SIEMENS

Ingenuity for life

# S7 user block for the OPC UA client of a SIMATIC S7-1500

SIMATIC S7-1500 / FW V2.6 / OPC UA client module

https://support.industry.siemens.com/cs/ww/en/view/109762770

Siemens Industry Online Support

# Legal information

# Table of contents

# 1 Introduction

## 1.1 Overview

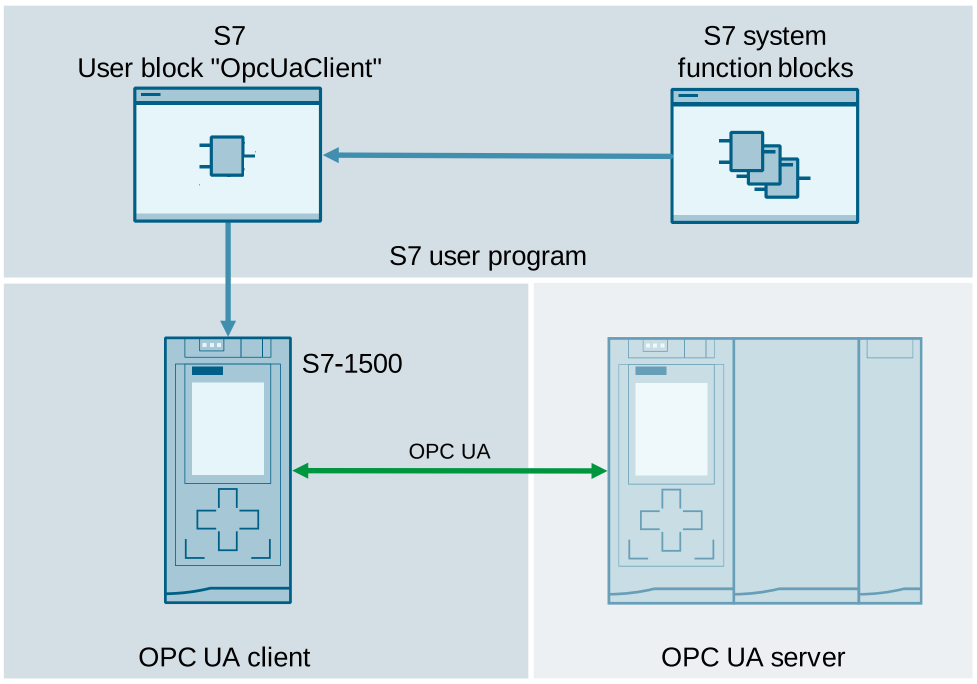
As of firmware V2.6 of the SIMATIC S7-1500 PLC family, an OPC UA client is introduced into the control system. With this extension you can implement M2M communication completely via OPC UA on a SIMATIC S7-1500 controller.

In contrast to the OPC UA server of the controller, the OPC UA client is programmed via many system function blocks and not only configured. The client supports all common security policies and modes as well as user authentication via username and password or anonymously.

With this application example we create for you the S7 user block "OpcUaClient", which summarizes the most important functions of the OPC UA system function blocks, accelerates the implementation for you and simplifies the programming.

The OPC UA server in the example is an S7-1500 controller with a simple simulation program for process values. Alternatively, you can use any other OPC UA server.

Figure 1-1



The following access types are supported by the OPC UA client and the user block:

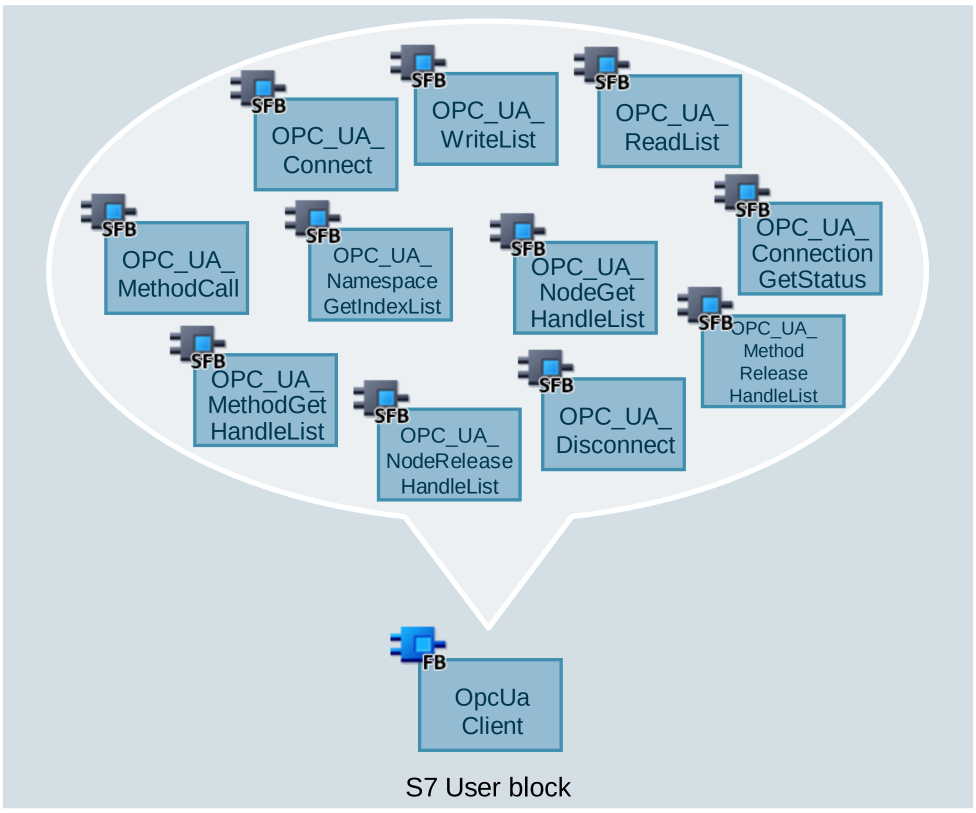
- Registered Read
- Registered Write
- Registered Method Call

## 1.2 How it works

To implement the OPC UA Client functionalities, the following system function blocks are called, parameterized and evaluated for you in the S7 user block "OpcUaClient":

- Connection setup
  - OPC_UA_Connect
  - OPC_UA_NamespaceGetIndexList
  - OPC_UA_NodeGetHandleList
  - OPC_UA_MethodGetHandleList
- Read, write, call methods
  - OPC_UA_ReadList
  - OPC_UA_WriteList
  - OPC_UA_MethodCall
- Disconnection
  - OPC_UA_NodeReleaseHandleList
  - OPC_UA_MethodReleaseHandleList
  - OPC_UA_Disconnect
- Diagnostics
  - OPC_UA_ConnectionGetStatus

Figure 1-2


SFB OPC_UA_ Connect
SFB OPC_UA_ WriteList
SFB OPC_UA_ ReadList
SFB OPC_UA_ MethodCall
SFB OPC_UA_ Namespace GetIndexList
SFB OPC_UA_ NodeGet HandleList
SFB OPC_UA_ Connection GetStatus
SFB OPC_UA_ Method Release HandleList
SFB OPC_UA_ MethodGet HandleList
SFB OPC_UA_ NodeRelease HandleList
SFB OPC_UA_ Disconnect
FB OpcUa Client
S7 User block

The following table gives you an overview of the functions of the individual SFBs within the S7 user block "OpcUaClient":

Table 1-1

| SFB | Function |
|---|---|
| OPC_UA_Connect | Establishes a connection and session to an OPC UA server |
| OPC_UA_Disconnect | Establishes a connection and session to an OPC UA server |
| OPC_UA_Namespace GetIndexList | Queries the current indices of the namespaces of the connected OPC UA server. |
| OPC_UA_Node GetHandleList | Registers the OPC UA node IDs to be read or written on an OPC UA server. |
| OPC_UA_Method GetHandleList | Registers the OPC UA method node IDs to be called on an OPC UA server. |
| OPC_UA_ReadList | Reads the registered variables of an OPC UA server |
| OPC_UA_WriteList | Writes the registered variables of an OPC UA server |
| OPC_UA_MethodCall | Calls a registered method of an OPC UA server |
| OPC_UA_Node ReleaseHandleList | Releases the registered Node-IDs on an OPC UA-Server again |
| OPC_UA_Method ReleaseHandleList | Releases the registered method node IDs on an OPC UA server again |
| OPC_UA_Connection GetStatus | Returns the quality of the connection to an OPC UA server. |

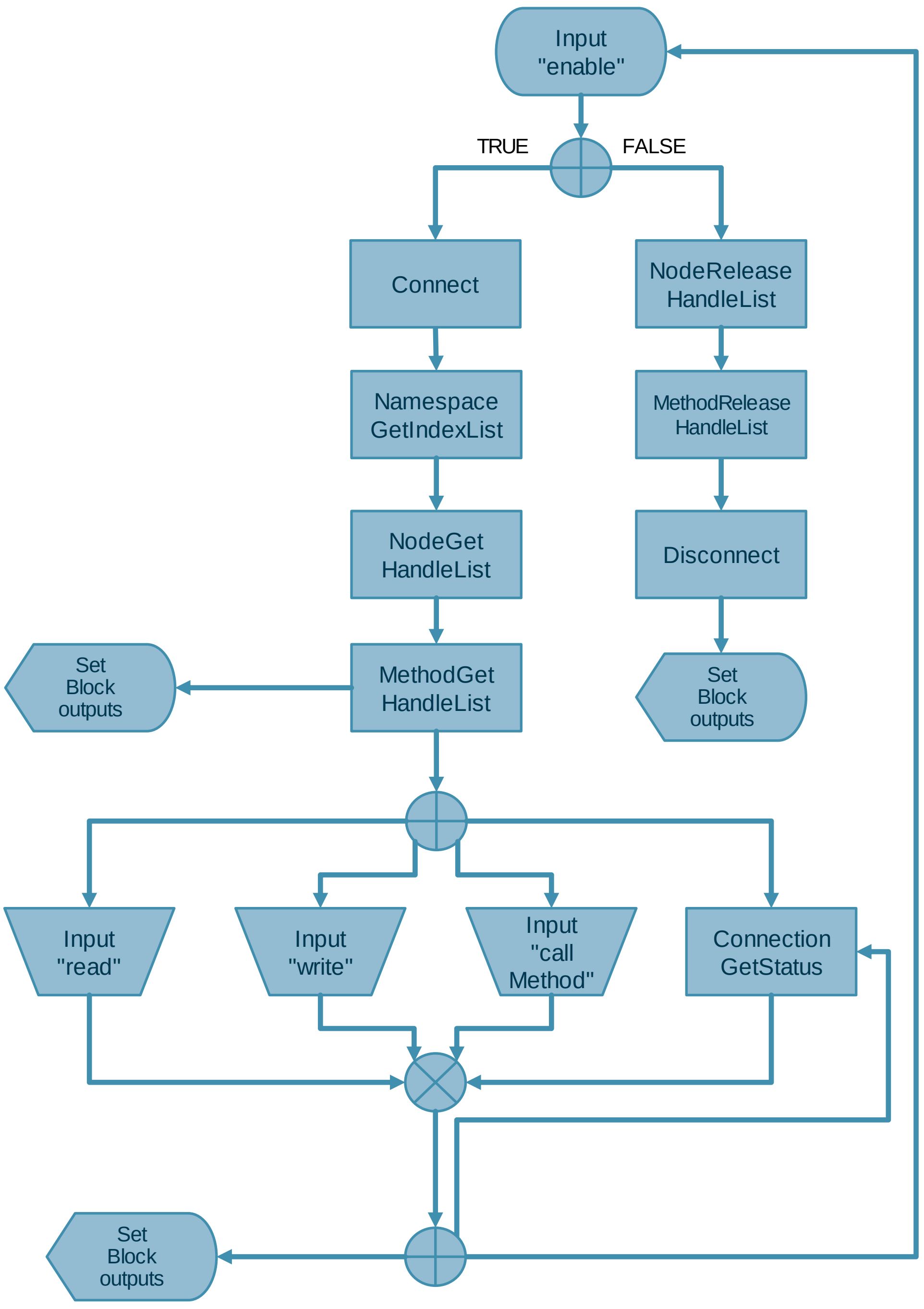
## Functional sequence

After you have parameterized and called up the block, you only need four input parameters to control the OPC UA functions: "enable", "read", "write" and "callMethod".

Via the input parameter "connect" you establish and terminate a connection/session. The "read" input starts a read job, the "write" input a write job. "CallMethod" calls a method on the connected OPC UA server.

The device diagnoses and maintains the connection to the server for you and reconnects automatically if the connection is terminated. In the event of malfunction, you will be informed via the block outputs.

Figure 1-3



## 1.3 Components used

The Application Example has been created with the following hardware and software components:

Table 1-2

| Components | Quantity | Article number | Note |
|---|---|---|---|
| S7-1500<br>CPU 1513-1 PN/DP | 1 | 6ES7513-1AL01-0AB0 | Client:<br>FW 2.6 for TIA Portal V15.1<br>FW 2.8 for TIA Portal V16 |
| S7-1500<br>CPU 1516F-3 PN/DP | 1 | 6ES7 516-3FN01-0AB0 | Server:<br>From FW 2.0 |
| STEP 7 Professional V15.1 | 1 | 6ES7822-1AA05-0YA5 | TIA Portal V15.1 |
| STEP 7 Professional V16 | 1 | 6ES7822-1AA06-0YA5 | TIA Portal V16 |

This application example consists of the following components:

Table 1-3

| Components | File name | Note |
|---|---|---|
| Documentation | 109762770_OPC_UA_PLC-Client_<br>DOC_V1_2_1_en.pdf | This document |
| Example project | 109762770_OPC_UA_PLC-Client_<br>15_PROJ_V1_2.zip | This ZIP archive contains the sample project for TIA Portal V15.1 and FW 2.6 |
| Example project | 109762770_OPC_UA_PLC-Client_<br>16_PROJ_V1_3.zip | This ZIP archive contains the sample project for TIA Portal V16 and FW 2.8 |

**Hinweis**

The included projects for TIA Portal V15.1 (S7-1500 FW 2.6) and V16 (from S7 1500 FW 2.8) differ in the connection monitoring process. Make sure to select the appropriate project and firmware.

If you use TIA Portal V16 with an S7-1500 with FW 2.6, use the project for TIA Portal V15 and upgrade it to TIA Portal V16.

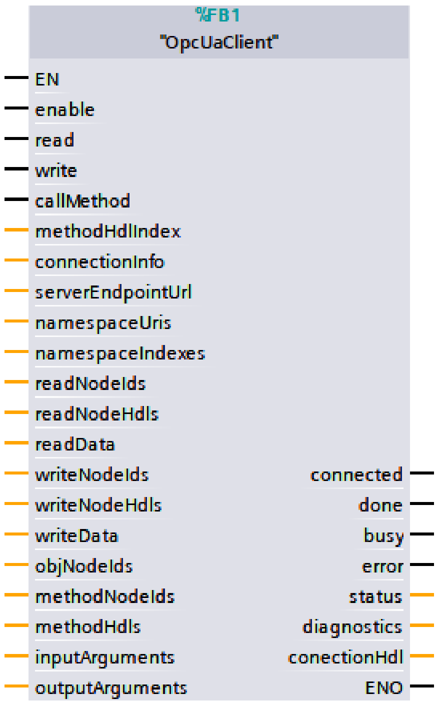
# 2 Engineering

## 2.1 Block description

### 2.1.1 Interface description

The following figure shows the OPC UA client module "OpcUaClient":

Figure 2-1



The following table explains the input and InOut parameters of the block:

Table 2-1

| Parameters | Data type | Description |
|---|---|---|
| enable | Bool | TRUE = Establish connection<br>FALSE = disconnection |
| read | Bool | Starts read job with positive edge |
| write | Bool | Starts write job with positive edge |
| methodHdlIndex | Int | The method to be called from the method list |
| callMethod | Bool | Calls the method of "methodHdlIndex" with positive edge |
| connectionInfo | "OPC_UA_Session ConnectInfo" | Connection Information for Session Setup with the OPC UA Server |
| serverEndpointUrl | String | The Endpoint URL of the OPC UA Server |
| namespaceUris | Variant | Pointer to the memory area of the queried namespace URIs. |
| namespaceIndexes | Variant | Pointer to the memory area of the queried namespace indexes. |
| readNodeIds | Variant | Pointer to the array of "OPC_UA_NodeId" of the node IDs to be read. |

| Parameters | Data type | Description |
|---|---|---|
| readNodeHdls | Variant | Pointer to the Array of DWORD of the node handles to be read. |
| readData | Variant | Pointer to the memory area (UDT or STRUCT) of the read values. |
| writeNodeIds | Variant | Pointer to the array of "OPC_UA_NodeId" of the node IDs to be written. |
| writeNodeHdls | Variant | Pointer to the memory area (array of DWORD) of the node handles to be written. |
| writeData | Variant | Pointer to the memory area (UDT) of the values to be written. |
| objNodeIds | Variant | Pointer to the array of "OPC_UA_NodeId" of the object node IDs of methods to be called. |
| methodNodeIds | Variant | Pointer to the Array of DWORD of the node IDs of methods to be called. |
| methodNodeHdls | Variant | Pointer to the memory area (DWORD) of the method handle to be called. |
| inputArguments | Variant | Pointer to the input parameters (UDT or STRUCT) of the method to be called. |
| outputArguments | Variant | Pointer to the outbound parameters (UDT or STRUCT) of the method to be called. |

**Note** If you use several methods, you must adapt the InOut parameters "inputArguments" and "outputArguments" to the methods to be called at runtime. Therefore note chapter "3.4 Tips & tricks".

The following table explains the output parameters of the block:

Table 2-2

| Parameters | Data type | Description |
|---|---|---|
| connected | Bool | TRUE = connected to server and session established |
| done | Bool | TRUE = the last job was completed without errors |
| busy | Bool | TRUE = an order is being processed |
| error | Bool | TRUE = the last job was terminated with an error |
| status | DWord | Cause of the error |
| diagnostics | "typeFbDiagnostics" | Output for extended diagnostics |
| connectionHdl | DWord | Connection handle of the current session for use with further calls of the OPC UA system function modules. |

## 2.1.2 Functional description

The S7 user module "OpcUaClient" implements the OPC UA client functions for you. The module contains step chains in which the OPC UA system function modules are called and evaluated. The block requires the client interface configured in the TIA Portal, which provides all information for OPC UA communication. For further information on the client interface, refer to section 2.2.2 Creating an OPC UA Client Interface.

The following illustrations explain the dependencies and workflow between the OPC UA system function modules in general:

Figure 2-2 Connection setup

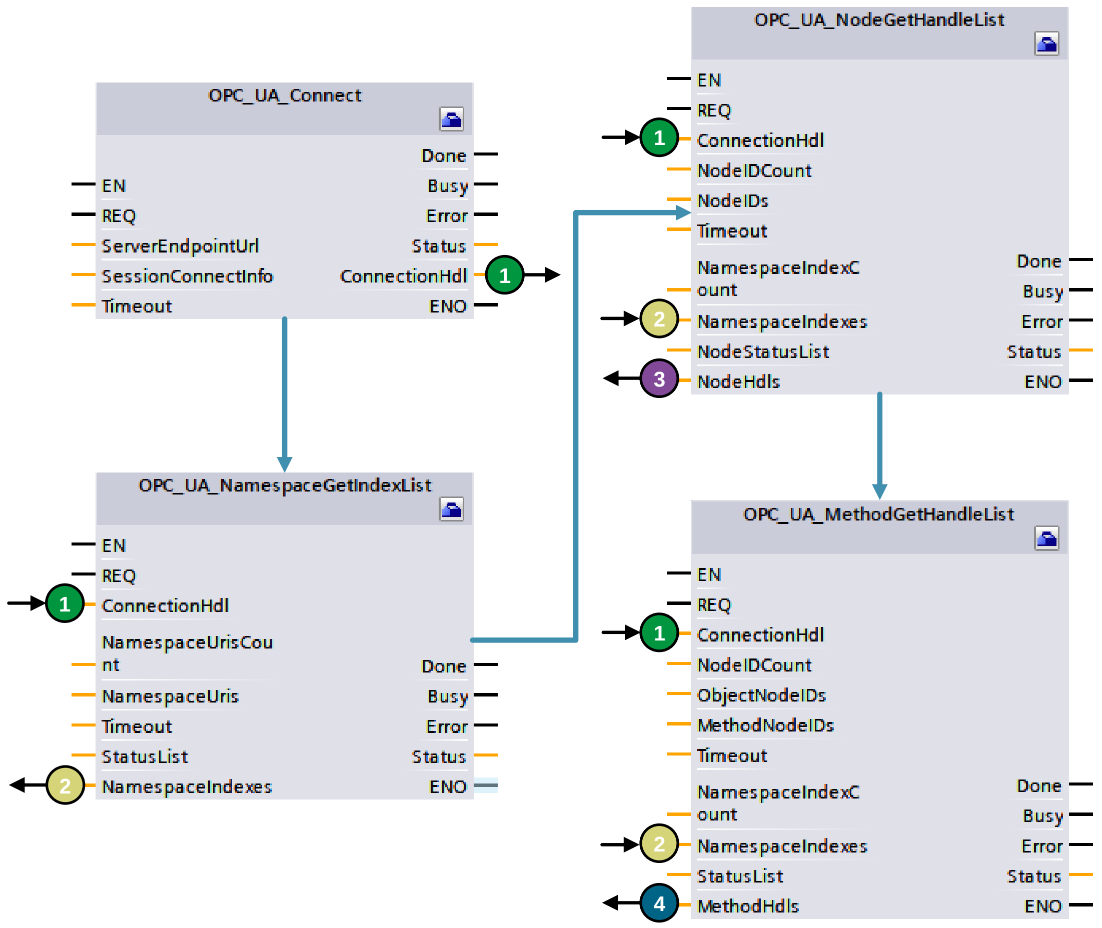


Figure 2-3 Reading/Writing Variables and Calling Methods

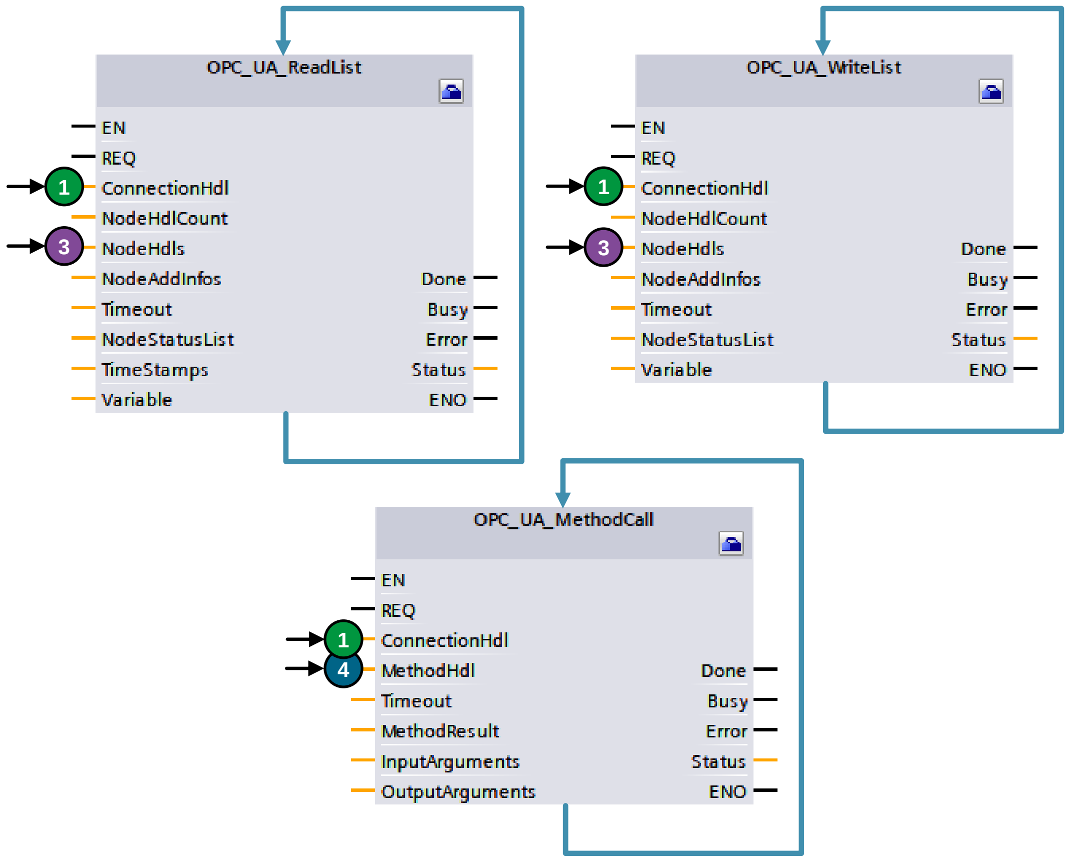
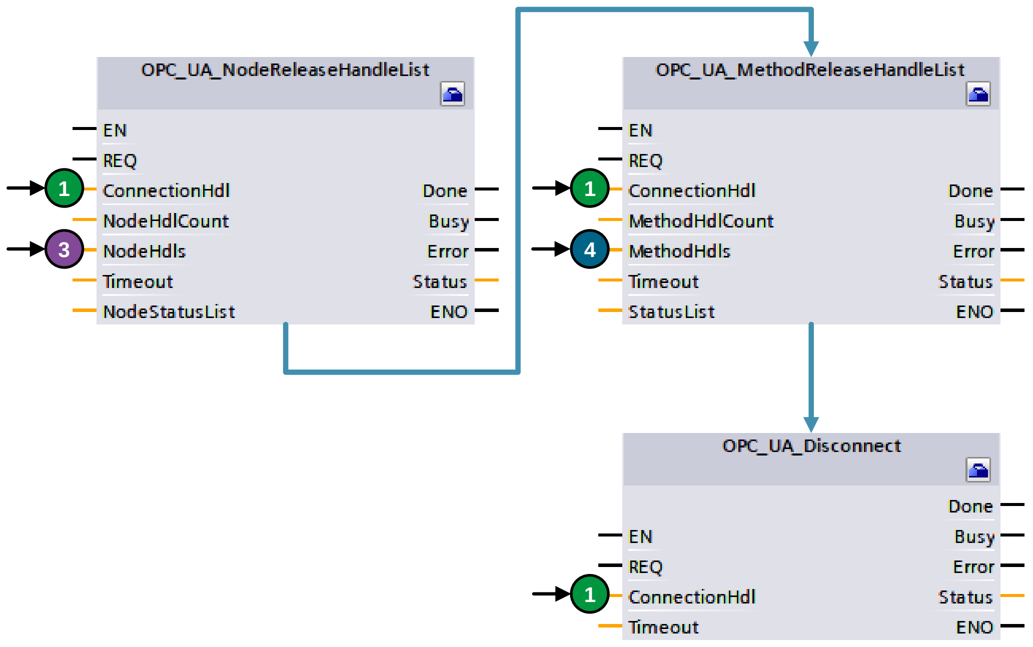


Figure 2-4 Approving disconnections and resources



### 2.1.3 Functional sequence

The following explanations explain the sequence of functions within the block "OpcUaClient":

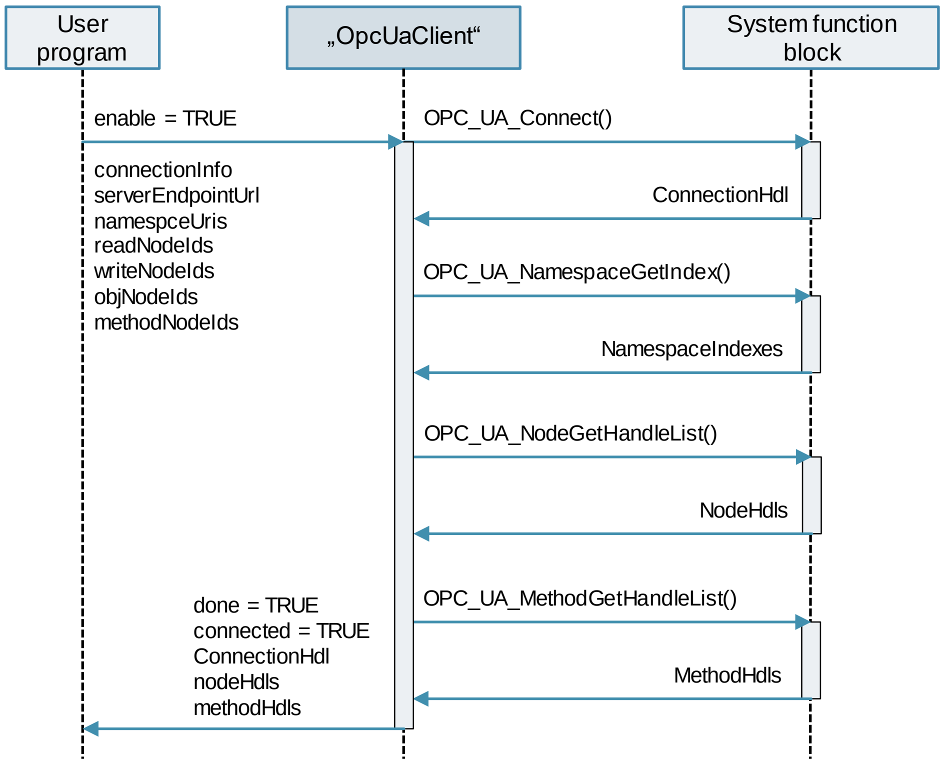
#### Connection setup

The system function block "OPC_UA_Connect" is executed to establish the connection. This module establishes a connection to the server and activates a session. After successful connection establishment, the block returns a "Connection Handle". This handle references the existing connection/session for all further OPC UA functions.

After the block has established the connection, the system function block "OPC_UA_NamespaceGetIndex" is executed. This block returns the namespace indexes for the nodes configured in the client interface. The indexes are mandatory for the addressing of the nodes in the later course.

The nodes of the read, write and method lists are then registered on the server. For this, the system function modules "OPC_UA_NodeGetHandleList" and "OPC_UA_MethodGetHandleList" are executed. For each registered node, a handle is returned that is needed to read or write the node later.

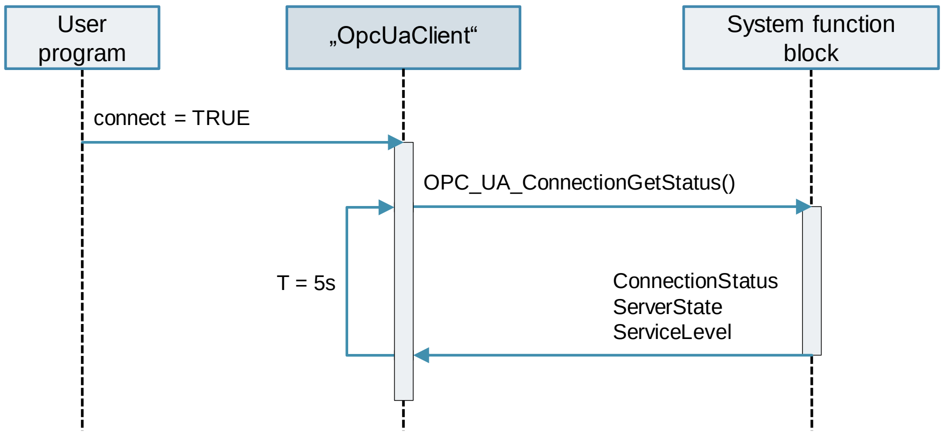
Figure 2-5



## Connection monitoring

For connection monitoring, the system function block "OPC_UA_ConnectionGetStatus" is called cyclically (5 s) after you have successfully established a connection. After each call, the data supplied by the block is evaluated in order to diagnose connection breakdowns or other errors. Further information on connection monitoring can be found in section 3.1 Connection monitoring in the "OpcUaClient" block.

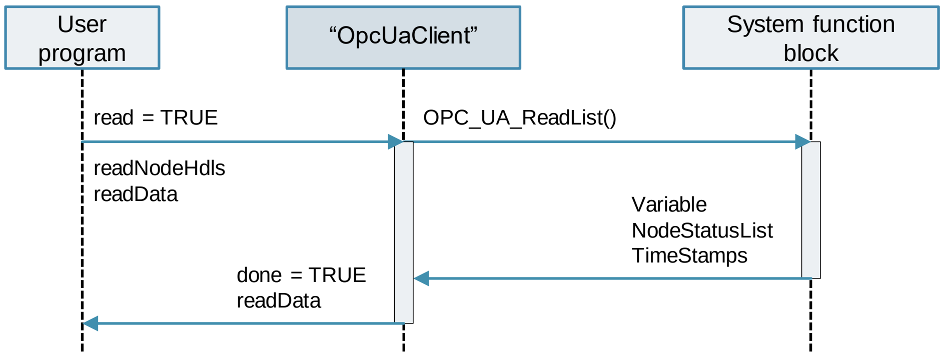
Figure 2-6



## Read variables

To read the variables from the server, the system function module "OPC_UA_ReadList" is executed. The block uses the handles registered during connection establishment to read the variables via the RegisteredRead service. The SFB returns the values of each node. In addition, a status code and a time stamp are output for each node. The user block "OpcUaClient" evaluates the status codes for you and sets the corresponding outputs in case of an error ("error" = "TRUE"; "errorId" = "71", "status" = "<StatusCode>").

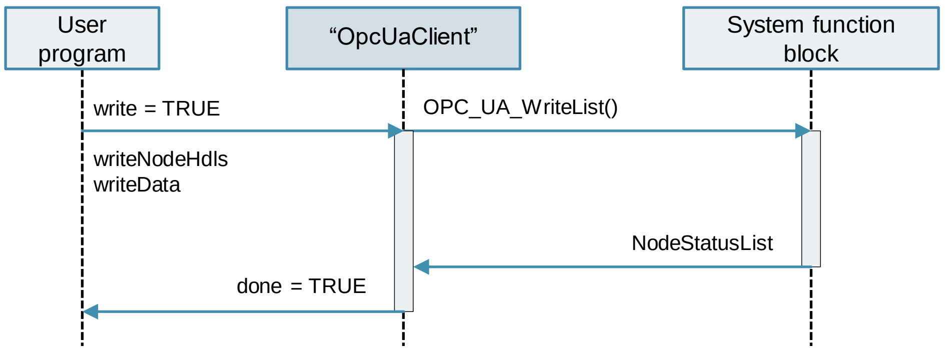
Figure 2-7



### Writing Variables

To write the variables to the server, the system function module "OPC_UA_WriteList" is executed. The block uses the handles registered during connection establishment to write the variables via the "RegisteredWrite" service. The SFB returns one status code per written node. The user block "OpcUaClient" evaluates the status codes for you and sets the corresponding outputs in case of an error ("error" = "TRUE"; "errorId" = "72", "status" = "<StatusCode>").

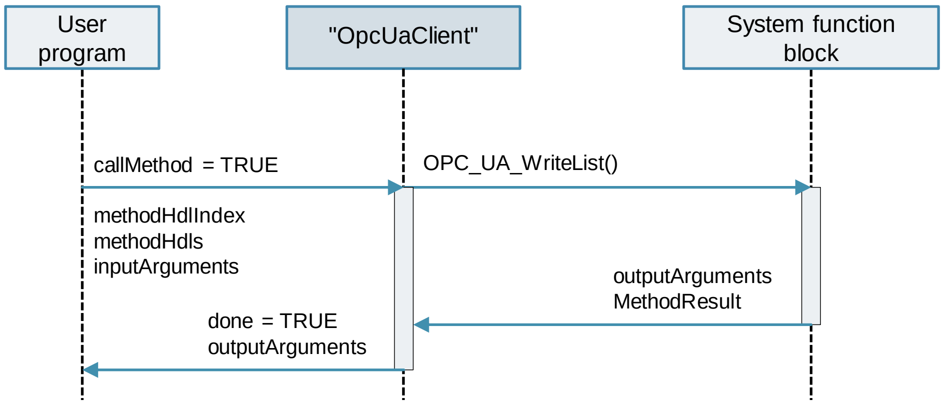
Figure 2-8



### Call methods

To execute the method on the server, the system function module "OPC_UA_MethodCall" is executed. The block uses the handles registered during connection establishment and the values at the input "inputArguments" to call the methods via the "RegisteredMethodCall" service. The SFB returns a method result. The user block "OpcUaClient" evaluates this result for you and sets the corresponding outputs ("error" = "TRUE"; "errorId" = "73", "status" = "<MethodResult>") in the event of an error. In addition, the return values of the method are output at the output "outputAruguments".

Figure 2-9

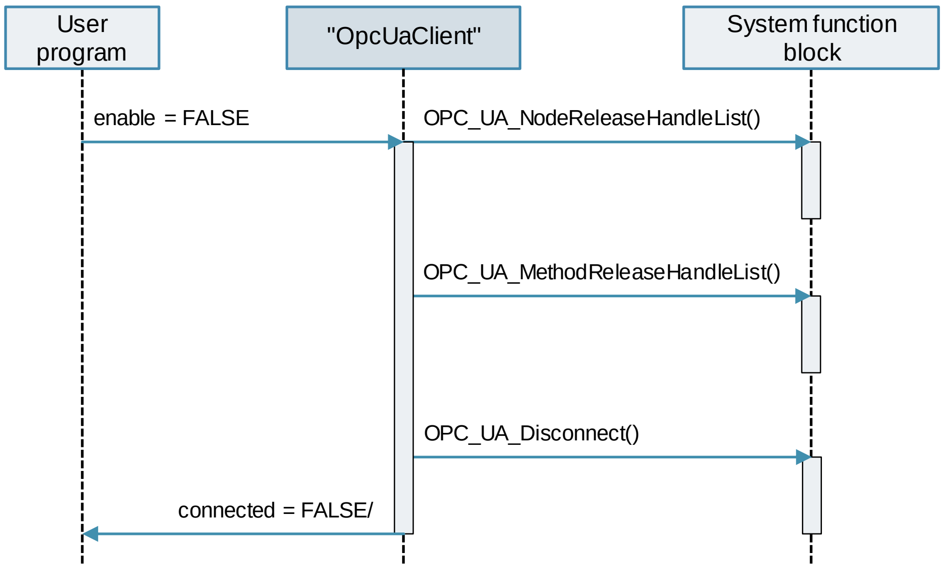


## Disconnection

The system function block "OPC_UA_Disconnect" is executed to terminate the connection. This block closes the connection to the server, deactivates the existing session and releases used resources of the client CPU.

Before the connection is terminated, the system function modules "OPC_UA_NodeReleaseHandleList" and "OPC_UA_MethodReleaseHandleList" are executed. These SFBs release the handles that were created by the server for all nodes of the read, write and method lists when the connection was established.

Figure 2-10



### 2.1.4 Device diagnostics

The following explanations describe the diagnostics of the FB "OpcUaClient".

#### Output parameters without error

The following table explains the values of the "status" parameter depending on "done" and "busy" when there is no error:

Table 2-3

| "status" | "done" | "busy" | Description |
|---|---|---|---|
| 16#0000_0000 | FALSE | FALSE | Client not connected. |
| 16#0000_0000 | TRUE | FALSE | Job successfully executed. |
| 16#0000_7000 | FALSE | FALSE | Client connected and ready for job. |
| 16#0000_7000 | TRUE | FALSE | Connection successfully established and ready for job. |
| 16#0000_7001 | FALSE | TRUE | Job is running. |

#### Output parameters with error

The interface parameters "error", "status" and "diagnostics" are used to diagnose the block. If an error is present, "error" is set. The output "status" provides the corresponding error source of the FB. The extended diagnostic information is available at output "diagnostics". The values of this structure remain until the next error.

The following table explains the structure of the variable "diagnostics":

Table 2-4

| Variable | Description |
|---|---|
| status | Last status code of the interface parameter "status" of the FB "OpcUaClient". |
| subfunctionStatus | Status or return value of the internal SBF at which the error occurred. For detailed information, refer to the online help ("F1") for the respective SBF or the status lists in the instance data block.<br>(16#80FF_0000 = read list, write list or method list not registered) |

The following table explains the error sources for the status codes of the interface parameter "status":

Table 2-5

| Status code | Source of errors |
|---|---|
| 16#8601 | Error during connect (SFB "OPC_UA_Connect"). |
| 16#8602 | Error during read of Namespace Index (SFB "OPC_UA_NamespaceGetIndexList"). |
| 16#8604 | Error during read of a read list (SFB "OPC_UA_ReadList"). |
| 16#8605 | Error during write of a write list (SFB "OPC_UA_WriteList"). |
| 16#8606 | Error during method call from a method list (SFB "OPC_UA_MethodCall"). |
| 16#8608 | Error during disconnect (SFB "OPC_UA_Disconnect"). |

| Status code | Source of errors |
|---|---|
| 16#8610 | Error during connection monitoring (SFB "OPC_UA_ConnectionGetStatus"). |
| 16#8611 | Error during connection monitoring related to the connection state ("CSTATE"). |
| 16#8612 | Error during connection monitoring related to the server state ("SSTATE"). |
| 16#8613 | Error during connection monitoring related to the service level ("SLVL"). |
| 16#8614 | Error in the status list after read of read list. |
| 16#8615 | Error in the status list after write of write list |
| 16#8616 | Error at serialization during method call (SFB "Serialize"). |
| 16#8617 | Error in the status list after read of namespace indexes. |
| 16#8626 | Fehler im OPC UA-Status eines Methodenaufrufs. |
| 16#8631 | Error during registration of read list (SFB "OPC_UA_NodeGetHandleList"). |
| 16#8632 | Error during registration of write list (SFB "OPC_UA_NodeGetHandleList"). |
| 16#8633 | Error during registration of method list (SFB "OPC_UA_MethodGetHandleList"). |
| 16#8671 | Error during release of read list (SFB "OPC_UA_NodeReleaseHandleListe"). |
| 16#8672 | Error during release of write list (SFB "OPC_UA_NodeReleaseHandleListe"). |
| 16#8673 | Error during release of method list (SFB "OPC_UA_MethodReleaseHandleListe"). |
| 16#86F4 | Error during read of read list; List not registered |
| 16#86F5 | Error during read of write list; List not registered |
| 16#86F6 | Error during method call from method list; List not registered |
| 16#8711 | Error during connection monitoring related to the connection state ("CSTATE"); Connection shut down. |
| 16#8731 | Error in the status list after registration of read list. |
| 16#8732 | Error in the status list after registration of write list. |
| 16#8733 | Error in the status list after registration of method list. |
| 16#8771 | Error in the status list after release of read list. |
| 16#8772 | Error in the status list after release of write list. |
| 16#8773 | Error in the status list after release of method list. |

## 2.2 Project planning and programming

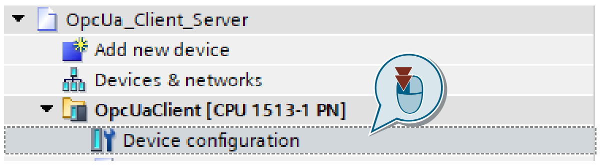
Before you can use the OPC UA module, you must perform the following steps:

- Activating the OPC UA Client in the Device Configuration
- Creating an OPC UA Client Interface
- Parameterizing the OPC UA "OpcUaClient" Module

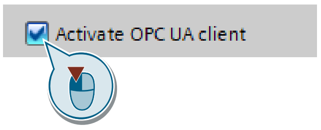
### 2.2.1 Activate OPC UA Client

To be able to use the functions of the OPC UA client, you have to activate the client in the properties of the CPU. Proceed as follows:

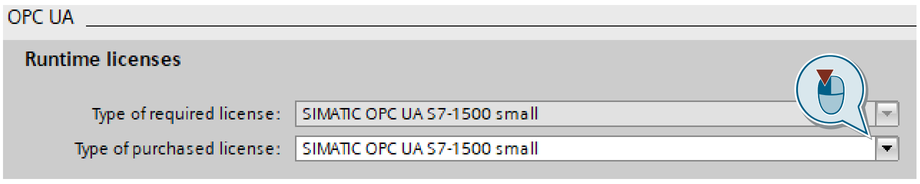
1. Navigate in the project tree in TIA Portal to the "Device configuration" of your CPU.



2. Navigate in the Inspector window to "OPC UA > Client > General" ("OPC UA > Client > General") and activate the checkbox "Activate OPC UA client".



3. Then confirm that you have the required license for OPC UA. Navigate to "Runtime licenses" and select the appropriate license in the category "OPC UA".



OPC UA

Runtime licenses

Type of required license: SIMATIC OPC UA S7-1500 small

Type of purchased license: SIMATIC OPC UA S7-1500 small

**Note** Depending on the used SIMATIC S7-1500 CPU you need a suitable OPC UA license ("small", "medium" or "large"). See section 3.3 Quantity Structure and Licenses of the OPC UA Client.

## 2.2.2 Creating an OPC UA Client Interface

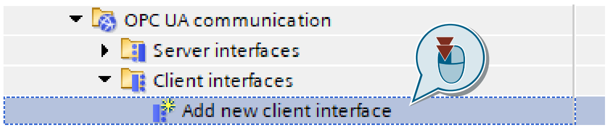
To use the OPC UA functions of the S7 function blocks, you must first create a client interface. The interface contains all relevant information required by the client module "OpcUaClient". The TIA Portal stores this information in two automatically generated data blocks:

- "<InterfaceName>_Configuration"
  This DB contains the connection information, node IDs and data types.
- "<InterfaceName>_Data"
  The values read or to be written are stored in this DB. The DB also contains a status list for the individual variables.

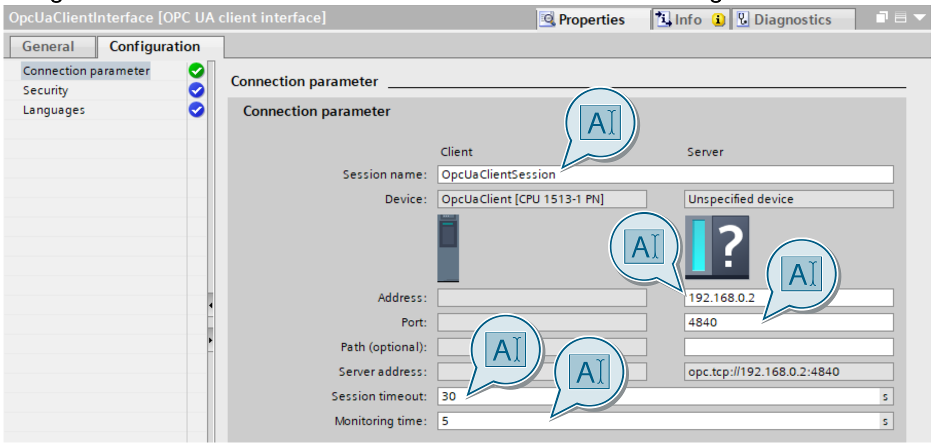
### Configuring Connection Parameters, Authorization and Authentication

Proceed as follows to configure the connection information:

1. Navigate to "OPC UA communication > Client interfaces" ("OPC UA communication > Client interfaces") in the project tree in TIA Portal and double-click on "Add new client interface".



2. Enter a session name ("Session name"), the "Address" and the "Port" of the server in the inspector window under "Properties > Configuration > Connection parameters" ("Properties > Configuration > Connection parameter"). Also assign a useful "session timeout" and the desired "monitoring time".

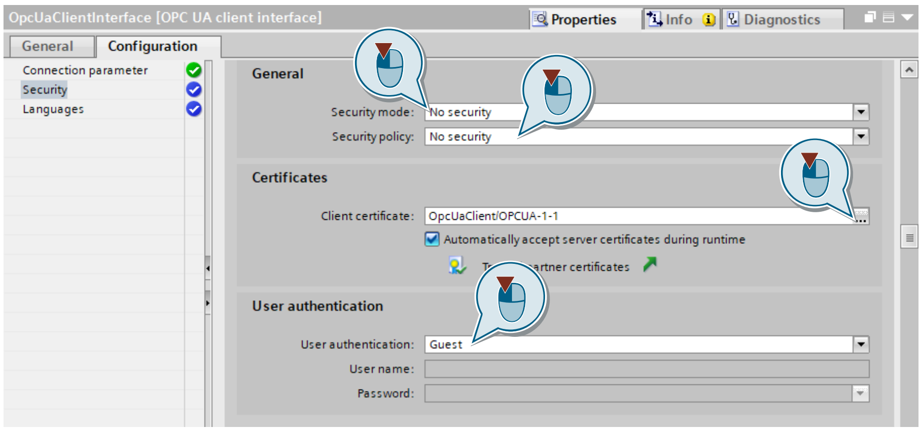


Connection parameter

Connection parameter

| | Client | Server |
|---|---|---|
| Session name: | OpcUaClientSession | |
| Device: | OpcUaClient [CPU 1513-1 PN] | Unspecified device |
| Address: | | 192.168.0.2 |
| Port: | | 4840 |
| Path (optional): | | |
| Server address: | | opc.tcp://192.168.0.2:4840 |
| Session timeout: | 30 s | |
| Monitoring time: | 5 s | |

3. In the Inspector window, under "Properties > Configuration > Security" ("Properties > Configuration > Security "), select the required "Security mode" and the "Security policy" Also create a "client certificate" if you want to establish a signed or encrypted connection and configure the required "user authentication".



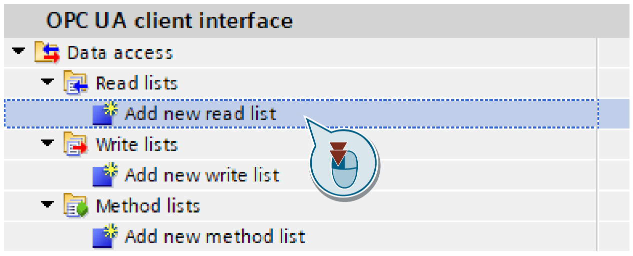
Security mode: No security
Security policy: No security
Client certificate: OpcUaClient/OPCUA-1-1
☑ Automatically accept server certificates during runtime
User authentication: Guest
User name:
Password:

**Note** The security of OPC UA is not further explained in this document.

## Configuring Read Lists, Write Lists, and Method Lists

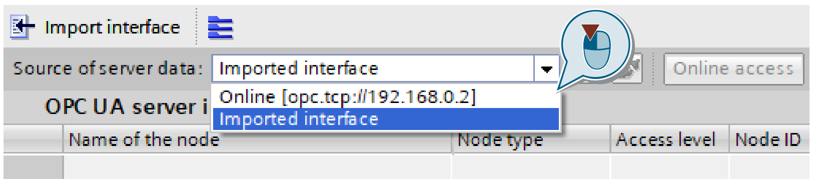
Proceed as follows to configure the lists:

1. In the TIA Portal workspace, navigate to Data access > Read lists in the left pane, and double-click Add new read list.



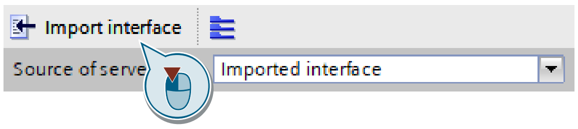
2. You now have two options for filling the reading list:
Via an imported server interface as an XML file (a) or via an online connection directly to a server (b).

In the Source of server data drop-down list on the right side of the workspace, select one of the two options.



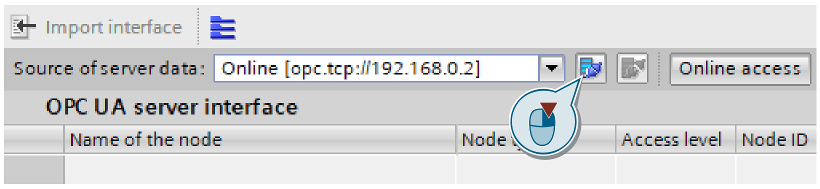
Source of server data: Imported interface
Online [opc.tcp://192.168.0.2]
Imported interface

a. Click the Import interface button if you are using the Imported interface option. In the dialog that appears, select the XML file of your server interface and confirm with "Import".



Imported interface

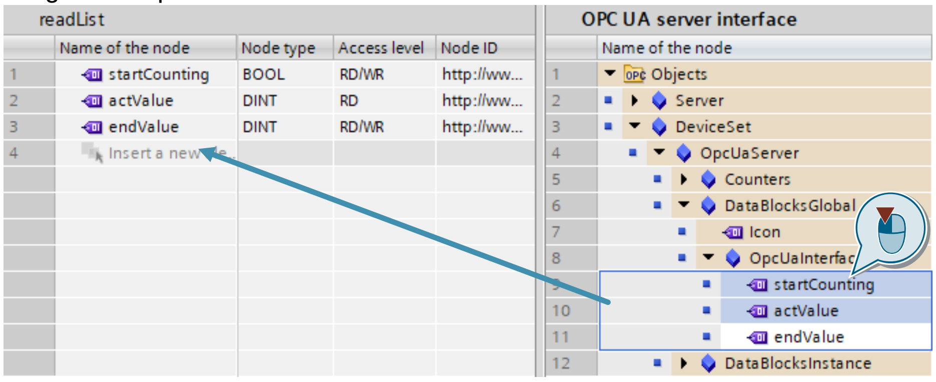
b. Click the "Connect to online server" button if you are using the "Online" option.



TIA Portal establishes an online connection to the already configured OPC UA server. Alternatively, you can use the "Online access" button to establish a connection to another OPC UA server.

**Note** Make sure that the OPC UA server is accessible from your engineering station before establishing an online connection.

3. Drag and drop the variables of the server interface to be read into the read list.



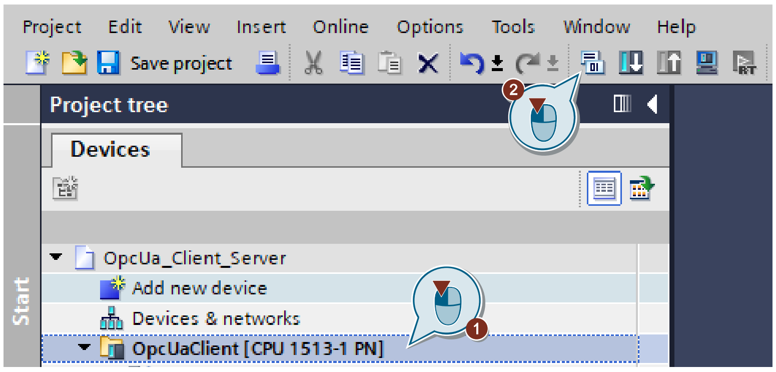
readList

| | Name of the node | Node type | Access level | Node ID |
|---|---|---|---|---|
| 1 | startCounting | BOOL | RD/WR | http://ww... |
| 2 | actValue | DINT | RD | http://ww... |
| 3 | endValue | DINT | RD/WR | http://ww... |
| 4 | Insert a new... | | | |

OPC UA server interface

| | Name of the node |
|---|---|
| 1 | Objects |
| 2 | Server |
| 3 | DeviceSet |
| 4 | OpcUaServer |
| 5 | Counters |
| 6 | DataBlocksGlobal |
| 7 | Icon |
| 8 | OpcUaInterfac |
| 9 | startCounting |
| 10 | actValue |
| 11 | endValue |
| 12 | DataBlocksInstance |

4. Repeat steps 1 to 3 for Write lists and Method lists.
5. Mark your CPU in the project navigation and click on the symbol "Compile", so that TIA Portal generates the data blocks "Configuration" and "Data". These two data blocks now contain all OPC UA transfer parameters for the S7 user block "OpcUaClient".



After each change in the OPC UA-Client interface you must recompile the CPU so that the parameters are transferred into the data blocks.

## 2.2.3 Parameterizing the OPC UA Client Module

First create a client interface as described in section 2.2.2 Creating an OPC UA Client Interface.

### Copy block to your project

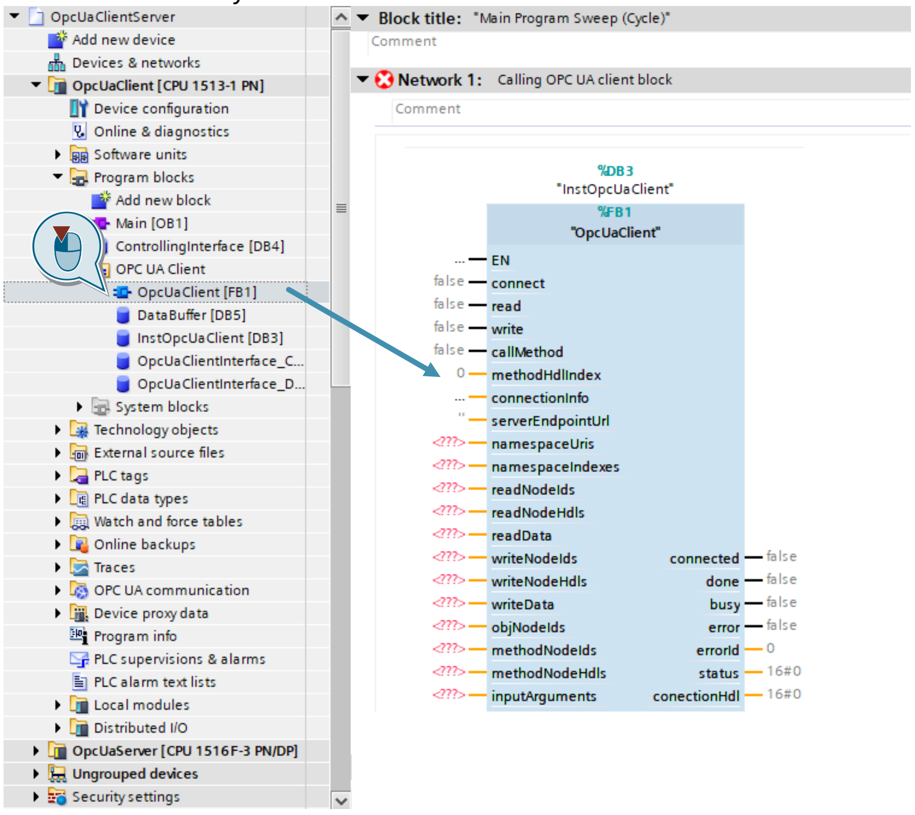
Copy the OPC UA client block "OpcUaClient" into your project as follows:

1. Load the example project "109762770_OPC_UA_PLC-Client_ CODE_en.zip" and unzip the ZIP archive.
2. Open the project file "OpcUaClientServer.ap15_1".
3. Navigate in the project tree to "OpcUaClient > Program blocks > OPC UA Client" ("OpcUaClient > Program blocks > OPC UA Client") and copy the block "OpcUaClient" into your own project.

### Parameterize block

Parameterize the interface of the block "OpcUaClient" in your user program as follows:

1. Call the block in a cyclic OB and create a new instance.



```
OpcUaClientServer
  Add new device
  Devices & networks
  OpcUaClient [CPU 1513-1 PN]
    Device configuration
    Online & diagnostics
    Software units
    Program blocks
      Add new block
      Main [OB1]
      ControllingInterface [DB4]
      OPC UA Client
        OpcUaClient [FB1]
        DataBuffer [DB5]
        InstOpcUaClient [DB3]
        OpcUaClientInterface_C...
        OpcUaClientInterface_D...
      System blocks
    Technology objects
    External source files
    PLC tags
    PLC data types
    Watch and force tables
    Online backups
    Traces
    OPC UA communication
    Device proxy data
    Program info
    PLC supervisions & alarms
    PLC alarm text lists
    Local modules
    Distributed I/O
  OpcUaServer [CPU 1516F-3 PN/DP]
  Ungrouped devices
  Security settings
```

```
Block title: "Main Program Sweep (Cycle)"
Comment
Network 1:  Calling OPC UA client block
Comment

                  %DB3
           "InstOpcUaClient"
                  %FB1
             "OpcUaClient"
...    — EN
false  — connect
false  — read
false  — write
false  — callMethod
0      — methodHdlIndex
...    — connectionInfo
''     — serverEndpointUrl
<???>  — namespaceUris
<???>  — namespaceIndexes
<???>  — readNodeIds
<???>  — readNodeHdls
<???>  — readData
<???>  — writeNodeIds          connected — false
<???>  — writeNodeHdls              done — false
<???>  — writeData                  busy — false
<???>  — objNodeIds                error — false
<???>  — methodNodeIds           errorId — 0
<???>  — methodNodeHdls           status — 16#0
<???>  — inputArguments     conectionHdl — 16#0
```

2. Connect the parameters for control and diagnosis of the block. Create suitable variables for this.

| Block parameters | Data type |
|---|---|
| connect | Bool |
| read | Bool |
| write | Bool |
| callMethod | Bool |
| connected | Bool |
| done | Bool |
| busy | Bool |
| error | Bool |
| status | DWord |
| diagnostics | "typeFbDiagnostics" |

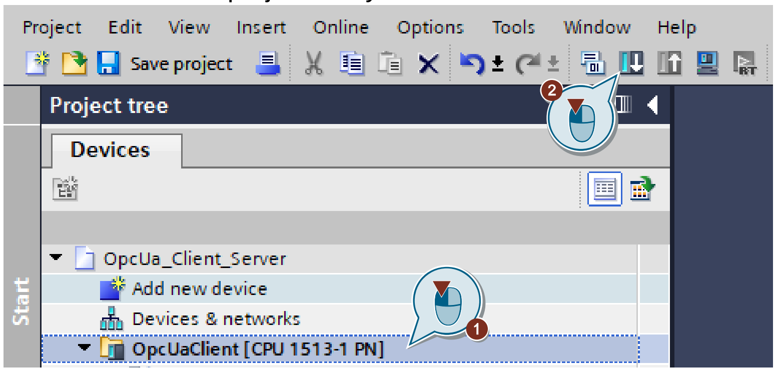
3. Connect the client interface parameters of the "OpcUaClient" block. The variables to be interconnected can be found in the data blocks that are generated via the client interface.
   Assign "NULL" to unneeded blocks of the following table. The parameters of the "Connection" block are mandatory.

| Block parameters | Data module and data module parameters |
|---|---|
| **Connection** | |
| connectionInfo | "<InterfaceName>_Configuration".Connection.ConnectInfo |
| serverEndpointUrl | "<InterfaceName>_Configuration".Connection. serverEndpointUrl |
| namespaceUris | "<InterfaceName>_Configuration".Namespaces. NamespaceUris |
| namespaceIndexes | "<InterfaceName>_Configuration".Namespaces. ServerNameSpaceIndexes |
| connectionHdl | (Optional) "<InterfaceName>_Configuration".Connection. ConenctionHdl |
| **Read** | |
| readNodeIds | "<InterfaceName>_Configuration".ReadLists. <ReadListName>.Nodes |
| readNodeHdls | "<InterfaceName>_Configuration".ReadLists. <ReadListName>.NodeHdls |
| readData | "<InterfaceName>_Data".<ReadListName>.Variable |
| **Write** | |
| writeNodeIds | "<InterfaceName>_Configuration".WriteLists. <WriteListName>.Nodes |
| writeNodeHdls | "<InterfaceName>_Configuration".WriteLists. <WriteListName>.NodeHdls |
| writeData | "<InterfaceName>_Data".<WriteListName>.Variable |
| **CallMethod** | |
| objNodeIds | "<InterfaceName>_Configuration".MethodLists. <MethodListName>.ObjectNodes |
| methodNodeIds | "<InterfaceName>_Configuration".MethodLists. <MethodListName>.MethodNodes |

| Block parameters | Data module and data module parameters |
|---|---|
| methodNodeHdls | "<InterfaceName>_Configuration".MethodLists.<MethodListName>.MethodHdls |
| inputArguments | "<InterfaceName>_Data".<MethodListName>.<MethodName>.Inputs |
| outputArguments | "<InterfaceName>_Data".<MethodListName>.<MethodName>.Outputs |

**Note** If the OPC UA method to be called does not contain "inputArguments" or "outputArguments", set the parameters to "NULL".

4. Mark your CPU in the project navigation and click on the icon "Download to device" to load the project into your CPU.



## 2.3 Commissioning and operation of the example

In this application example, the operation is carried out via an observation table from TIA Portal.

In section 3.2 OPC UA server in the example you will find an overview of the functions that you can control on the OPC UA Server with your OPC UA Client.

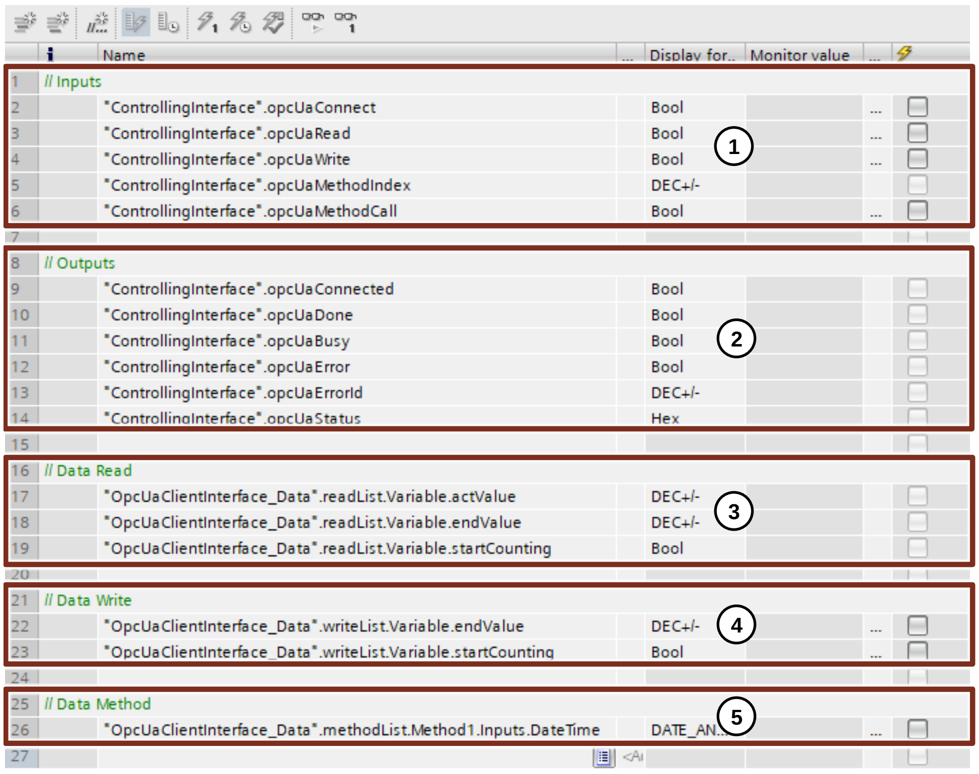
### 2.3.1 Commissioning the example

Proceed as follows to put the example into operation:

1. Download the example project "109762770_OPC_UA_PLC-Client_ 15_PROJ_V1_2.zip" or "109762770_OPC_UA_PLC-Client_ 16_PROJ_V1_2.zip" and unzip the ZIP archive.
2. Open the project file "OpcUaClientServer.ap15_1" or "OpcUaClientServer.ap16".
3. (Optional) Exchange your CPUs in the project if you use other SIMATIC S7-1500 controllers. Click in the project tree with the right mouse button on the CPUs and then in the context menu on "Exchange device...". ("Change device...").
4. Load the two configured CPUs ("OpcUaClient" and "OpcUaServer") into your controllers.

### 2.3.2 Description of the user interface

The following figure shows the observation table "ControllingOpcUa" of the example:



| | Name | Display for.. | Monitor value |
|---|---|---|---|
| 1 | // Inputs | | |
| 2 | "ControllingInterface".opcUaConnect | Bool | |
| 3 | "ControllingInterface".opcUaRead | Bool | |
| 4 | "ControllingInterface".opcUaWrite | Bool | |
| 5 | "ControllingInterface".opcUaMethodIndex | DEC+/- | |
| 6 | "ControllingInterface".opcUaMethodCall | Bool | |
| 7 | | | |
| 8 | // Outputs | | |
| 9 | "ControllingInterface".opcUaConnected | Bool | |
| 10 | "ControllingInterface".opcUaDone | Bool | |
| 11 | "ControllingInterface".opcUaBusy | Bool | |
| 12 | "ControllingInterface".opcUaError | Bool | |
| 13 | "ControllingInterface".opcUaErrorId | DEC+/- | |
| 14 | "ControllingInterface".opcUaStatus | Hex | |
| 15 | | | |
| 16 | // Data Read | | |
| 17 | "OpcUaClientInterface_Data".readList.Variable.actValue | DEC+/- | |
| 18 | "OpcUaClientInterface_Data".readList.Variable.endValue | DEC+/- | |
| 19 | "OpcUaClientInterface_Data".readList.Variable.startCounting | Bool | |
| 20 | | | |
| 21 | // Data Write | | |
| 22 | "OpcUaClientInterface_Data".writeList.Variable.endValue | DEC+/- | |
| 23 | "OpcUaClientInterface_Data".writeList.Variable.startCounting | Bool | |
| 24 | | | |
| 25 | // Data Method | | |
| 26 | "OpcUaClientInterface_Data".methodList.Method1.Inputs.DateTime | DATE_AN... | |
| 27 | | <A | |

Figure 2-11

The following table explains the subdivision of the observation table

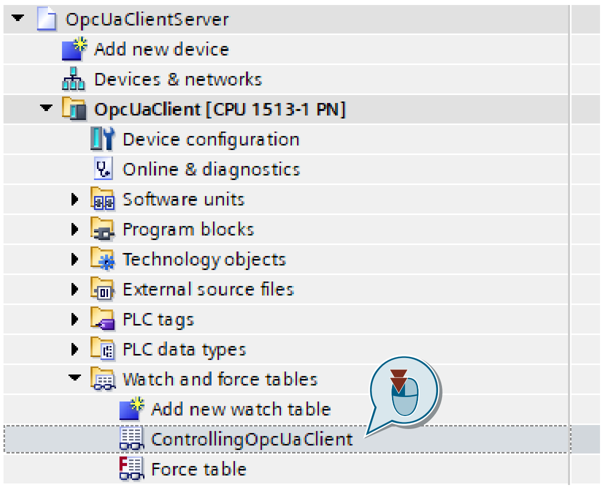
Table 2-6

| No. | Description |
|---|---|
| 1. | The "Inputs" area contains the variables for controlling the block. |
| 2. | The "Outputs" area contains the variables for diagnosis of the block. |
| 3. | The "Data Read" area contains the variables that are read by the OPC UA server. |
| 4. | The "Data Write" area contains the variables that are written to the OPC UA server. |
| 5. | The "Data Method" section contains the "InputArgument" for the method that is executed on the OPC UA server. |

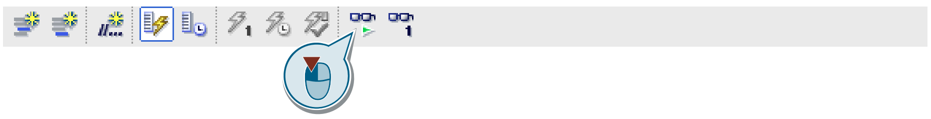
## 2.3.3 Operation of the example

Proceed as follows to operate the example:

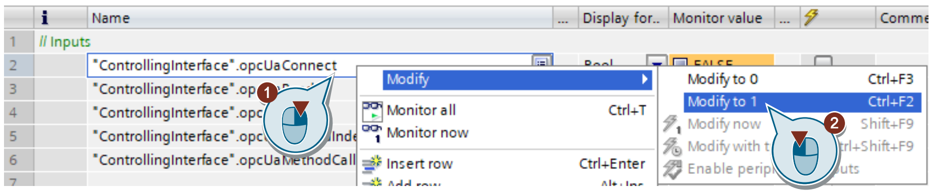
1. Navigate in the project tree to "OpcUaClient > Watch and force tables" and open the observation table "ControllingOpcUaClient" with a double click.



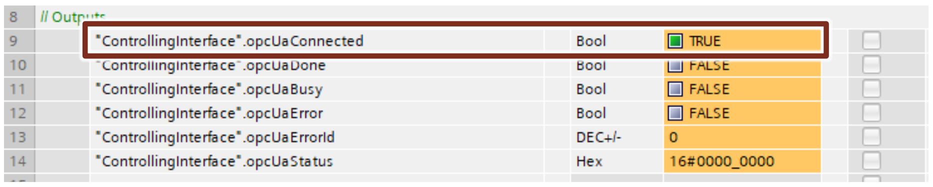
2. Click on the "Monitor all" icon in the workspace to observe the variables of the controller live.



3. First connect your client to the server by setting the variable "opcUaConnect" in the "Inputs" area to "TRUE". To do this, right-click the variable and select Tax > Tax at 1 ("Modify > Modify to 1").



4. Check the connection setup. Observe the variable "opcUaConnected" in the "Outputs" area:
"TRUE" - Client is connected.
"FALSE" client is not connected.



| | | | |
|---|---|---|---|
| 8 | // Outputs | | |
| 9 | "ControllingInterface".opcUaConnected | Bool | 🟩 TRUE |
| 10 | "ControllingInterface".opcUaDone | Bool | ▨ FALSE |
| 11 | "ControllingInterface".opcUaBusy | Bool | ▨ FALSE |
| 12 | "ControllingInterface".opcUaError | Bool | ▨ FALSE |
| 13 | "ControllingInterface".opcUaErrorId | DEC+/- | 0 |
| 14 | "ControllingInterface".opcUaStatus | Hex | 16#0000_0000 |

Further information about the diagnosis of the block can be found in section **2.1.4** Device diagnostics.

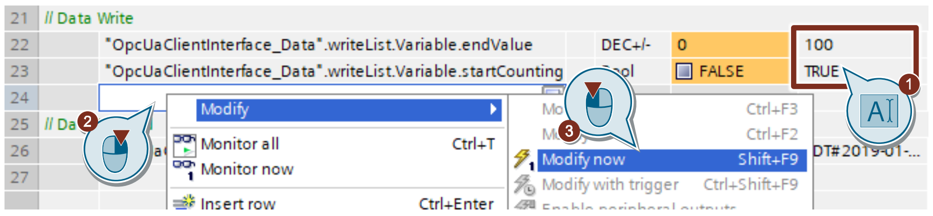
After you have connected your client to the OPC UA server, you can perform the following operations:

- Read variables from server
- Writing Variables to the Server
- Execute method

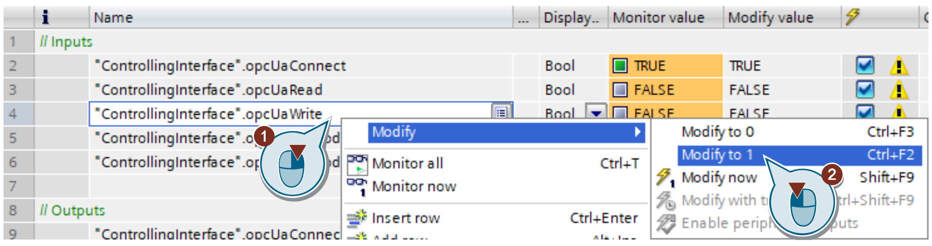
## Writing server variables

Proceed as follows to write the server's variables:

1. Control the variables in the "Data Write" area. Enter the desired values to be written in the "Modify value" column. The variable "endValue" determines how high the counter in the OPC UA server is to count (in the example: "100"). The variable "startCounting" determines whether the counter of the server is active or not (in the example: "TRUE").
Right-click in the workspace and select Control > Modify > Modify now to transfer the values to the controller.

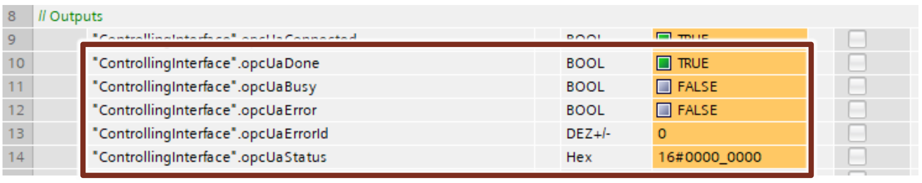

21 // Data Write
22 "OpcUaClientInterface_Data".writeList.Variable.endValue DEC+/- 0 100
23 "OpcUaClientInterface_Data".writeList.Variable.startCounting FALSE TRUE
DT#2019-01-...
1
2
3
Modify
Monitor all Ctrl+T
Monitor now
Insert row Ctrl+Enter
Ctrl+F3
Ctrl+F2
Modify now Shift+F9
Modify with trigger Ctrl+Shift+F9

2. Write the OPC UA variables of the server by setting the variable "OpcUaWrite" in the "Inputs" area to "TRUE". To do this, right-click the variable and select Tax > Tax at 1 ("Modify > Modify to 1").



| | i | Name | ... | Display.. | Monitor value | Modify value | ⚡ |
|---|---|---|---|---|---|---|---|
| 1 | | // Inputs | | | | | |
| 2 | | "ControllingInterface".opcUaConnect | | Bool | 🟩 TRUE | TRUE | ✓ ⚠ |
| 3 | | "ControllingInterface".opcUaRead | | Bool | ▨ FALSE | FALSE | ✓ ⚠ |
| 4 | | "ControllingInterface".opcUaWrite | | Bool | ▨ FALSE | FALSE | ✓ ⚠ |
| 5 | | "ControllingInterface".o | | | | | |
| 6 | | "ControllingInterface".op | | | | | |
| 7 | | | | | | | |
| 8 | | // Outputs | | | | | |
| 9 | | "ControllingInterface".opcUaConnec | | | | | |

3. Check the write job. Observe the variables "opcUaDone", "opcUaError", "opcUaErrorId" and "opcUaStatus" in the "Outputs" area:



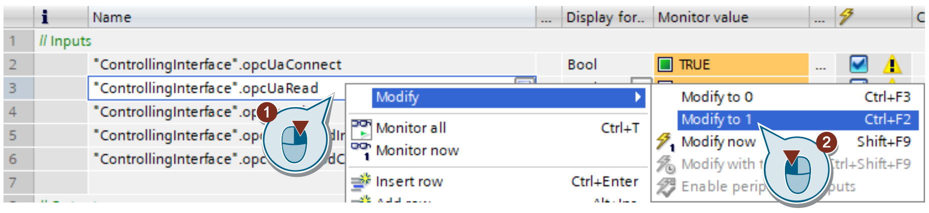
| | | | | |
|---|---|---|---|---|
| 8 | // Outputs | | | |
| 9 | "ControllingInterface".opcUaConnected | BOOL | 🟩 TRUE | |
| 10 | "ControllingInterface".opcUaDone | BOOL | 🟩 TRUE | ☐ |
| 11 | "ControllingInterface".opcUaBusy | BOOL | ▨ FALSE | ☐ |
| 12 | "ControllingInterface".opcUaError | BOOL | ▨ FALSE | ☐ |
| 13 | "ControllingInterface".opcUaErrorId | DEZ+/- | 0 | ☐ |
| 14 | "ControllingInterface".opcUaStatus | Hex | 16#0000_0000 | ☐ |

Further information about the diagnosis of the block can be found in section 2.1.4 Device diagnostics.

## Read server variables

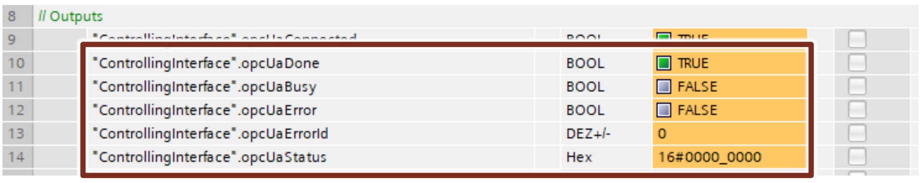
Proceed as follows to read the variables of the server:

1. Read the OPC UA variables of the server by setting the variable "OpcUaRead" in the "Inputs" area to "TRUE". To do this, right-click the variable and select Tax > Tax at 1 ("Modify > Modify to 1").



| | i | Name | ... | Display for.. | Monitor value | ... | ⚡ |
|---|---|---|---|---|---|---|---|
| 1 | | // Inputs | | | | | |
| 2 | | "ControllingInterface".opcUaConnect | | Bool | 🟩 TRUE | ... | ✓ ⚠ |
| 3 | | "ControllingInterface".opcUaRead | | | | | |
| 4 | | "ControllingInterface".op | | | | | |
| 5 | | "ControllingInterface".op | | | | | |
| 6 | | "ControllingInterface".opc | | | | | |
| 7 | | | | | | | |

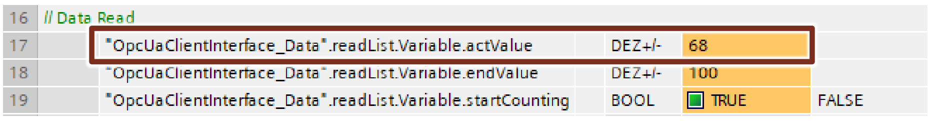
2. Check the read request. Observe the variables "opcUaDone", "opcUaError", "opcUaErrorId" and "opcUaStatus" in the "Outputs" area:



| | | | | |
|---|---|---|---|---|
| 8 | // Outputs | | | |
| 9 | "ControllingInterface".opcUaConnected | BOOL | 🟩 TRUE | |
| 10 | "ControllingInterface".opcUaDone | BOOL | 🟩 TRUE | ☐ |
| 11 | "ControllingInterface".opcUaBusy | BOOL | ▨ FALSE | ☐ |
| 12 | "ControllingInterface".opcUaError | BOOL | ▨ FALSE | ☐ |
| 13 | "ControllingInterface".opcUaErrorId | DEZ+/- | 0 | ☐ |
| 14 | "ControllingInterface".opcUaStatus | Hex | 16#0000_0000 | ☐ |

Further information about the diagnosis of the block can be found in section 2.1.4 Device diagnostics.

3. Observe the values in the "Data Read" area. If you have previously specified a value for "endValue" and set "startCounting", then the value "actValue" changes every second. Check the value change with further read requests.

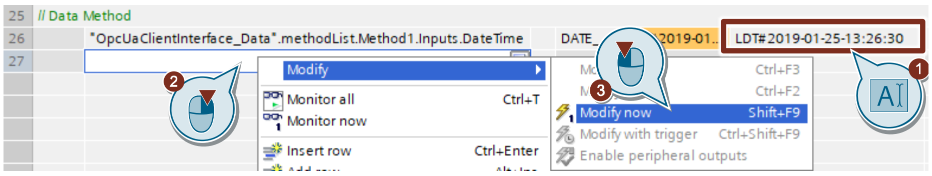


| | | | | |
|---|---|---|---|---|
| 16 | // Data Read | | | |
| 17 | "OpcUaClientInterface_Data".readList.Variable.actValue | DEZ+/- | 68 | |
| 18 | "OpcUaClientInterface_Data".readList.Variable.endValue | DEZ+/- | 100 | |
| 19 | "OpcUaClientInterface_Data".readList.Variable.startCounting | BOOL | 🟩 TRUE | FALSE |

## Call method of server

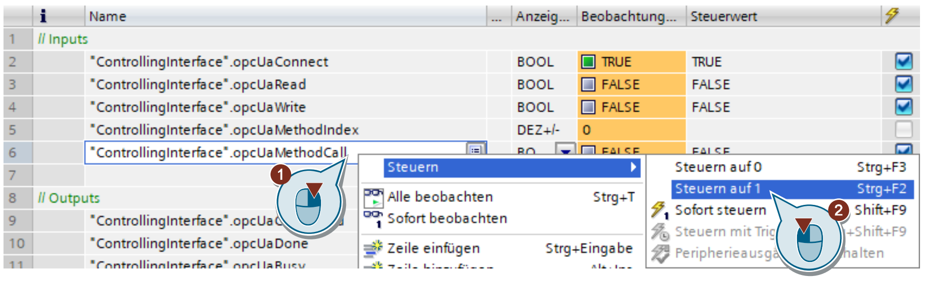
To call the server's OPC UA method, follow these steps:

1. Control the variable "dataTime" in the "Data Method" area. Enter the PLC time to be transferred in the "Modify value" column. Right-click in the workspace and select Control > Modify > Modify now to transfer the value to the controller.



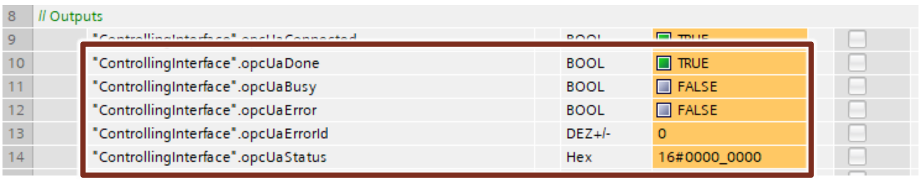
| | | | | |
|---|---|---|---|---|
| 25 | // Data Method | | | |
| 26 | "OpcUaClientInterface_Data".methodList.Method1.Inputs.DateTime | DATE_ | 2019-01.. | LDT#2019-01-25-13:26:30 |
| 27 | | | | |

2. Call the server's OPC UA method by setting the OpcUaMethodCall variable in the Inputs section to TRUE. To do this, right-click the variable and select Tax > Tax at 1 ("Modify > Modify to 1").



| | i | Name | ... | Anzeig... | Beobachtung... | Steuerwert | ⚡ |
|---|---|---|---|---|---|---|---|
| 1 | // Inputs | | | | | | |
| 2 | | "ControllingInterface".opcUaConnect | | BOOL | 🟩 TRUE | TRUE | ☑ |
| 3 | | "ControllingInterface".opcUaRead | | BOOL | ▢ FALSE | FALSE | ☑ |
| 4 | | "ControllingInterface".opcUaWrite | | BOOL | ▢ FALSE | FALSE | ☑ |
| 5 | | "ControllingInterface".opcUaMethodIndex | | DEZ+/- | 0 | | ☐ |
| 6 | | "ControllingInterface".opcUaMethodCall | | BO | ▢ FALSE | FALSE | ☑ |
| 7 | | | | | | | |
| 8 | // Outputs | | | | | | |
| 9 | | "ControllingInterface".opcUaC | | | | | |
| 10 | | "ControllingInterface".opcUaDone | | | | | |
| 11 | | "ControllingInterface".opcUaBusy | | | | | |

3. Check the method call. Observe the variables "opcUaDone", "opcUaError", "opcUaErrorId" and "opcUaStatus" in the "Outputs" area:



| | | | | |
|---|---|---|---|---|
| 8 | // Outputs | | | |
| 9 | "ControllingInterface".opcUaConnected | BOOL | 🟩 TRUE | ☐ |
| 10 | "ControllingInterface".opcUaDone | BOOL | 🟩 TRUE | ☐ |
| 11 | "ControllingInterface".opcUaBusy | BOOL | ▢ FALSE | ☐ |
| 12 | "ControllingInterface".opcUaError | BOOL | ▢ FALSE | ☐ |
| 13 | "ControllingInterface".opcUaErrorId | DEZ+/- | 0 | ☐ |
| 14 | "ControllingInterface".opcUaStatus | Hex | 16#0000_0000 | ☐ |

Further information about the diagnosis of the block can be found in section 2.1.4 Device diagnostics.

4. Check the new PLC time on the display of your server CPU.

# 3 Useful information

## 3.1 Connection monitoring in the "OpcUaClient" block

### Connection monitoring for S7-1500 FW 2.6

The following diagram describes the error handling of the implemented connection monitoring by the system function block "ConnectionGetStatus" (CGS):

Figure 3-1

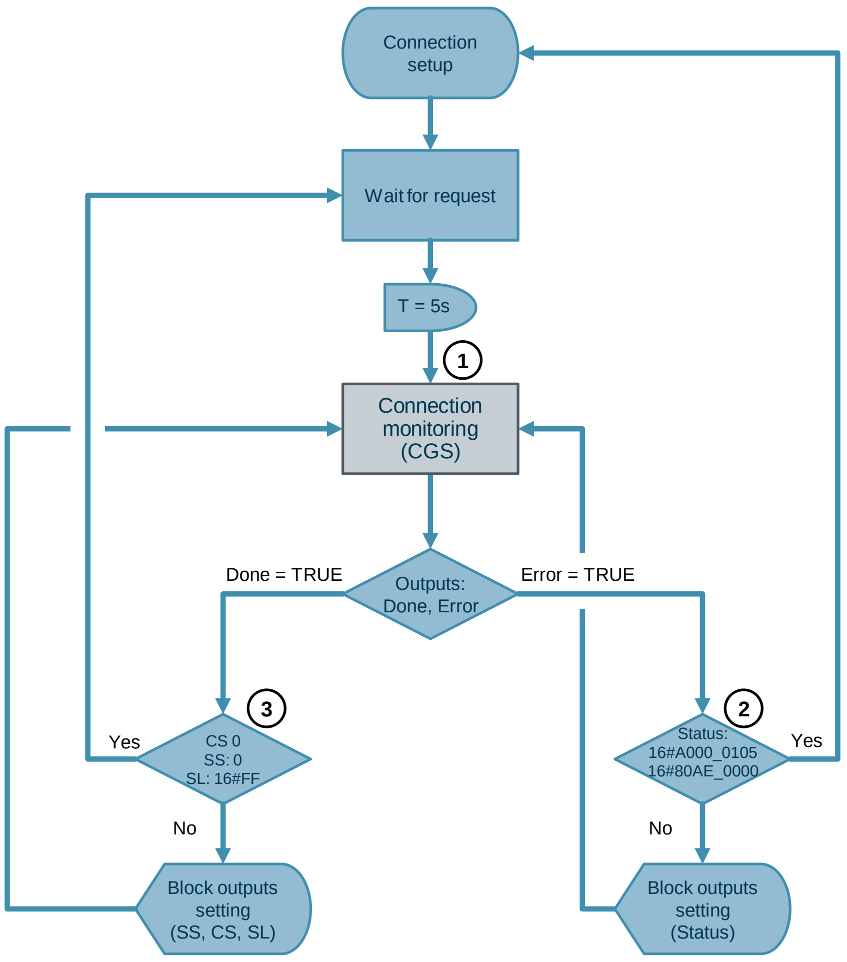

Connection setup
Wait for request
T = 5s
1
Connection monitoring (CGS)
Done = TRUE
Outputs: Done, Error
Error = TRUE
3
Yes
CS 0
SS: 0
SL: 16#FF
No
Block outputs setting (SS, CS, SL)
2
Status: 16#A000_0105 16#80AE_0000
Yes
No
Block outputs setting (Status)

Table 3-1

| No. | Description |
|---|---|
| 1. | After a connection has been successfully established from the "OpcUaClient" block to a server, the system function block "OPC_UA_ConnectionGetStatus" is called periodically (5 seconds) to monitor the connection to the server. |
| 2. | If the "Connection Handle" to the server is invalid, you get an error at the SFB "OPC_UA_ConnectionGetStatus" ("Error" = "TRUE"). A new connection is established for the associated "Status" = "16#A000_0105" or "16#80AE_0000". Other values indicate errors that must be corrected by the user before a new connection can be established. In this case, the outputs of the block "OpcUaClient" are set and the SFB is executed again. For more information on the outputs, see the TIA online help (F1). |

| No. | Description |
|---|---|
| 3. | If a valid "Connection Handle" to the server exists, the SFB "OPC_UA_ConnectionGetStatus" is successfully executed ("Done" = "TRUE"). In this case the block "OpcUaClient" evaluates the outputs "ConnectionStatus" (CS), "ServerState" (SS) and "ServiceLevel" (SL). The values CS = "0", SS = "0" and SL = "16#FF" indicate a correct connection, the block "OpcUaClient" is ready to execute further jobs. Other values indicate errors that must be corrected by the user before a new connection can be established. In this case, the outputs of the block "OpcUaClient" are set and the SFB is executed again. For more information on the outputs, see the TIA online help (F1). |

## Connection monitoring for S7-1500 FW 2.8 or newer

The following diagram describes the error handling of the implemented connection monitoring by the system function block "OPC_UA_ConnectionGetStatus" (CGS):

Figure 3-2

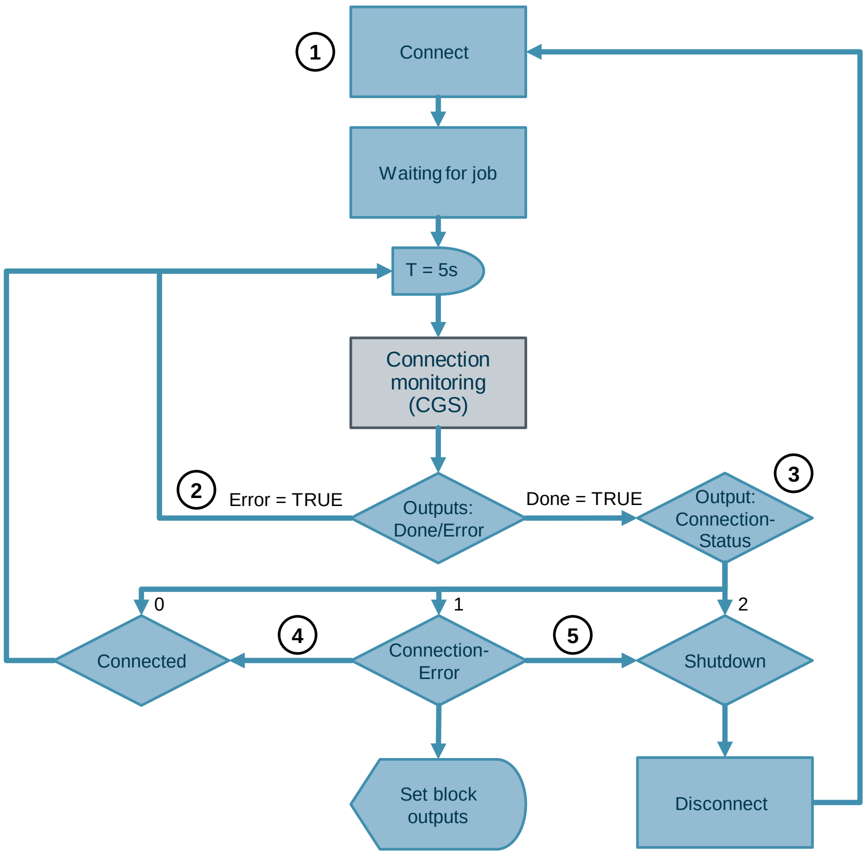


Table 3-2

| No. | Description |
|---|---|
| 1. | After a connection from the block "OpcUaClient" to a server has been established successfully, the system function block "OPC_UA_ConnectionGetStatus" is executed periodically (5 seconds) to monitor the connection to the server. |
| 2. | If the block "OPC_UA_ConnectionGetStatus" returns an error ("Error" = TRUE) during connection monitoring, it is to be assumed that a underlaying read job of the nodes "ServerState" and "ServiceLevel" is currently taking place. In this case the block must be executed again. |
| 3. | If the connection monitoring via the block "OPC_UA_ConnectionGetStatus" is successful ("Done" = TRUE), the block output "ConnectionStatus" must be evaluated:<br>"0": Connection established ("UACS_Connected")<br>"1": Connection interrupted ("UACS_ConnectionError")<br>"2": Disconnected ("UACS_Shutdown") |
| 4. | When the cause of a connection error is solved ("UACS_ConnectionError" > "UACS_Connected") the OPC UA client of a SIMATIC S7-1500 reactivates the connection automatically. |
| 5. | When the cause of a connection error is solved but the session on the server has already been deactivated ("UACS_ConnectionError" > "UACS_Shutdown") the connection must be terminated (SFB "OPC_UA_Disconnect") and then re-established (SFB "OPC_UA_Connect"). |

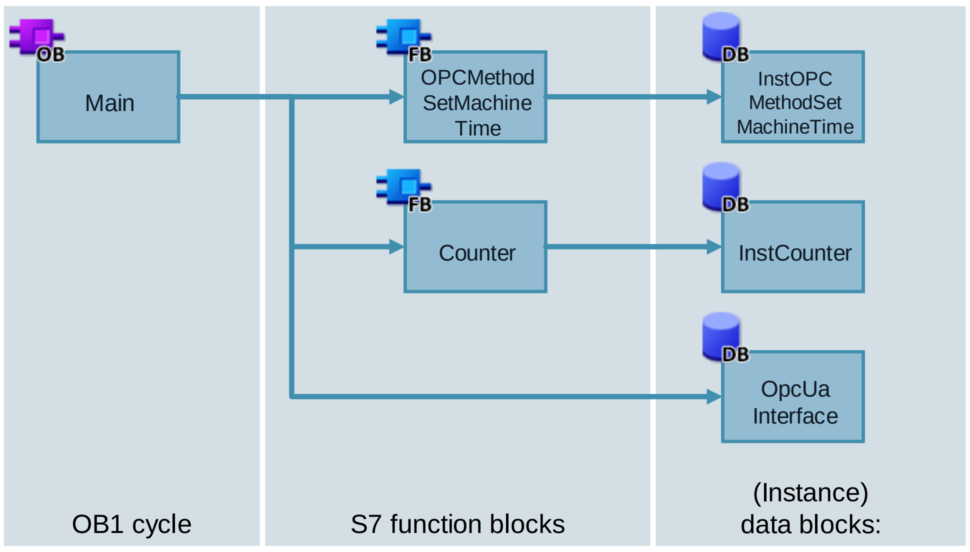
## 3.2 OPC UA server in the example

The S7 user program of the second CPU "OpcUaServer" in the example project contains the OB1, two user blocks and a data block. The CPU serves only as an example server.

### Call structure

The following figure shows the call hierarchy of the S7 user program of the CPU "OpcUaServer":

Figure 3-3



### Explanation of the blocks

In the cyclic user program, only the function blocks "Counter" and "OPCMethodSetMachineTime" are called.

The block "Counter" increments the variable "actValue" of the data block "OpcUaInterface" every second, if the variable "startCounting" is also set. You can use the variable "endValue" to determine how far "Counter" can count. When "endValue" is reached, the counter starts at "0" again.

The block "OPCMethodSetMachineTime" implements an OPC UA method that sets the PLC time.

## 3.3 Quantity Structure and Licenses of the OPC UA Client

The following table explains the quantity structure and the licenses of the OPC UA client of the SIMATIC S7-1500:

Table 3-3

| **Runtime license/** CPU type | **Small** ET 200SP CPU CPU 1511 CPU 1513 | **Medium** CPU 1515 CPU 1516 CPU 1507S | **Large** CPU 1517 CPU1518 CPU 1508S |
|---|---|---|---|
| Max. Number of connections: | 4 | 10 | 40 |
| Max. Elements in a read/write/method list: | 300 | 300 | 300 |
| Max. Nodes in a client interface: | 1000 | 2000 | 5000 |
| Max. parallel read/write/method jobs: | 5 | 5 | 5 |
| Max. Number of methods Handles: | 100 | 100 | 100 |
| Max. Number of input/output arguments: | 20 | 20 | 20 |

**Note** Despite the "Medium" license, the SIMATIC S7-1500 CPU 1507S offers the "Large" quantity structure.

## 3.4 Tips & tricks

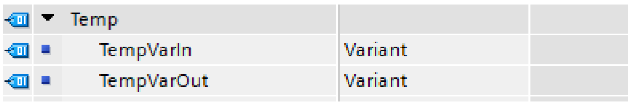
### Set PLC time

With certificate-based authentication, the validity date of the certificates is checked. For this it is necessary to set the PLC time correctly. We recommend that you use an NTP server for this purpose.

### Change Input and Output Arguments for Methods at Runtime

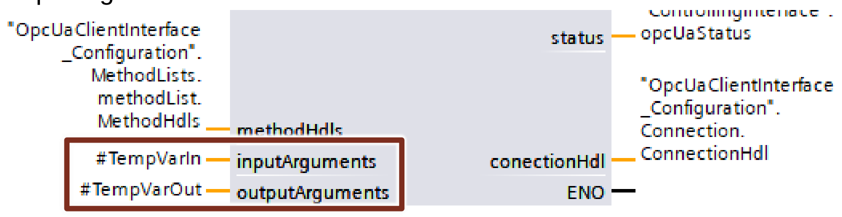
Since the block "OpcUaClient" provides only one input each for the input and output arguments, you must assign the inputs at runtime if you want to call several different methods. Proceed as follows:

1. Create a temporary variable of type "Variant" in the block interface of the block for the input and output arguments:



| Name | Data type |
|---|---|
| Temp | |
| TempVarIn | Variant |
| TempVarOut | Variant |

2. Assign the temporary variables to the InOut parameters "inputArguments" and "outputArguments" of the user block:


"OpcUaClientInterface_Configuration".MethodLists.methodList.MethodHdls
methodHdls
#TempVarIn
inputArguments
#TempVarOut
outputArguments
status
opcUaStatus
"OpcUaClientInterface_Configuration".Connection.ConnectionHdl
conectionHdl
ENO

3. Reference the temporary variables to the desired input or output argument using the "REF" statement:

```
#TempVarIn :=
REF("OpcUaClientInterface_Data".methodList.Method1.Inputs);
#TempVarOut :=
REF("OpcUaClientInterface_Data".methodList.Method1.Outputs);
```

4. Then, depending on the selected method, assign an input and output argument to the temporary variables using the "VariantPut" statement:

```
IF methodHdlIndex = 1 THEN
    VariantPut(SRC:="OpcUaClientInterface_Data".methodList.
    Method1.Inputs, DST:=#TempVarIn);
    VariantPut(SRC:="OpcUaClientInterface_Data".methodList.
    Method1.Outputs, DST:=#TempVarOut);
END_IF;


IF methodHdlIndex = 2 THEN
    VariantPut(SRC:="OpcUaClientInterface_Data".methodList.
    Method2.Inputs, DST:=#TempVarIn);
    VariantPut(SRC:="OpcUaClientInterface_Data".methodList.
    Method2.Outputs, DST:=#TempVarOut);
END_IF;
```

This procedure enables you to change the assignment of the module interfaces at runtime. Follow the procedure above to assign the appropriate input and output arguments to the device, depending on the method selected (input parameter: "methodHdlIndex"). You will find the arguments in the following memory area of the data block of the client interface:

- "<InterfaceName>_Data".methodList.<MethodName>.Inputs
- "<InterfaceName>_Data".methodList.<MethodName>.Outputs

# 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:

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

### 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\ | Siemens Industry Online Support<br>https://support.industry.siemens.com |
| \2\ | Link to the entry page of this application example<br>https://support.industry.siemens.com/cs/ww/en/view/109762770 |

## 4.3 Change documentation

Table 4-2

| Version | Date | Modifications |
|---|---|---|
| V1.0 | 03/2019 | First version |
| V1.1 | 10/2019 | Bug fixing; Extended error handling; Update for TIA Portal V16 and SIMATIC S7-1500 FW 2.8 |
| V1.2 | 10/2020 | Bug fix for the TIA Portal V16 project |
| V1.2.1 | 08/2021 | Bug fix for the TIA Portal V16 project |