipboard or copy the value of the “Node Id” field from the data view with the <CTRL+C> button combination.



3. Go to the "Method Calling" tab.
4. Add the previously copied node ID with the CTRL-V key combination in the “Node Id” field.

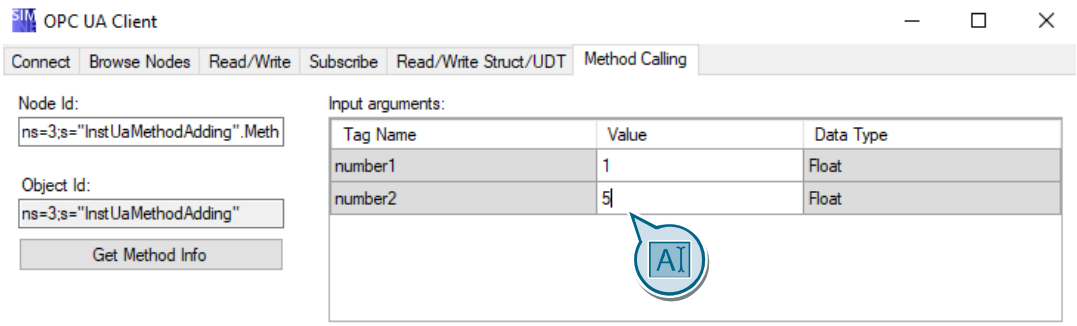


5. Click the "Get Method Info" button to receive the object id and the input and output arguments of the method.

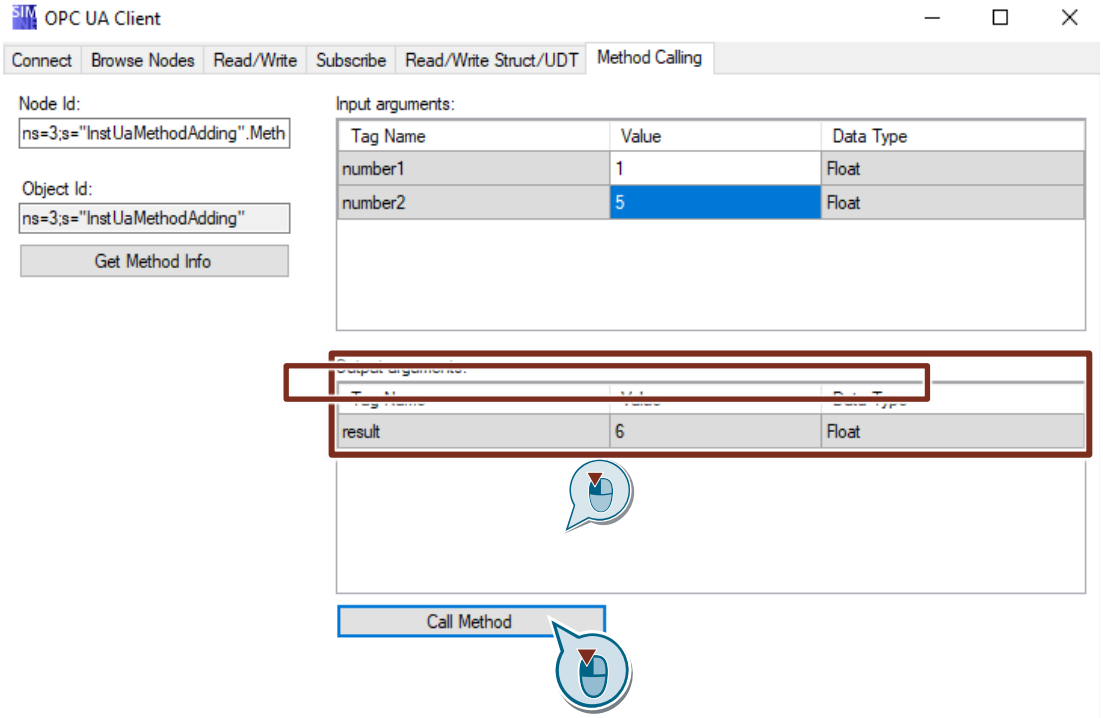


NOTE For floating point numbers, the English notation is used, which is why a "." must be used when entering them.

6. Enter the values you want to submit to the method into the "Value" column of the "Input arguments" list.



7. Click the "Call Method" button to call the method. The output argument values are output in the "Output arguments" list after the method was called successfully.



3 Valuable Information

3.1 Basics

3.1.1 General OPC UA information

Overview

In recent years, the OPC Foundation (an interest group of well-known manufacturers for the definition of standard interfaces) has defined a large number of software interfaces to standardize the information flow from the process level to the management level. According to the different requirements within an industrial application, different OPC specifications have been developed in the past: 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 acknowledgement, HDA describes functions for archived data and DX defines a lateral server to server communication.

Based on the experience with these classic OPC interfaces, the OPC Foundation defined a new platform, called OPC Unified Architecture (UA). The aim of this standard is the generic description and uniform access to all information which is to be exchanged between systems or applications. This includes the functionality of all previous OPC interfaces. Furthermore, this has generated the option of natively integrating the interface into the appropriate system, irrespective of which operating system the system is operated on and irrespective of the programming language in which the system was created.

Any more information can be found on the homepage or the OPC Foundation (\4\).

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 have been defined according to the rules of Microsoft COM (Component Object Model) and can therefore be easily integrated into Microsoft operating systems. COM or DCOM (Distributed COM) provides the functionality of inter process communication and organizes the information exchange between applications, even across network boundaries (DCOM). Using mechanisms of the Microsoft operating system, an OPC client (COM client) can use it to exchange information with an OPC server (COM server).

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

The use of COM or DCOM causes OPC servers and clients to run only on a Windows PC or in the local network and that the communication to the respective automation system has to be realized mainly via proprietary protocols. Additional tunneling tools often have to be used for the network communication between client and server in order to get through firewalls or to avoid the complicated DCOM configuration. The interface can furthermore only be accessed natively with C++ applications; .NET or JAVA applications can only gain access via a wrapper layer. In real-life situations, these restrictions lead to additional communication and software layers which increase the configuration workload and the complexity.

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

To solve the mentioned restrictions in real-life situations and to fulfill the additional requirements, the OPC Foundation has defined a new platform in the last 7 years, called OPC Unified Architecture, which offers a uniform basis for the exchange of information between components and systems. OPC UA is available as an IEC 62541 standard and therefore 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 a generic interface.

- Use of open and platform-independent protocols for inter-process or network communication.
- Internet access and communication by means of firewalls.
- Integrated access control and security mechanisms on protocol and application level.
- Extensive representation options for object-oriented models; objects can have tags and methods and can fire events.
- Expandable type system for objects and complex data types.
- Transport mechanisms and modeling rules form the basis for other standards.
- Scalability of small embedded systems up to business applications and from simple DA address spaces up to complex, object-oriented models.

3.1.2 OPC UA address space

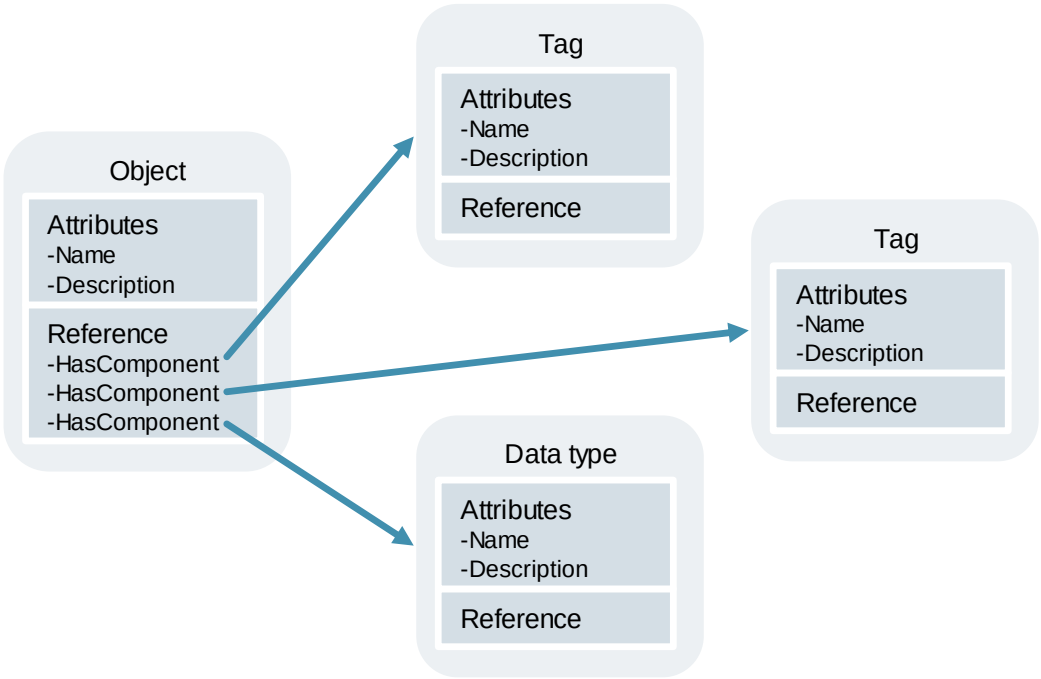
The following descriptions explain the address space of an OPC UA server.

Nodes in the address space

A node in the OPC UA address space is of a certain type, such as, for example, object, tag or method and is described by a list of attributes. All nodes have joint attributes such as name or description and specific attributes such as, for example, the value of a tag. The list of attributes cannot be extended. Additional information on the node can be added as property. Properties are a special type of tag. The nodes are interconnected with references. The references are typified. There are two main groups: Hierarchical references, such as, for example, HasComponent for the components of an object or non-hierarchical references such as, for example, HasTypeDefinition for a connection of an object instance to an object type.

The following figure shows an example for nodes and the connecting references:

Figure 3-1



Available types of nodes in the address space

The following table shows the node types defined in the standard:

Table 3-1

Node type	Description
Object	An object is used as typified container or folder for tags, methods and events.
Tag	Tags represent the data of objects or the properties of a node as attributes.
Method	Methods are components of objects and can have a list of input or output parameters. The parameters are described via defined attributes.
View	Views represent a part of the address space. The node is used as access point and as filter when browsing.
Object type	Object types supply information on the structure or the components of an object.
Tag type	Tag types typically describe which attributes or data types can be found in an instance of a tag.
Reference type	Reference types define the possible types of references between nodes.
Data type	Data types describe the content of the value in a tag.

Name spaces and node IDs

Each node in the OPC UA address space is uniquely identified by a node ID. This node ID is made up of a namespace to distinguish codes from different subsystems and a code which can either be a numerical value, a string or a GUID.

Strings are typically used for the ID. This is analog to OPC Data Access, where the item ID as identifier is also a string. Numerical values are used for statistical namespaces such as, for example, type system. OPC UA defines a namespace with associated namespace index for the nodes defined by the OPC Foundation. The OPC UA servers additionally define one or several namespaces with index. The namespaces defined by the servers are variable and can change. This is why it is recommended to request the current namespace for the client when establishing the session.

The figure below explains the structure of a node ID:

Figure 3-2

Table 3-2

No.	Description
1.	Namespace index
2.	Node ID type (s=String; i=Numeric: g=GUID)
3.	ID

Attributes of the nodes

The table below explains the most important node attributes:

Table 3-3

Attribute	Node type	Description
Node ID	All	The unique node ID with namespace index
Namespace index	All	The namespace index that is assigned to the node.
Identifier Type	All	The node ID type
Identifier	All	The unique node ID within the namespace index
Browse Name	All	The browse name
Display Name	All	The display name
Node Class	All	The node class (object, tag, data type)
Description	All	Short description of the node
Type Definition	All	Reference for data type description of the tag
Write Mask	All	Write rights to node attributes (0=no, 1=yes) without consideration of user groups
User Write Mask	All	Write rights to node attributes (0=no, 1=yes) without consideration of the current user
Data Type	Tag	Data type of the tag
Value Rank	Tag	Value type of the tag (none, scalar, vector, array)
Array Dimensions	Tag	Number of array dimensions
Access Level	Tag	Access authorization (read, write, read/write) to the node
Minimum Sampling Interval	Tag	The smallest possible sampling interval of the tag on the server side
Historizing	Tag	Course of time of the tag available on server (yes, no)

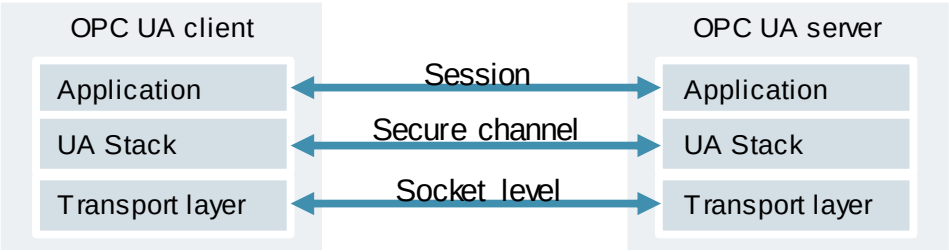
3.1.3 OPC UA Security

The following explanations outline the security concept of OPC UA.

Security layers

The following figure gives an overview of the security layers of OPC UA:

Figure 3-3



The user authentication is carried out via the **Session**. This is done, for example, through a user name and a password or via certificates.

Via a **Secure Channel** the applications are mutually authenticated and a message-based security of the communication is performed. Each message is signed and encrypted to ensure the integrity and secrecy of the messages. Basis of these mechanisms are certificates (X509) which uniquely identify the applications based on a Public Key Infrastructure (PKI) system.

On the **socket level**, a connection-oriented security of the socket connection via Secure Socket Layer (SSL) or via Virtual Private Network (VPN) can be used in addition or as an alternative to the secure channel.

Configuration options for the security

The following table describes the different configuration options for the security mechanisms:

Table 3-4

Option	Description
Security Policy	**None** – In the secure channel no security is used. **Basic128Rsa15** – Set of encryption algorithms. **Basic256** – Set of expandable encryption algorithms.
Message Security Mode	**None** – The messages are not secured. **Sign** – The messages are signed. **Sign&Encrypt** – The messages are signed and encrypted.
User Authentication	**Anonymous** – User authentication is not necessary. **User Password** – The user authentication is performed using user names and password. **Certificate** – The user authentication is performed using a certificate.

Certificate exchange between client and server

When all applications involved, implement the guidelines of the OPC UA regarding the security configuration, only one manual step (4) is necessary at the server for the exchange of certificates, since the certificates are automatically exchanged between the applications and the certificates only have to be accepted by an administrator.

The following figure illustrates the certificate exchange between client and server:

Figure 3-4

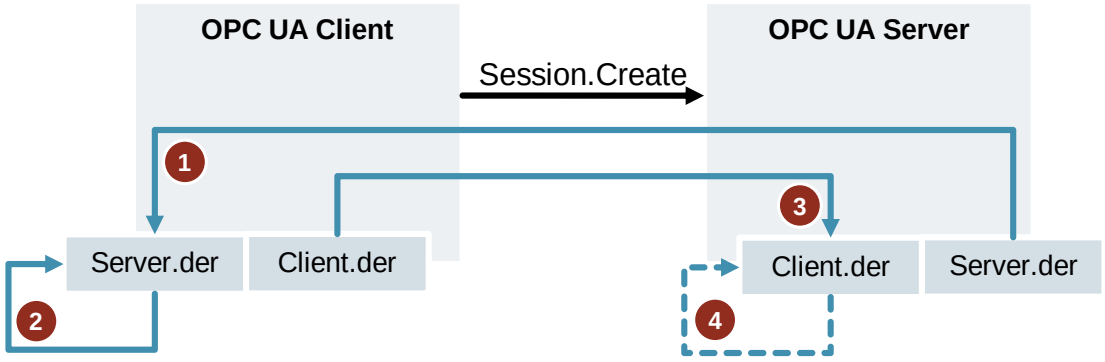


Table 3-5

No.	Description
1.	When establishing a connection to the server (Session.Create), the client receives the server certificate via the server endpoint.
2.	The client program can then decide how it deals with the certificate: Reject or accept.
3.	In the same process the client sends its certificate to the server. The server rejects the certificate at first and then stores it in a reject folder.
4.	As a result, the client certificate has to be accepted manually by an administrator on the server. In most cases, this is done by an administrator copying the client certificate from a reject folder into a trusted folder.

NOTE For the OPC UA Server of the S7-1500 the client certificate has to be loaded via the TIA Portal onto the controller, in order to accept it.

3.1.4 OPC UA server of the S7-1500

This chapter gives you an overview of some key data of the OPC UA server of the S7-1500. Additionally, notes and tips in handling the server are also given.

NOTE Further information on the OPC UA server of the S7-1500 can be found in the “Function Manual: S/-1500, ET 200MP, ET 200SP, ET 200AL, ET 200pro Communication” (\3\).

Supported OPC UA services of the S7-1500 data access

The OPC UA server of the S7-1500 currently supports the following services for data access:

- Read
- Write
- Registered read/write
- Subscriptions

Performance when accessing many tags of the server

When you want to read or write many tags from a S7-1500, you can increase the performance considerably by structuring the tags on the S7-1500. To do this, use arrays and structures, in order to declare read/write tags.

Viewed individually, arrays are the most performant. They are faster by factor 2 to 3 than structures and they are faster by factor 10 to 100 than individual accesses (by a number of approximately 1000 tags).

Use the “Registered read/write” for recurring accesses, in order to increase the performance.

License concept

Table 3-6

CPU type	ET 200SP CPU up to S7-1513(F)	1515 / 1516(F)	1517 / 1518(F)
Required license	Small	Medium	Large

3.2 TIA Portal project details

The explanation below describes the TIA Portal project included in this application example.

3.2.1 S7-1500 and OPC UA configuration

The following configuration steps from chapter 2.1 “Configuring the OPC UA Servers of the S7-1500” have already been carried out for you in the TIA Portal project of this entry.

- The OPC UA server is enabled.
- The global security settings are enabled (User: User / password Siemens.1).
- The server endpoints are set up.
- The tags for the OPC UA communication are enabled.
- An anonymous access is enabled.

Only the settings in chapter 2.1.4 “Security via certificate management” are optional and can still be carried out by you, in order to additionally increase the security in the project.

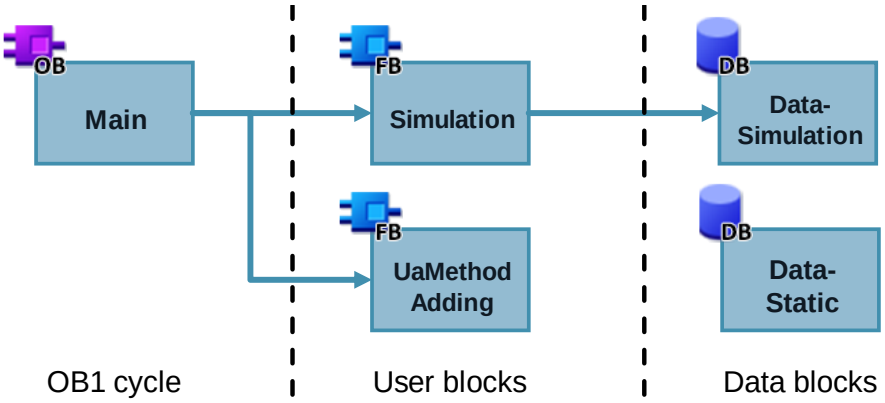
3.2.2 S7 program

The S7 program of the TIA project consists of the OB1, a user block and two data blocks.

Call hierarchy

The following figure shows the call hierarchy of the S7 user program.

Figure 3-5



Explanation of the blocks

In the cyclic user program only the “Simulation” and "UaMethodAdding" function blocks are called. "Simulation" creates pseudo-randomly generated values and fills the tags in the “DataSimulation” data block with them. "UaMethodAdding" implements an OPC UA method, which adds two input values and provides the result as output. The “DataStatic” data block includes predefined tags with statistic values.

3.3 License model for OPC UA .NET stack/SDK

The OPC UA .NET stack used in this example was created by the OPC Foundation and is continued as Open Source project. The GitHub community is responsible for the further development and troubleshooting of the stack.

Depending on its affiliation to the OPC Foundation, the stack is subject to two different license models:

1. For members of the OPC foundation the "RCL" model applies.
2. For non-members of the OPC foundation the "GPL 2.0" model is valid.

NOTE Obtain some information on the respective license models, depending on their affiliation to the OPC foundation, before you start with the development of your own application.

4 Appendix

4.1 Service and support

Industry Online Support

Do you have any questions or need assistance?

Siemens Industry Online Support offers round the clock access to our entire service and support know-how and portfolio.

The Industry Online Support is the central address for information about our products, solutions and services.

Product information, manuals, downloads, FAQs, application examples and videos – all information is accessible with just a few mouse clicks:

support.industry.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:

siemens.com/SupportRequest

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

Service offer

Our range of services includes the following:

- Plant data services
- Spare parts services
- Repair services
- On-site and maintenance services
- Retrofitting and modernization services
- Service programs and contracts

You can find detailed information on our range of services in the service catalog web page:

support.industry.siemens.com/cs/sc

Industry Online Support app

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

support.industry.siemens.com/cs/ww/en/sc/2067

4.2 Links and literature

Table 4-1

No.	Topic
\1\	This entry in the Siemens Industry Online Support https://support.industry.siemens.com/cs/ww/en/view/109737901
\2\	Download of the OPC UA .NET stack of the OPC Foundation https://github.com/OPCFoundation/UA-.NETStandard/releases
\3\	Function Manual: SIMATIC S7-1500, ET 200MP, ET 200SP, ET 200AL, ET 200pro Communication https://support.industry.siemens.com/cs/ww/en/view/59192925
\4\	Homepage of the OPC Foundation https://opcfoundation.org/

4.3 Change documentation

Table 4-2

Version	Date	Modification
V1.0	10/2016	First version
V1.1	04/2017	• Expansions for the user authentication on the server • Expansions for reading/writing of structures/UDTs Various troubleshooting
V1.2	02/2018	• Expansion for calling methods Various troubleshooting
V1.3	06/2018	• Expansion for Namespace operations • Expansion of decodable data types (UDT/Struct) • Various troubleshooting Update to OPC UA 1.4
V1.5	09/2022	• Update to OPC UA Stack 1.04.369 • Adaptation of dependencies • Adaptation of certificate creation • New development of methods for reading and writing structures/UDTs • Various restructuring and simplification of methods Various troubleshooting