Siemens
Industry
Online
Support

APPLICATION EXAMPLE

OPC UA methods for the SIMATIC S7-1500 OPC UA server

S7-1500 / OPC UA Methods / Programming support

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. Introduction

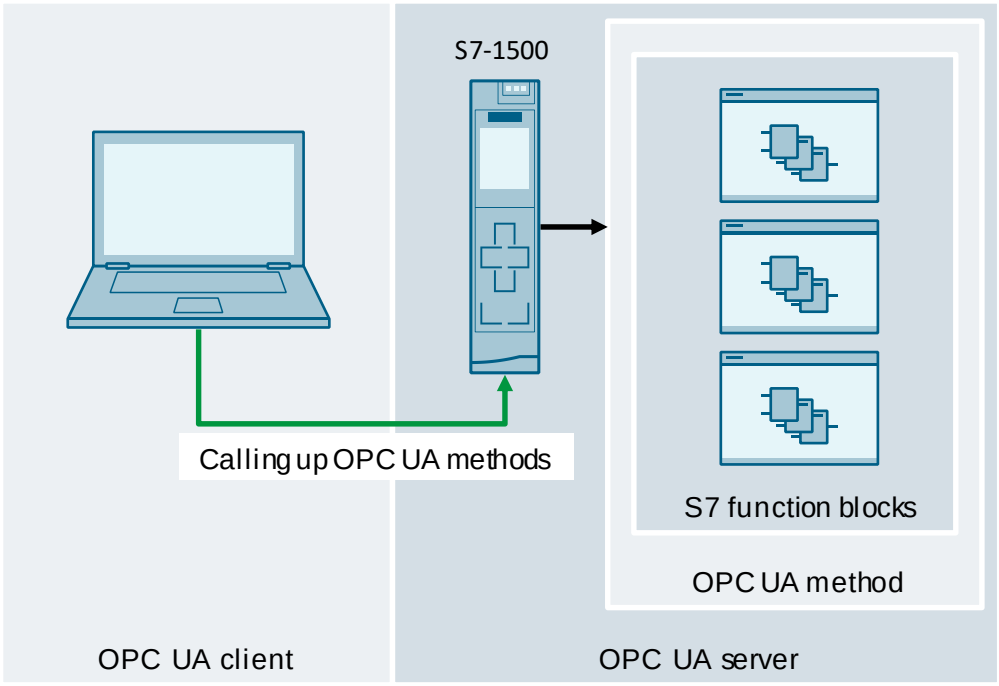
1.1. Overview

For standardized, communication of an S7 control system that applies to all platforms and manufacturers, Siemens offers you the communication protocol OPC UA.

As of firmware version V2.5 and TIA Portal V15, the SIMATIC S7-1500 control system can also offer OPC UA methods for the clients via its integrated OPC UA server. With this addition to the functions, you not only have the option of reading or writing OPC UA variables of the control system, you can now also start complex functional sequences via OPC UA. This enables almost complete M2M communication via OPC UA, for the networking of plants or for the control of plants from an ERP / MES level.

In this application example, we explain how to program an S7 function block so that it can be called up as an OPC UA method on the server of an S7-1500.

Figure 1-1 Pictorial schematic



1.2. Mode of operation

From a PLC programmer's point of view, OPC UA methods on a SIMATIC S7 controller are just conventional S7 function blocks. In addition to the normal program code, you must call up two system function blocks in the S7 function block in order to implement the OPC UA functionality. This does not restrict the program logic within the S7 function module compared to normal function blocks.

If you have created and programmed an S7 function block correctly, the method appears in the OPC UA address space of the OPC UA server of the S7-1500 and can be called up by an OPC UA client.

The example project supplied with this application example contains four fully programmed OPC UA methods for you:

- "OpcMethodAllowedDataTypes"
- "OpcMethodBubbleSort"
- "OpcMethodSetPlcTime"
- "OpcMethodStateMachine"

You can extend the methods or use them as a template for your own implementation.

1.3. Components used

This application example was created with the following hardware and software components:

Table 1-1

Component	Number	Article number	Note
SIMATIC S7-1500 CPU 1513-1 PN	1	6ES7 513-1AL01-0AB0	Firmware V2.5 or later
STEP 7 Professional	1	6ES7822-1AA05-0YA5	TIA Portal V15.1 or later

This application example consists of the following components:

Table 1-2

Component	File name	Note
Dokumentation	109756885_OpcUa_ServerMethods_DOC_V1_2_en.pdf	This document.
Beispielprojekt	109756885_OpcUa_ServerMethods_V17_PROJ_V1_2.zip	TIA Portal project for TIA V18.
OPC UA Status-Codes	109756885_OpcUa_ServerMethods_XML_V1_0.zip	Status codes for import into TIA Portal.

2. Engineering

2.1. Creation of OPC UA methods

In this chapter, we explain the procedure and basis for creating an OPC UA method. The system function blocks "OPC_UA_ServerMethodPre" and "OPC_UA_ServerMethodPost", which implement the OPC UA method, are the central component of an OPC UA method.

2.1.1. Explanation of the system function blocks

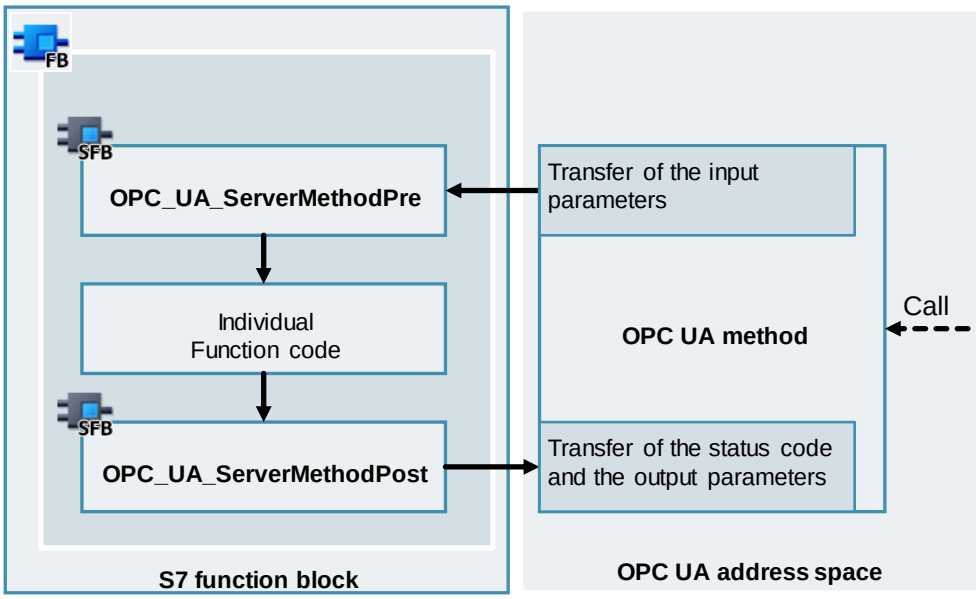
The creation of OPC UA methods requires calling the system function blocks "OPC_UA_ServerMethodPre" and "OPC_UA_ServerMethodPost". Call these system functions in an S7 function block to implement the functionality of the S7 function block as an OPC UA method. You will find the system function blocks in the instruction list of the TIA Portal under "Communication> OPC UA> OPC UA Server" ("Communication> OPC UA> OPC UA server").

Functional sequence

When calling an OPC UA method, the input parameters are transferred to the system function "OPC_UA_ServerMethodPre". The system function then provides these in the user program. Additionally, an interface parameter on the function block indicates that the method was called by an OPC UA client. Run your custom function code after the parameter is set.

After processing the function code, you must transfer the output parameters and the status code to the "OPC_UA_ServerMethodPost" system function and also set an interface parameter to indicate to the function that the code has been executed. This system function then transmits the output parameters and the status code to the OPC UA client.

Figure 2-1 Sequence of functions



Interface description of the system function blocks

The following figure and table explain the interface of the “OPC_UA_ServerMethodPre” system function:

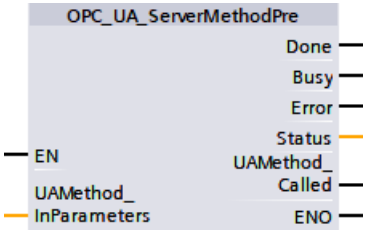


Table 2-1

Name	Data type	Description
UAMethod_ InParameters	Version	Input parameters of the method; Transferred by the client.
Done	Bool	True if the system function was called up successfully.
Busy	Bool	True if the system function is in progress.
Error	Bool	True if an error occurred while calling up the system function.
Status	DWord	Cause of the error, see TIA Help ("F1").
UAMethod_Called	Bool	True, if the method was called up by a client.

The following figure and table explain the interface of the "OPC_UA_ServerMethodPost" system function:

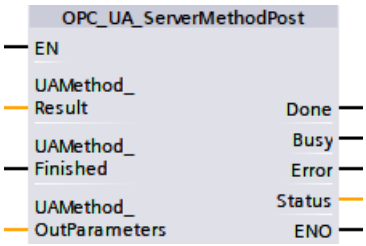


Table 2-2

Name	Data type	Description
UAMethod_Result	DWord	OPC UA status code; Will be transfered to the client.
UAMethod_Finished	Bool	Set to True when function code is complete; The method is hereby terminated.
UAMethod_Out parameters	Version	Output parameter of the method; transfered to the client.
Done	Bool	True if the system function was called up successfully.
Busy	Bool	True if the system function is in progress.
Error	Bool	True if an error occurred while calling up the system function.
Status	Status	Cause of the error, see TIA Help ("F1").

2.1.2. Preparation

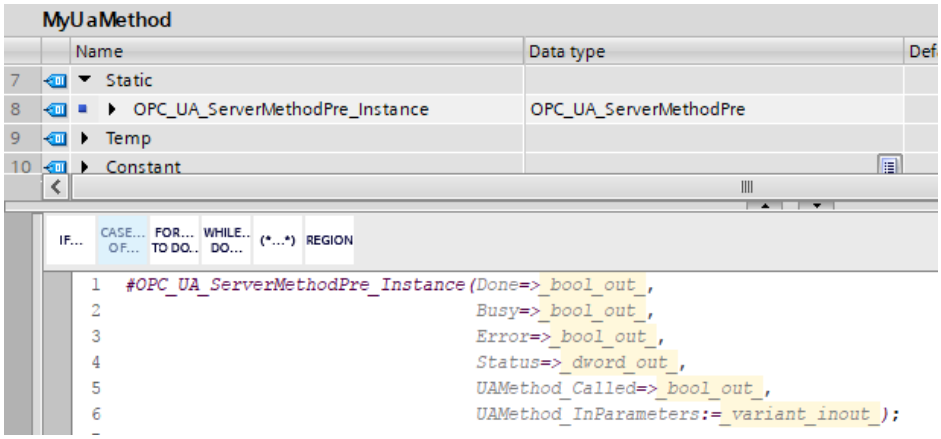
You must make the following preparations to create a functional OPC UA method for a SIMATIC S7-1500:

- Create a STEP 7 project.
- Configure a SIMATIC S7-1500 with firmware 2.5 or higher.
- Activate the OPC UA server of the S7-1500 so that the methods can be accessed via OPC UA. For instructions, see the appendix at \4\.
- Create a block of the "function block" type. The programming language used in the module is irrelevant. This example uses the scripting language "SCL".
- (Optional) Import the XML file "OpcUaMethodStatusCodes.xml" provided with this entry to use standardized OPC UA status codes. Instructions on this can be found in the chapter 3.2 OPC UA Status Codes.

2.1.3. Creation of the basic framework

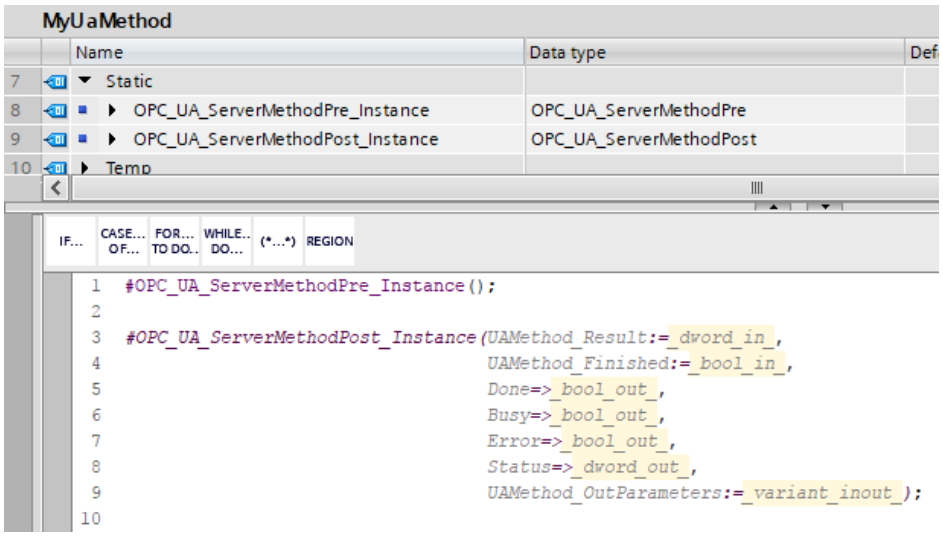
To create OPC UA methods, always create a basic framework in the module that is mandatory for the function of the method.
Perform steps 1 through 4 of the following procedure for each OPC UA method. Follow steps 5 through 8 only if your method requires input or output parameters. Proceed as follows:

1. Open the function block created in advance.
2. Call the "OPC_UA_ServerMethodPre" function in the block. Make sure that the function is created as a "multi-instance".



Note The name of the multi-instance must be "OPC_UA_ServerMethodPre_Instance" for the method to be created in the address space.

3. In the block, call the "OPC_UA_ServerMethodPost" function under the pre-function. Make sure that the function is created as a "multi-instance".



Note

The name of the multi-instance must be "OPC_UA_ServerMethodPost_Instance" for the method to be created in the address space.

4. Assign suitable variables to the block interfaces of the pre- and post-function. Information about the block interfaces can be found in chapter 2.1.1 Explanation of the system function blocks.

```
#OPC_UA_ServerMethodPre_Instance(Done=>#statPreDone,
                                 Busy=>#statPreBusy,
                                 Error=>#statPreError,
                                 Status=>#statPreStatus,
                                 UAMethod_Called=>#statPreCalled);

#OPC_UA_ServerMethodPost_Instance(UAMethod_Result:=#statPostResult,
                                  UAMethod_Finished:=#statPostFinished,
                                  Done=>#statPostDone,
                                  Busy=>#statPostBusy,
                                  Error=>#statPostError,
                                  Status=>#statPostStatus);
```

5. To define input parameters for the OPC UA method, create a variable with the name "UAMethod_InParameters" of the data type "Struct" or UDT in the static area of the block. Within the structure you define the individual transfer parameters (example: "myInt1" and "myInt2" of the data type "Int").

UAMethod_InParameters	Struct
myInt1	Int
myInt2	Int

Note

The name of the variable must be "UAMethod_InParameters" to display the input parameters of the method.

6. Assign the variable "UAMethod_InParameters" to the block interface "UAMethod_InParameters" of the pre-function.

```
//Call pre
#OPC_UA_ServerMethodPre_Instance(Done=>#statPreDone,
                                 Busy=>#statPreBusy,
                                 Error=>#statPreError,
                                 Status=>#statPreStatus,
                                 UAMethod_Called=>#statPreCalled
                                 UAMethod_InParameters:=#UAMethod_InParameters);
```

7. To define output parameters for the OPC UA method, create a variable with the name "UAMethod_OutParameters" of the data type "Struct" or UDT in the static area of the block. Within the structure, you define the individual transfer parameters (example: "myIntResult" of the data type "DInt").

24	UAMethod_OutParameters	Struct
25	myIntResult	DInt

Note The name of the variable must be "UAMethod_OutParameters" to display the output parameters of the method.

8. Assign the variable "UAMethod_OutParameters" to the block interface "UAMethod_OutParameters" of the post function.

```
13  //Call post
14 #OPC_UA_ServerMethodPost_Instance(UAMethod_Result:=#statPostResult,
15                                   UAMethod_Finished:=#statPostFinished,
16                                   Done=>#statPostDone,
17                                   Busy=>#statPostBusy,
18                                   Error=>#statPostError,
19                                   Status=>#statPostStatus,
20                                   UAMethod_OutParameters:=#UAMethod_OutParameters);
```

9. Make sure that the "Accessible from HMI / OPC UA" check boxes are set in the declaration of the pre- and post-function as well as in the input and output parameters.

	Name	Data type	Accessible from HMI/OPC UA	Writa...	Visible in ...
7	Static		☐	☐	☐
8	UAMethod_InParameters	Struct	☑	☑	☑
9	UAMethod_OutParameters	Struct	☑	☑	☑
10	OPC_UA_ServerMethodPre_Instance	OPC_UA_ServerMethodPre	☑	☑	☑
11	OPC_UA_ServerMethodPost_Instance	OPC_UA_ServerMethodPost	☑	☑	☑

2.1.4. Programming the functionality

The following instructions explain the basic procedure for programming the function of the OPC UA method:

1. Create a basic framework as described in chapter 2.1.3 Creation of the basic framework.
2. Create an IF query on the interface variables "statUamStatusPre.done" and "statUamMethodHandling.called". Both variables must be defined as "Static". Program the functionality of the method within this IF block so that the code is only executed when the method is called by a client.

```
32
33  REGION Method Code
34    IF #statUamStatusPre.done AND #statUamMethodHandling.called THEN
35      (*******************************************************
36      Part 1 Sort : Execute sort mechanism via "Bubble Sort".
37      *******************************************************)
38      REPEAT
39        #tempExchange := FALSE;
40        FOR #tempIndex := #LIMIT TO 1 BY -1 DO
41          IF #UAMethod_InParameters.sortbuffer[#tempIndex - 1] > #UAMethod_InParameters.sortbuffer[#tempIndex] THEN
42            #tempHelp := #UAMethod_InParameters.sortbuffer[#tempIndex];
43            #UAMethod_InParameters.sortbuffer[#tempIndex] := #UAMethod_InParameters.sortbuffer[#tempIndex - 1];
44            #UAMethod_InParameters.sortbuffer[#tempIndex - 1] := #tempHelp;
45            #tempExchange := TRUE;
```

3. Program the functionality inside the IF block. (In this example, a bubble sort of variables is implemented.).

```
32
33  REGION Method Code
34    IF #statUamStatusPre.done AND #statUamMethodHandling.called THEN
35      (*******************************************************
36      Part 1 Sort : Execute sort mechanism via "Bubble Sort".
37      *******************************************************)
38      REPEAT
39        #tempExchange := FALSE;
40        FOR #tempIndex := #LIMIT TO 1 BY -1 DO
41          IF #UAMethod_InParameters.sortbuffer[#tempIndex - 1] > #UAMethod_InParameters.sortbuffer[#tempIndex] THE
42            #tempHelp := #UAMethod_InParameters.sortbuffer[#tempIndex];
43            #UAMethod_InParameters.sortbuffer[#tempIndex] := #UAMethod_InParameters.sortbuffer[#tempIndex - 1];
44            #UAMethod_InParameters.sortbuffer[#tempIndex - 1] := #tempHelp;
45            #tempExchange := TRUE;
```

Note You can use the input and output interface of the function block for the functionality in order to interact with the rest of the user program.

4. Set the static interface variable "statUamMethodHandling.finished" to FALSE when your function code is completed. Additionally, assign an appropriate status code to the interface variable "statUAMethodHandling.result" . Please note the chapter 3.2 OPC UA Status Codes.

```
75  REGION POST
76    // Call of Method Post function
77    #OPC_UA_ServerMethodPost_Instance(UAMethod_Result := #statUamMethodHandling.result,
78                                      UAMethod_Finished := #statUamMethodHandling.finished,
79                                      Done => #statUamStatusPost.done,
80                                      Busy =>#statUamStatusPost.busy,
81                                      Error =>#statUamStatusPost.error,
82                                      Status => #statUamStatusPost.status,
83                                      UAMethod_OutParameters := #UAMethod_OutParameters);
84    // Errors in Post_Instance mostly occur when parameters are assigned wrong or OPC UA Server is not activ when us
85    // To monitor status in a watchtable, data is saved in static variable
86
87    IF #statUamStatusPost.done OR #statUamStatusPost.error THEN
88      #statUamMethodHandling.finished := FALSE;
89      #statUamMethodHandling.result := "OpcUa_Good";
90    END_IF;
91
92  END_REGION ;
93
```

5. You have created your OPC UA method. Call the function block in the user program so that the method appears in the OPC UA address space of your SIMATIC S7-1500. Please note the explanations on this in chapter 2.1.5 Call in the user program.

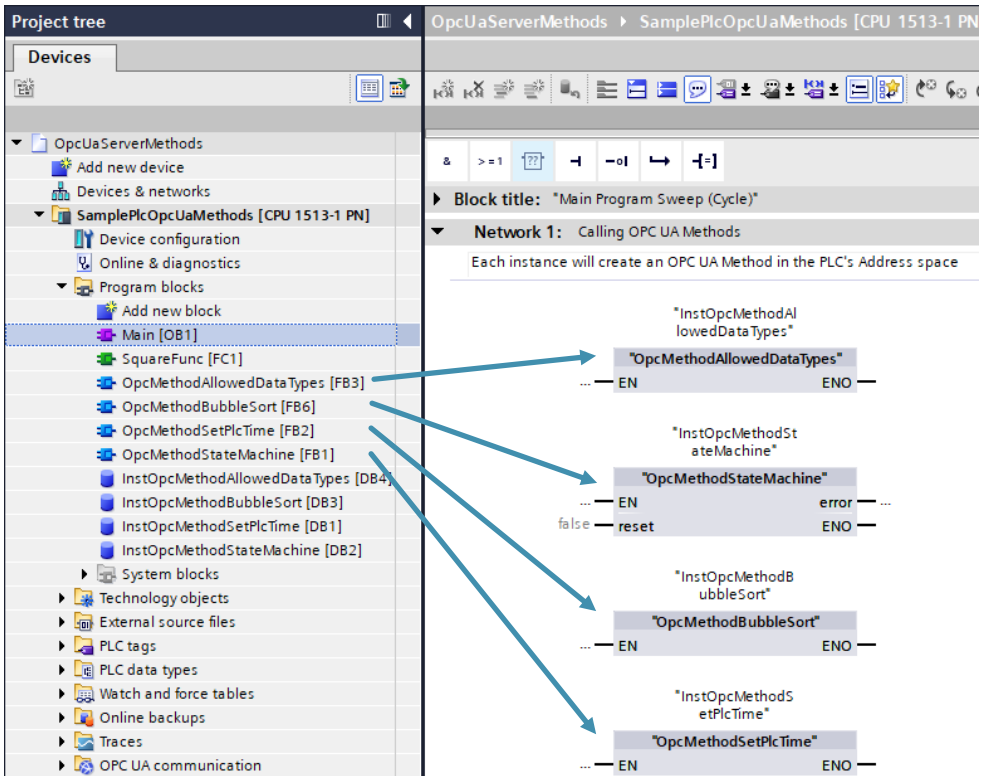
2.1.5. Call in the user program

To implement the OPC UA functionality of an S7 function block created for this purpose, call the block in the user program. Call the block in a cyclic OB ("program cycle") or in a cyclic interrupt OB ("cyclic interrupt").

Alternatively, you can call the block in a higher-level function block. However, you must also call the higher-level function block in one of the two permitted OB types.

The following figure shows an example of calling S7 function blocks in OB1 ("Main"). The individual instances of the blocks implement the OPC UA methods.

Figure 2-2



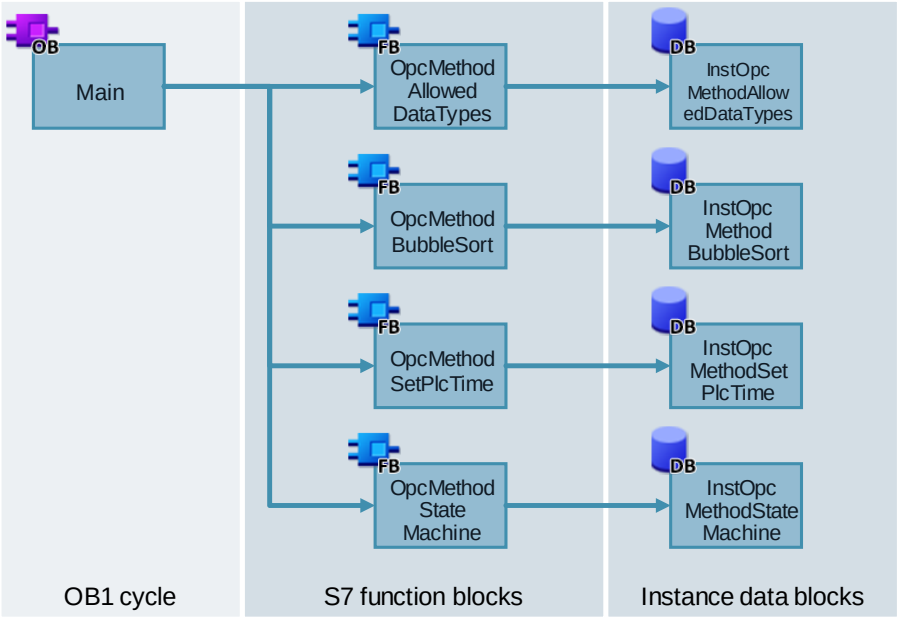
2.2. Explanations to the example project.

In this chapter, we will explain the example project provided with this.

2.2.1. Call structure

The following figure shows you the call hierarchy of the blocks of the example project:

Figure 2-3



All S7 function blocks are called directly in the OB1 cycle of the user program. Here, the instance data blocks of the individual function blocks implement the OPC UA methods.

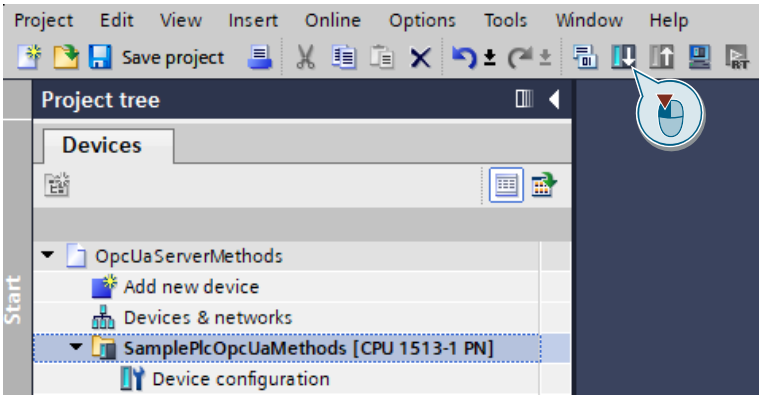
The following methods can be called by an OPC UA client in this example:

- "OpcMethodAllowedDataTypes":
  This method contains all permitted data types as input parameters of the method. During execution, the transferred values are provided as output parameters of the method.
- "OpcMethodBubbleSort":
  This method offers an array of type "Int" as input parameter. The transferred values are sorted by value size and the result is provided as an output parameter.
- "OpcMethodSetPlcTime":
  You can use this method to set the PLC time.
- "OpcMethodStateMachine"
  This method contains a step sequence that is started when the method is called. The method has no input or output parameters. While the step sequence is being processed, you receive a negative response when calling again.

2.2.2. Commissioning the example project

In order to commission the sample project, proceed as follows:

1. Download the project file (for TIA V15.1 or TIA V17) to your hard disk. The download can be found on the HTML page of this article (\2\).
2. Unzip the ZIP archive to any location.
3. In the unpacked archive, navigate to the folder "OpcUaServerMethods". This folder contains the TIA Portal project.
4. Open the project by double-clicking on the file "OpcUaServerMethods.apXX".
5. Select the CPU in the project navigation and click on "Download to device".



2.2.3. Operation of the example project

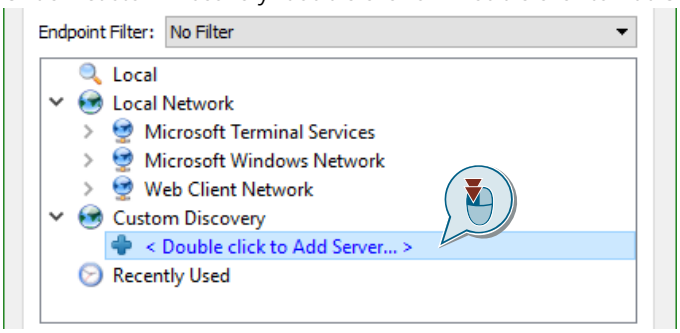
To test the OPC UA methods of the sample project, you can use any OPC UA client that controls the method functionality. This example uses the "UaExpert" tool from Unified Automation. The free download can be found in the appendix (\5\).

Establish the connection to the server

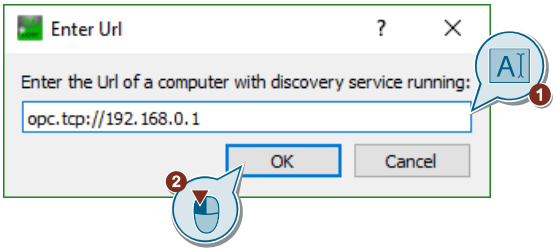
First, connect UaExpert to the server of the PLC:

1. Start the tool "UaExpert".
2. Click on the "Add server" button.

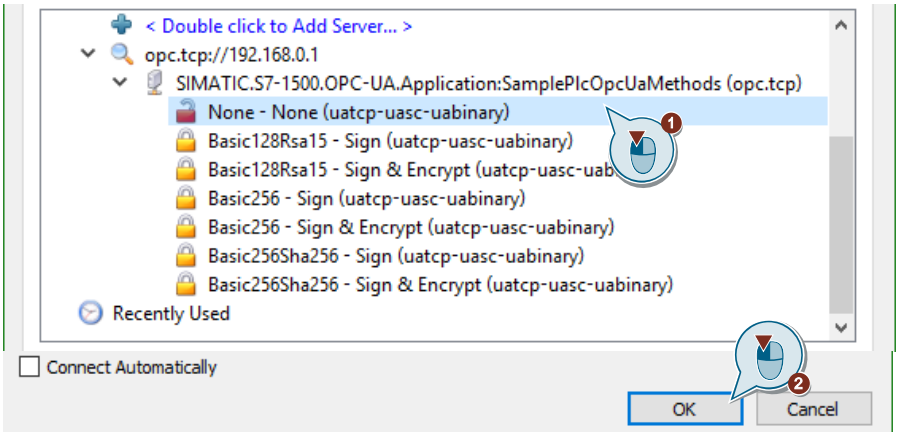
3. Under "Custom Discovery" double click on "Double click to Add Server"



4. Enter the IP address of the PLC in the text field of the dialog that appears and then confirm with "OK".

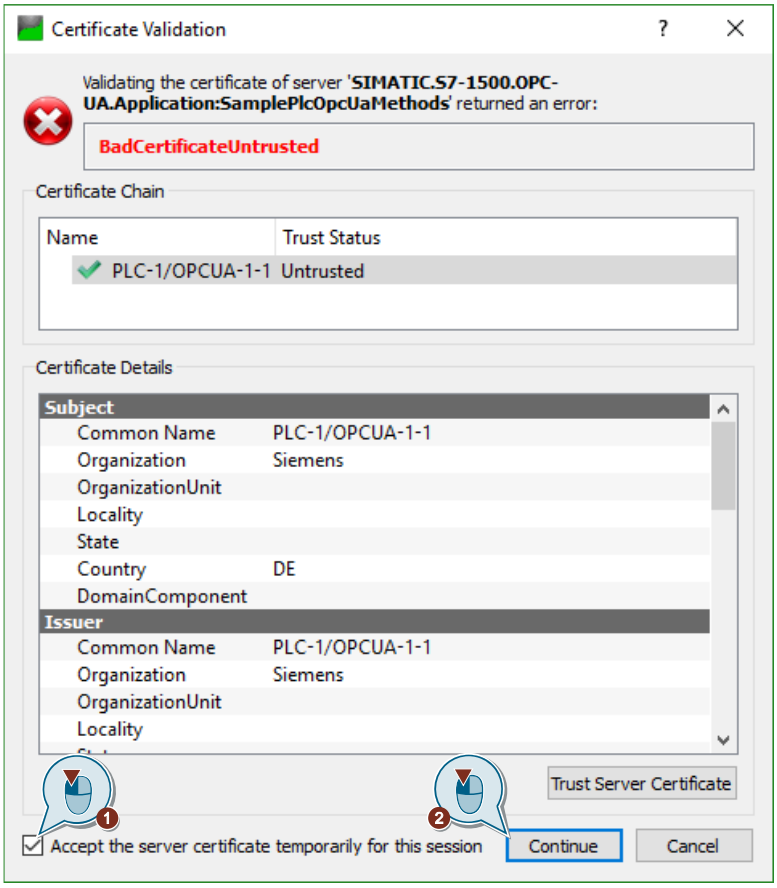


5. Select an end point ("None" in this example) and then confirm with "OK".



6. Select the selected endpoint in the project navigation and click on the "Connect Server" button.

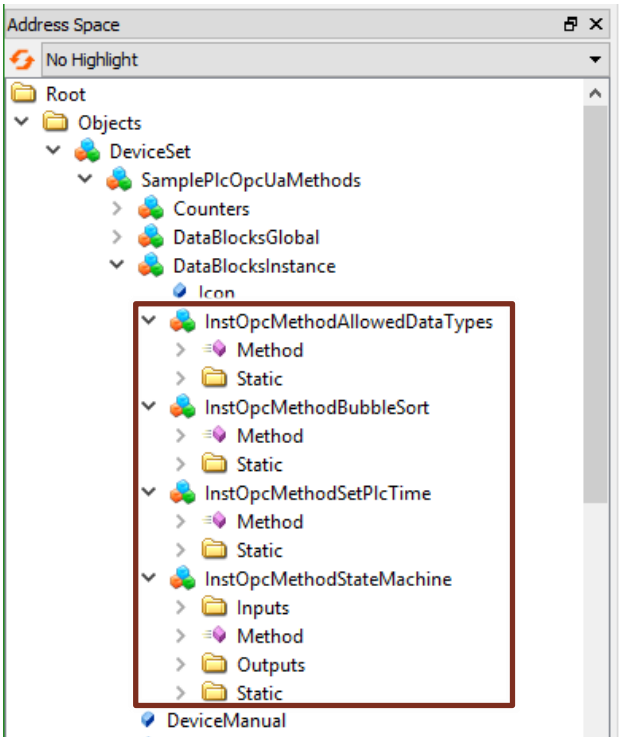
7. Confirm the certificate of the PLC. For this purpose, activate the checkbox "Accept the server certificate for this session" and confirm with "Continue". Alternatively, you can permanently accept the certificate by clicking the "Trust Server Certificate" button.



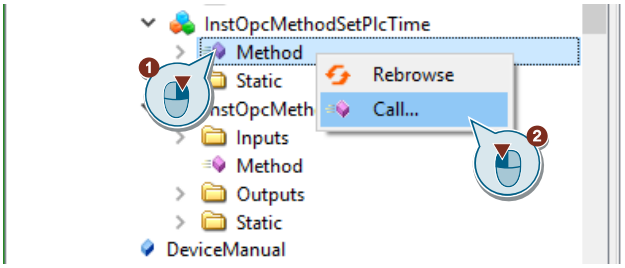
Call methods

Once you have established a connection to the server of the PLC, you can call up the methods created in advance:

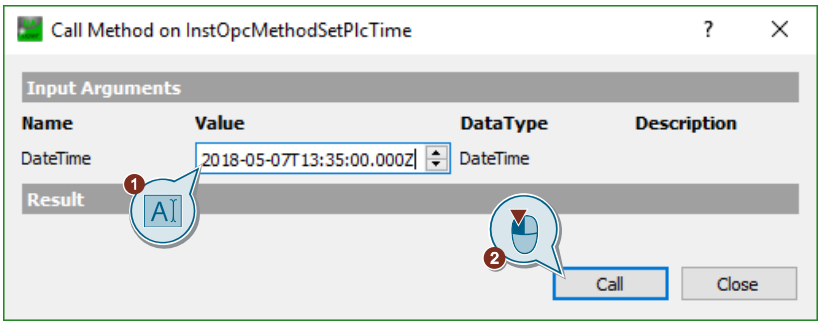
1. In the “Address Space” area, navigate to “Root> Objects> DeviceSet> SamplePlcOpcUaMethods> DataBlockInstance”. Within the objects of the block instances, you will find the OPC UA methods created in advance.



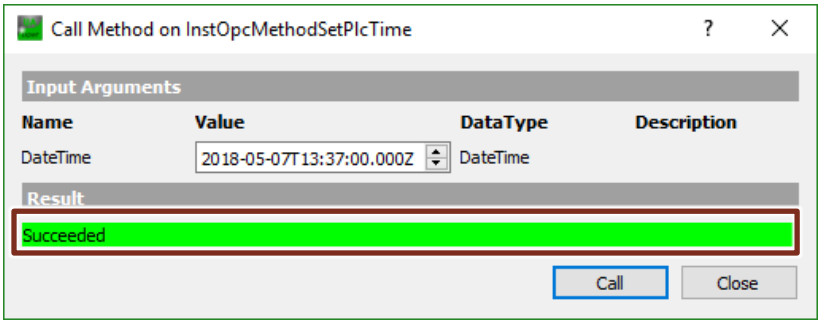
2. Right-click on a method (for example, within the instance of “InstOpcMethodSetPlcTime”), and then click “Call....”



3. In the dialog that appears, enter an input parameter of the method (in the example: The PLC time) and then click on "Call" to execute the method.



4. After the successful execution of the method, you will receive a message in the dialog. In case of a faulty execution, the error code will be displayed there. If the method has output parameters, these are also displayed in the dialog.



3. Useful information

In this chapter, we provide you with valuable information regarding the implementation of OPC UA methods on an S7-PLC.

3.1. Allowed data types as interface parameters

You can define the following PLC data types as input and output parameters ("UAMethod_InParameters" and "UAMethod_OutParameters") of your OPC UA methods:

Table 3-1

Approved data types	
BOOL	Boolean
SINT	SByte
INT	Int16
DINT	Int32
LINT	Int64
USINT	Byte
UINT	UInt16
UDINT	UInt32
ULINT	UInt64
REAL	Float
LREAL	Double
LDT	DateTime
WSTRING	String

Note You can declare a maximum of 20 input and output parameters. A structure or array is interpreted as just one parameter here.

3.2. OPC UA Status Codes

The OPC Foundation has defined a multitude of status codes for OPC UA. We recommend that you use these status codes for the OPC UA methods created in the TIA Portal.

In this application example, we provide you with an XML file with the most important status codes. Proceed as follows to import the file into the TIA Portal:

1. Download the archive "109756885_OpcUa_ServerMethods_XML_V1_0.zip" to your hard disk. The download can be found on the HTML page of this article (\2\).
2. Unzip the ZIP archive to any location.
3. Open your TIA Portal project and create a ("variable table") ("tag table").
4. Click the "Import" button in the workspace.
5. In the dialog that appears, navigate to the location of the unpacked archive and select the file "OpcUaStatusCodes.xml". Activate the check box "Constants" and confirm with "OK".

4. Appendix

4.1. 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:

4.2. Application support

Siemens AG
Digital Factory Division
Factory Automation
Production Machines
DF FA PMA APC
Frauenauracher Str. 80
91056 Erlangen, Germany

mailto: tech.team.motioncontrol@siemens.com

4.3. Links and Literature

Table 4-1

No.	Topic
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to the entry page for the application example https://support.industry.siemens.com/cs/ww/en/view/109756885
\3\	SIMATIC S7-1500, ET 200MP, ET 200SP, ET 200AL, ET 200pro Communication https://support.industry.siemens.com/cs/ww/en/view/59192925
\4\	OPC UA .NET client for the SIMATIC S7-1500 OPC UA server https://support.industry.siemens.com/cs/ww/en/view/109737901
\5\	Link to the download of the UaExpert tool https://www.unified-automation.com/products/development-tools/uaexpert.html

4.4. Change documentation

Table 4-2

Version	Date	Change
V1.0	05/2018	First edition
V1.1	07/2019	Upgrade to TIA Portal V15.1
V1.2	01/2022	Error corrections and upgrade to TIA Portal V17
V1.3	03/2025	Corrections and upgrade to TIA Portal V18