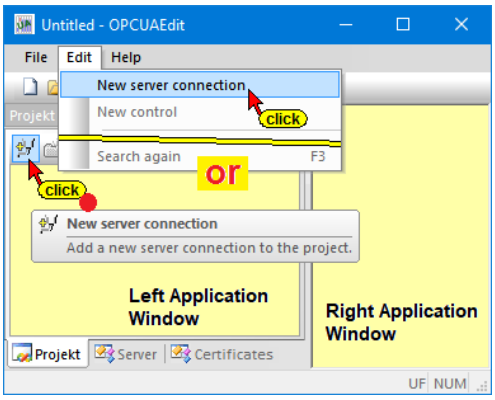
1 IBH OPC Editor: Modbus connection

With the IBH OPC Editor (version 7.3.4 or higher), a Modbus configuration can be transferred to the IBH Link UA (firmware V4.12 or higher).

1.1 New Server Connection

The ***New Server Connection command*** from the ***Edit*** menu or clicking the icon opens the ***New Server Connection*** dialog box.



1.1.1 Server Connection

To establish a connection to an OPC UA server, the connection data must be specified. The ***New server connection*** dialog box makes it easier to specify the connection data.

Note:

The connection data specified by the ***New server connection*** dialog box will be displayed in the right part of the Project window after completion.

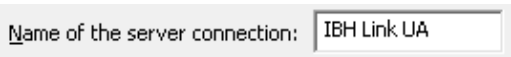
This connection data can be changed at any time in the right part of the project window. If a selection is possible, the changes can be made via drop-down list boxes.

New server connection dialog box

The fields for the general settings for the connection to an OPC UA server must be filled out.

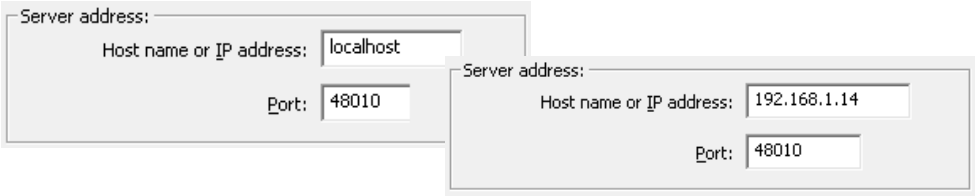
Name of the server connection

The name is freely selectable.

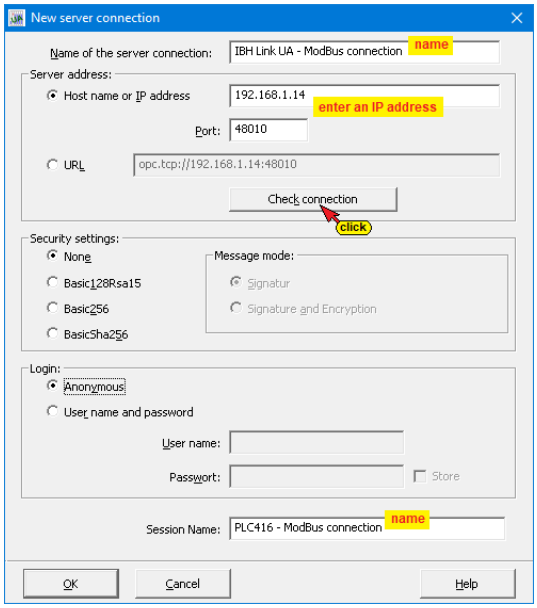


Server address

If the ***IBH Link UA port*** to be used for connection is in a network with ***a DHCP server***, replace ***localhost*** with the actual ***host name***.

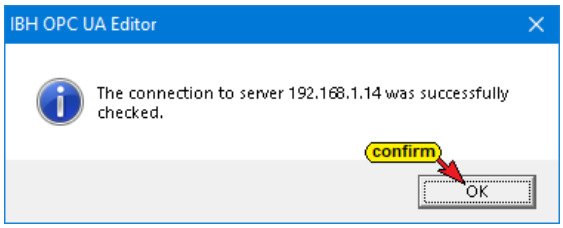


If there is ***no DHCP server***, replace ***localhost*** with the IBH Link UA Port IP address (192.168.1.14).



Check connection

After completing the ***New Server Connection*** dialog box, the connection to the online connected IBH Link UA can be tested. Information about the successful connection is displayed.



Security settings

In this field the security procedure and the message mode can be selected.

If a security procedure is selected, certificates must be exchanged between the ***IBH OPC Editor*** and the OPC UA Server (IBH Link UA). The suggested value is ***None***.

As a safety procedure, you can set ***None***, Basic128Rsa15, ***Basic256*** or ***BasicSha256***.

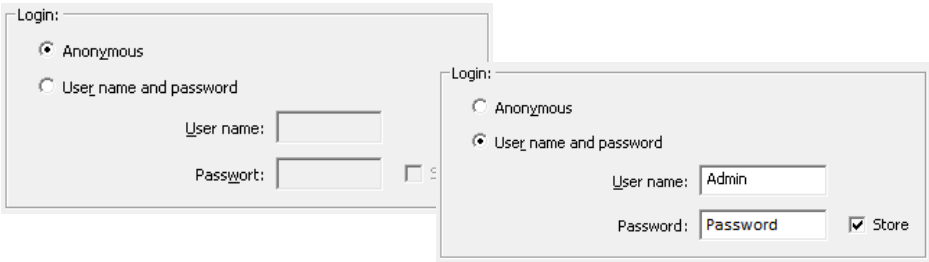
Message mode

If a security procedure is used, Signature as well as Signature and Encryption are available.

Otherwise without.

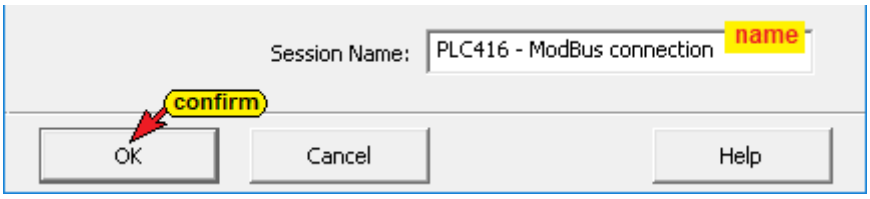
Login

In this field, the user name can be specified with the associated password. The default login mode is ***Anonymous***.



Session name

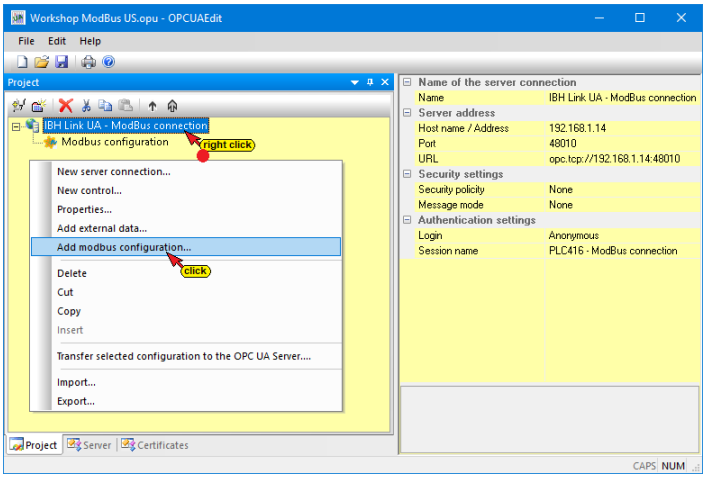
The name of the session is freely selectable and can be left blank.



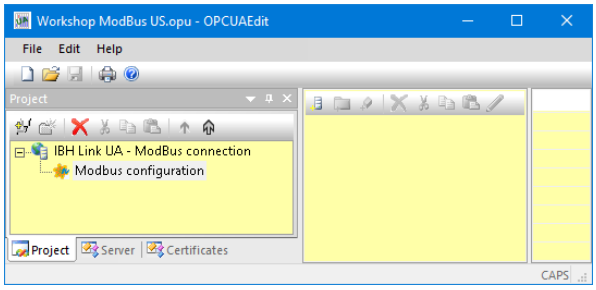
1.2 Add Modbus configuration

The right part of the project window displays the name of the OPC UA server with host name / IP address.

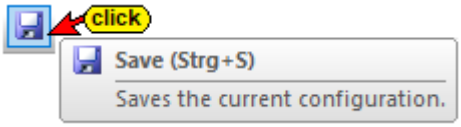
Right-clicking on the name of the server connection opens the context menu. The ***Add Modbus Configuration*** command starts the configuration process.



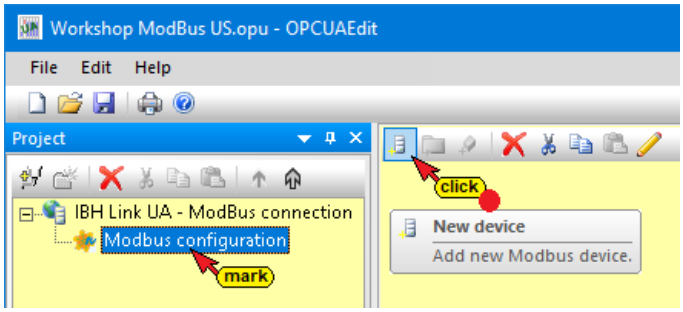
In the right part of the project window, the ***Modbus configuration*** window opens.



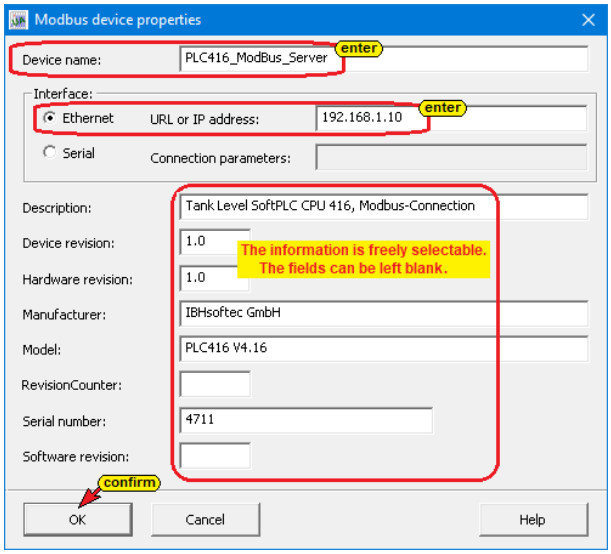
The existing configuration can be saved at any time.



1.2.1 Add new Modbus device



If Modbus configuration is marked in the left part of the project window, the Modbus device properties dialog box can be opened by clicking on the New device icon.



If the device name and the interface are specified, the dialog box can be confirmed with ***OK***.

1.2.2 Modbus TCP interface / Modbus variable

The Modbus organization specifies the form in which variables can be present.

Device manufacturers follow these guidelines and provide device-specific tables that list the address of the variable and how it is addressed.

In order to define the Modbus variable as OPC tag in the ***Modbus Variable Properties*** dialog box of the IBH OPC UA Editor, knowledge of the fieldbus node with its data, data coding, addressing and transactions is required.

Following are excerpts from specifications of the Modbus organization, which are necessary for the definition of the OPC tags. The tables / descriptions of the fieldbus node provide this information in a similar form.

Modbus Functions – Partially – (Modbus Organization)

The action of reading and writing variables is determined by functions.

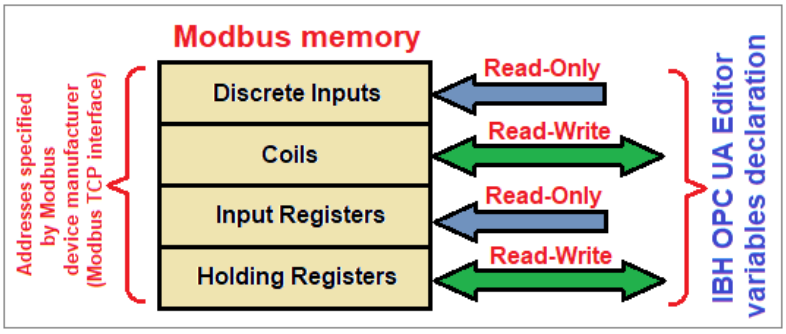
Physical Discrete Inputs	Read Discrete Inputs
Internal Bits or Physical coils	Read Coils
	Write Single Coil
	Write Multiple Coils
Physical Input Registers	Read Input Register
Internal Registers or Physical Output Registers	Read Holding Registers
	Write Single Register
	Write Multiple Registers
	Read/Write Multiple Registers
	Mask Write Register
	Read FIFO queue
File record access	Read File record
	Write File record
Diagnostics	Read Exception status
	Diagnostic
	Get Com event counter
	Get Com Event Log
	Report Server ID
	Read device Identification
Other	Encapsulated Interface Transport
	CANopen General Reference

Most Modbus devices do not provide all the functions. However, writing or reading functions are usually supported by all. Therefore, the IBH OPC UA Editor only supports the functions for reading and writing values.

Modbus data formats

Name data access (Storage area)	Quantity Data type	Access Type memory access	Comment
Discrete Input physical inputs process image	**1 Bit** (Single bit)	**Read-Only**	These data types can be provided by an I / O system.
Coils Discrete outputs physical outputs process image	**1 Bit** (Single bit)	**Read-Write**	These data types can be changed by an application program.
Input Registers	**16-Bit** Wort	**Read-Only**	These data types can be provided by an I / O system.
Holding Registers	**16-Bit** Wort	**Read-Write**	These data types can be changed by an application program.

The Modbus devices provide memory in which variables are available that can be accessed using the access type specified in the table.



Modbus devices usually have separate memory areas per table type.

The addresses of the memory areas and their access options are provided by the Modbus device manufacturers, mostly in tabular form.

Example - Listing: Access to variables (fictitious Modbus devices manufacturer information)

Modbus register addressing

	Addressing		Start address	End address	Access	Description / Storage area
	1 Bit	Register				
Process Data	X(1)		0x0000	0x00CF	Read/Write	Process data interface. Physical inputs, Process image
		X(2)	0x00D0	0x00FF	Read/Write	
	X(3)		0x0100	0x01CF	Read-only	
		X(4)	0x01D0	0x01FF	Read-only	Process data interface. Physical outputs, Process image
		X(5)	0x0200	0x02FF	Read/Write	
	X(6)		0x0300	0x03FF	Read/Write	

	Addressing		Start address	End address	Access	Description / Storage area
	1 Bit	Register				
Diagnosis		X(7)	0x0400	0x040F	Read-only	Status register
		X(8)	0x0410		Read-only	Process image length in bits, analog outputs
		X	0x0411		Read-only	Process image length in bits, analog inputs
		X	0x0412		Read-only	Process image length in bits, digital outputs
		X	0x0413		Read-only	Process image length in bits, digital inputs
Special Register		X(9)	0x0420	0x042F	Read/Write	Watchdog Register
		X(10)	0x0430	0x043F	Read/Write	Error Register
		X	0x0440	0x044F	Read/Write	Command Register
		X	0x0450	0x045F	Read/Write	Internal Register

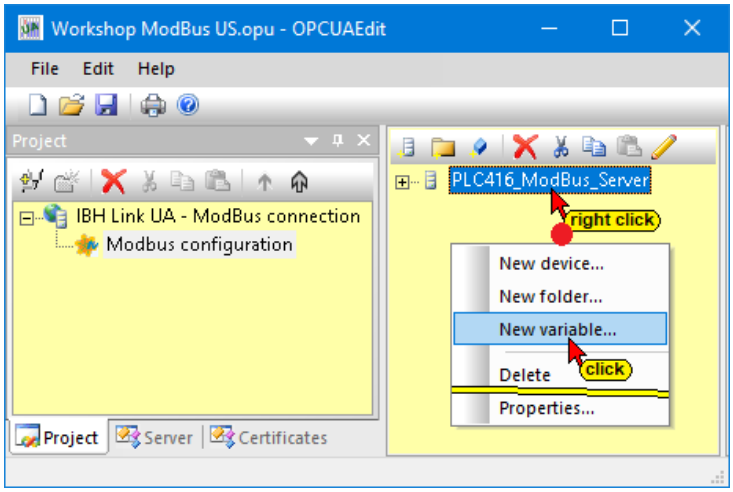
(nn) As an example, for the definition of a variable exists.

Note:

The address information in the Modbus devices Manufacturer specifications are often in hexadecimal form. These addresses are to be converted into a decimal address for input in the IBH OPC UA Editors.

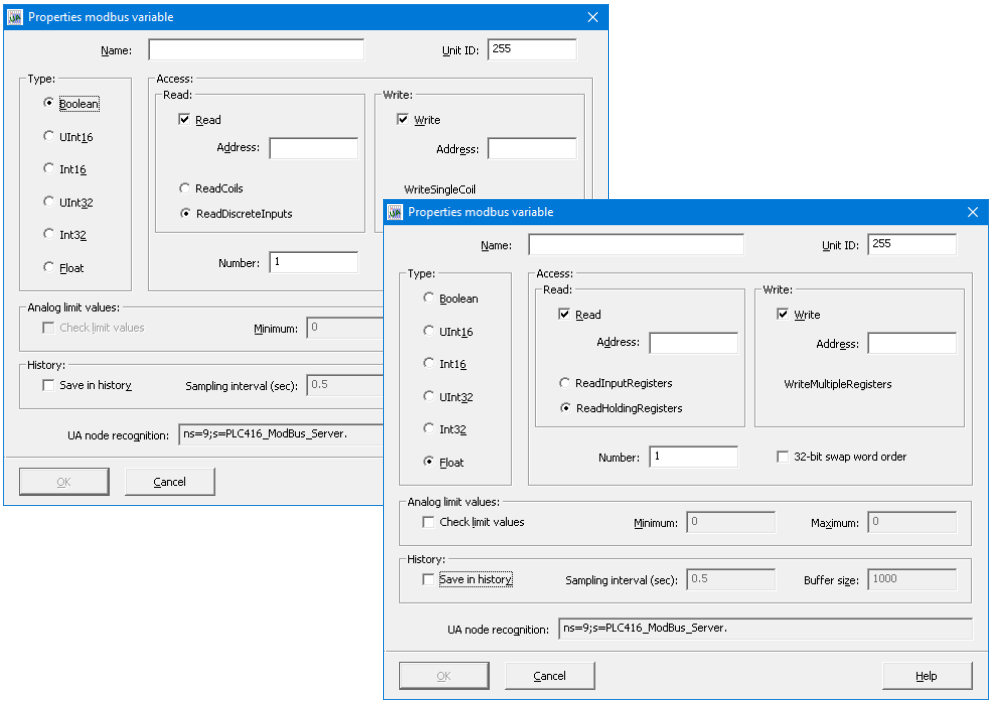
1.3 Defining Modbus variables in the IBH OPC UA Editors

Right-clicking on the Modbus device name (PLC 416 ModBus Server) opens the context menu.



The ***New Variable*** command opens the ***Modbus Variable Properties*** dialog box.

This dialog defines the variables that the OPC server should connect to. This can be a write variable, a read variable or a read / write variable.



1.3.1 Properties dialog box Modbus variable

Modbus TCP / RTU interface

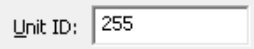
Modbus device manufacturers describe in tables the Modbus fieldbus node functions. Based on these descriptions, the definitions of the accesses to variables are defined.

Name (Variable)

The name can be selected freely but must correspond to the OPC UA specification (letters **A-Z**, **a-z**, numbers **0-9**, no special characters, no symbols, no dots, no colon.) As a special character only _ under line.

Unity-ID

For Modbus TCP the Unity-ID = 255. This is set at default. For Modbus RTU, the address must be set according to the slave address.



Type

Boolean

One (1) bit information (**Boolean**), which can have the states TRUE (1) and FALSE (0). A variable of type **Boolean** occupies 1 bit in a register address (16 bits). An array of 16 variables of type Boolean occupy an entire register address (16 bits).

UInt16

Positive 16-bit **unidirectional integer** (number) between 0 and 65,535 (2^0 to 2^{+16}). A **UInt16** variable occupies a register address (16 bits).

Int16

Positive or negative 16-bit integer (number) between -32,767 and +32,767 (-2^{+15} to 2^{+15} -1). An Int16 variable occupies a register address (16 bits).

UInt32

Positive 32-bit unidirectional integer (number) between 0 and 4,294,967,295 (2^0 to 2^{+32}). A **UInt32** variable occupies two (16-bit) register addresses.

Int32

Positive or negative 32-bit integer (number) between -2,147,483,654 and +2,147,483,654 (-2^{+31} to 2^{+31} -1). An **Int32** variable occupies two (16-bit) register addresses.

Float

A variable of data type **Float** represents a fractional number that exists as a 32-bit floating-point number (REAL). A **Float** variable occupies two (16-bit) register addresses.

1.3.2 Addresses when accessing read / write

Note:

The address information in the Modbus devices Manufacturer specifications are often in hexadecimal form. These addresses are to be converted into a decimal address for input in the IBH OPC UA Editors.

Data type UInt16 and Int16

A word address (16-bit) is used for accessing variables of a data type. Address 0 addresses the variable that occupies the first 16 bits of a data area. With address 1, the variable occupying the second 16 bits of a data area is addressed. Address 3 addresses the third variable (16-bit) of a data area, and so on.

Data types UInt32, Int32 and Float

Two words (2 x16-bit = 32 bits) are always required for accessing variables of these data types. Address 0 addresses the first 32-bit variable of these data types. Address 1 must not be addressed because this address would address the low word of the 32-bit variable. Address 2 addresses the variable occupying two words 2 and 3 in the data area. The address 4 addresses the third 32-bit variable. etc.

The order of the two 16-bit words, which consist of the data types UInt32, Int32 and Float, can be set.

- **High-Word** – Low-Word or Low-Word – **High-Word**.

Data type Boolean

A bit address is used to access bits in the data area.

- Address 0 addresses bit 0 in the word address 0 area.
- Address 1 addresses bit 1 in the word address 0 area.
- Address 16 addresses bit 0 in the word address 1 area.
- Address 66 addresses bit 2 in the word address 3 area.

1.4 Access Read Only

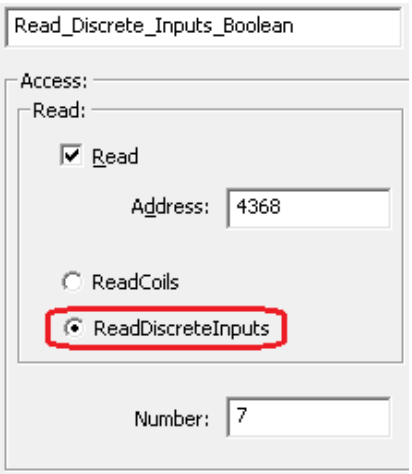
Read Discrete Input (read only – Bit access)

Data type Boolean

Example: - Fictitious Modbus table (3)

Start Word Address 0111_{hex} = Bit Address 1110_{hex} = 4368_{dec} - Read-only - Bit Access - Physical inputs, Process image. There are 7 input bits to be defined as OPC tags.

This setting addresses variables from the registers of the ***Discrete Inputs*** whose contiguous status comes from digital inputs.
The address of the first variable and the number of variables must be specified.



Screenshot: 7 individual bits are read from bit address 4368 of the memory area of the physical inputs.

Read Input Registers (Read Only)

All data types except Boolean (example: UInt16 - 16-bit)

Example: - Fictitious Modbus table (7)

Initial register address 400_{hex} = 1024_{dez} - Read-only - Word access - The contents of 10 status registers are to be addressed as OPC tags.

This setting addresses variables from the diagnostics area (***Input Registers***). The start register address and the number of registers must be specified.

Screenshot: 10 numbers (unsigned) are read with 16 bits each from word 1024 of the status registers.

1.5 Access Read and Write

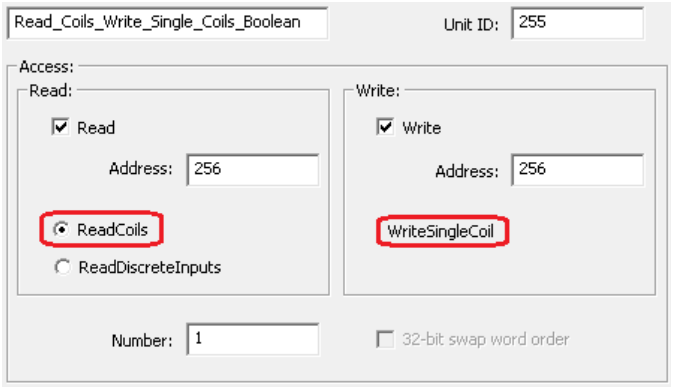
Read Coils / Write Single Coil

Data type Boolean

Example: - Fictitious Modbus table (1)

Register Address 0010_{hex} = Bit Address 100_{hex} = 256_{dec} - Read-Write - Bit Access - Physical Inputs, Process image. Define 1 input bit as OPC tag.

With this setting, variables from the registers of the coils are addressed. The address of the variable and the number of variables (1) must be specified.



Screenshot: 1 single bit is read with bit address 256 of the memory area of the physical inputs.

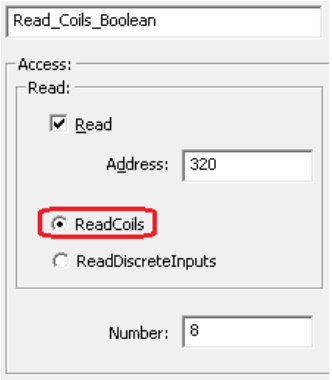
Read Coils

Data type Boolean

Example: - Fictitious Modbus table (1)

If such a variable is only defined as a read variable, it is defined as an OPC tag with the status **Read**.

Register Address 0014_{hex} = Bit Address 140_{hex} = 320_{dec} - Read - Bit Access - Physical Inputs, Process image. 8 input bits should be defined as OPC tag.



Screenshot: 8 single bits are read from the bit address 320 of the memory area of the physical inputs.

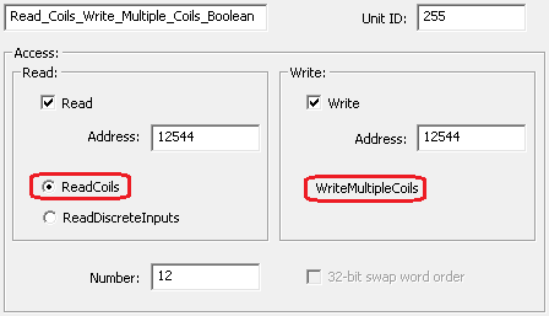
Read Coils / Write Multiple Coils

Data type Boolean

Example: - Fictitious Modbus table (6)

Register Address 0310_{hex} = Bit Address 3100_{hex} = 12544_{dec} - Read-Write - Bit Access - Physical Inputs, Process image.

12 input bits should be defined as OPC tag.



With this setting, variables are addressed from the registers of the coils whose contents reflect individual bits. These can be individual outputs as well as individual inputs. The address of the first specified variable and the number of variables are specified.

Screenshot: 12 individual bits are defined from the bit address 12544 of the memory area of the physical outputs.

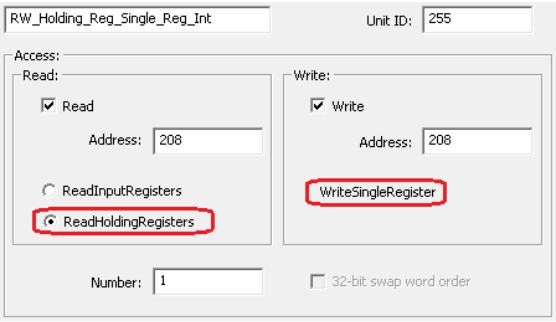
Read Holding Register / Write Single Register

(All data types except Boolean)

Example: - Fictitious Modbus table (2) - Data type INT16 (integer).

Register address $00D0_{hex}$ = 208_{dec} - Read-Write - word access - Physical Inputs, Process image.

One (1) integer number should be defined as OPC tags.



With this setting variables from the ***Holding Registers***

are addressed whose contents reflect individual registers. This can be, for example, an analogue input. The address of the first variable and the number (1) of the variables are specified.

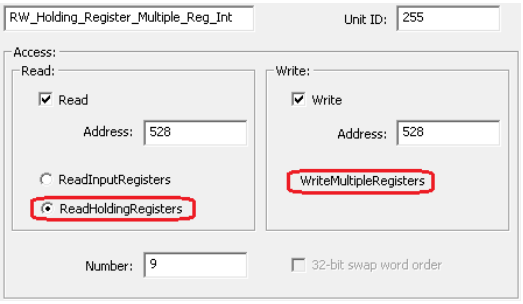
Screenshot: An integer number is defined from the word address 208 of the storage area of the physical outputs.

Read Holding Registers / Write Multiple Registers

(All data types except Boolean)

Example: - Fictitious Modbus table (5) - Data type INT16 (integer).

Register Address 0210_{hex} = 528_{dez} - Read-Write - Word Access - Physical Outputs, Process image.



9 integer numbers should be defined as OPC tags.

With this setting variables from the ***Holding Registers*** are addressed whose contents reflect individual registers. These can be individual outputs (analog outputs). The address of the first specified variable and the number of variables are specified.

Screenshot: 9 integer numbers from the word address 528 of the memory area of the physical outputs are defined as OPC tags.

Read Holding Registers

(All data types except Boolean)

If such a variable is only defined as a read variable, it is defined as an OPC tag with the status **Read**.

Example: - Fictitious Modbus table (8) - Data type UINT16 unsigned integer number.

Register address 0450_{hex} = 1104_{dez} - Read-Write - word access - special register.

6 unsigned integer numbers should be defined as OPC tags.

With this setting variables from the ***Holding Registers*** are addressed whose contents reflect individual registers. The address of the first specified variable and the number of variables are specified.

Screenshot: 6 numbers (unsigned) from the word address 1104 of the memory area of the special registers are defined as OPC tags.

Read Holding Registers / Write Multiple Registers

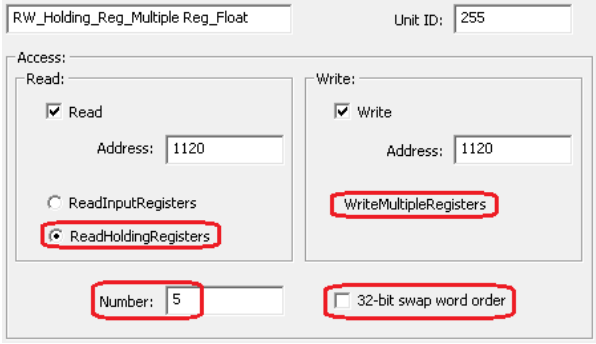
(Data types UInt32, Int32, Float)

Example: - Data type float (real number).

Register Address 0460_{hex} = 1120_{dez} - Read-Write - Word Access - Physical Outputs, Process image.

You must define 5 floating point numbers as OPC tags.

With this setting variables from the ***Holding Registers*** are addressed whose contents reflect individual registers. These can be individual outputs (analog outputs).



The address of the first specified variable and the number of variables are specified.

Screenshot: 5 floating-point numbers starting from the word address 1120 are defined in the ***Holding Registers*** as OPC tags.

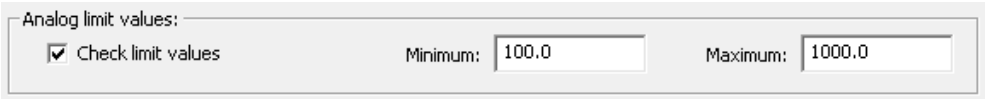
Note:

UInt32, **Int32**, and **Float** numbers occupy two (2) 16-bit (32-bit) words. The order of the two 16-bit words that make up these data types is adjustable.

- **High Word** - Low Word or Low Word - **High Word**

1.5.1 Analog-limits

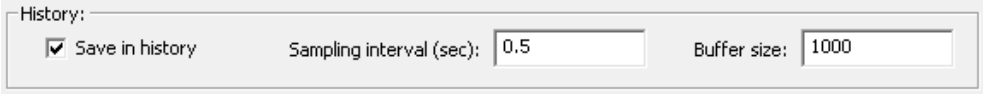
Analog limit values can be specified.



1.5.2 History

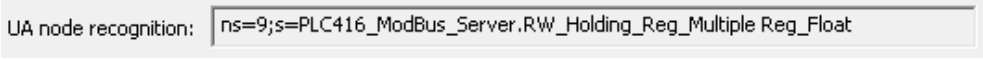
While ***OPC Data Access*** provides real-time access to data, ***OPC Historical Data Access***, also known as OPC HDA, supports access to data already stored.

Activation of a variable as historical data as well as sampling interval and number of values (buffer size) is done via the dialog box.



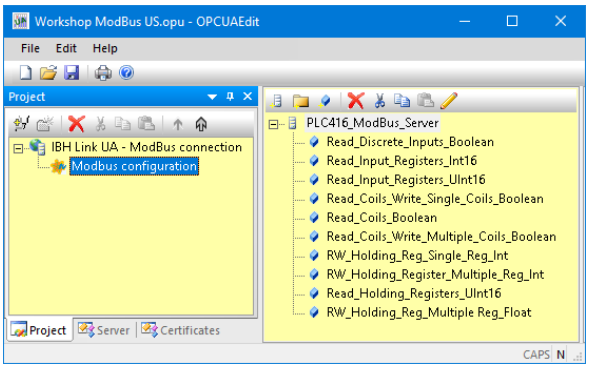
OPC UA Node detection

The OPC UA Node name of a variable is displayed in the dialog box.



1.6 Transfer Modbus configuration to the OPC UA server (IBH Link UA)

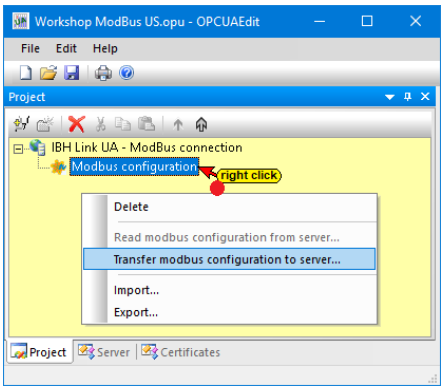
The variables defined as OPC tags are displayed.



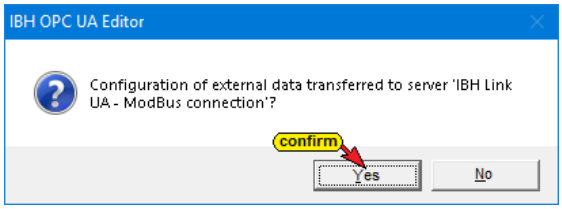
Name	Data type	Access	R address	W address	Number of	Node name
Read_Discrete_Inputs_Boolean	Boolean	R	4368		7	PLC416_ModBus_Server.Read_Discrete_Inputs_Boolean
Read_Input_Registers_Int16	Int16	R	480		4	PLC416_ModBus_Server.Read_Input_Registers_Int16
Read_Input_Registers_UInt16	UInt16	R	1024		10	PLC416_ModBus_Server.Read_Input_Registers_UInt16
Read_Coils_Write_Single_Coils_Boolean	Boolean	RW	256	256	1	PLC416_ModBus_Server.Read_Coils_Write_Single_Coils_Boolean
Read_Coils_Boolean	Boolean	R	320		8	PLC416_ModBus_Server.Read_Coils_Boolean
Read_Coils_Write_Multiple_Coils_Boolean	Boolean	RW	12544	12544	12	PLC416_ModBus_Server.Read_Coils_Write_Multiple_Coils_Boolean
RW_Holding_Reg_Single_Reg_Int	UInt16	RW	208	208	1	PLC416_ModBus_Server.RW_Holding_Reg_Single_Reg_Int
RW_Holding_Register_Multiple_Reg_Int	Int16	RW	528	528	9	PLC416_ModBus_Server.RW_Holding_Register_Multiple_Reg_Int
Read_Holding_Registers_UInt16	UInt16	R	1104		6	PLC416_ModBus_Server.Read_Holding_Registers_UInt16
RW_Holding_Reg_Multiple Reg_Float	Float	RW	1120	1120	5	PLC416_ModBus_Server.RW_Holding_Reg_Multiple Reg_Float

If all Modbus variables are defined as OPC UA tags, the Modbus configuration can be transferred to the OPC UA server.

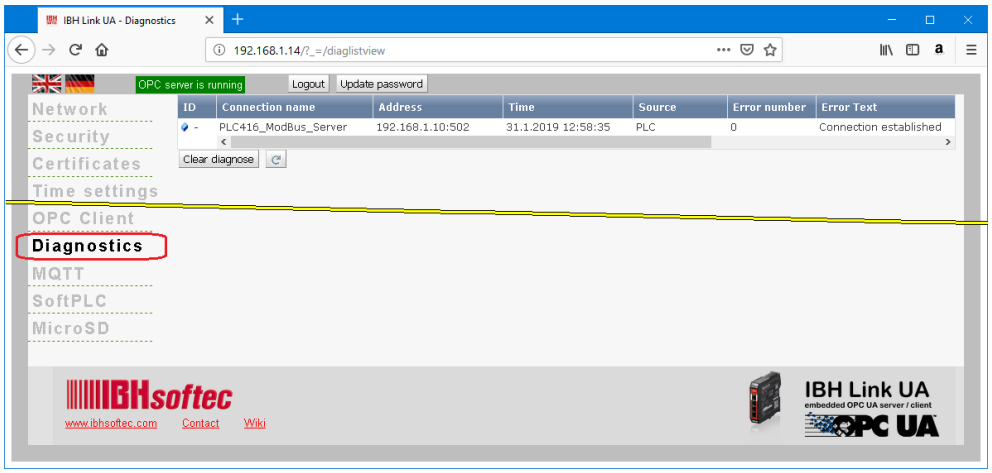
A right-click on ***Modbus configuration*** opens the context menu.



The command to transfer the Modbus configuration must be confirmed.



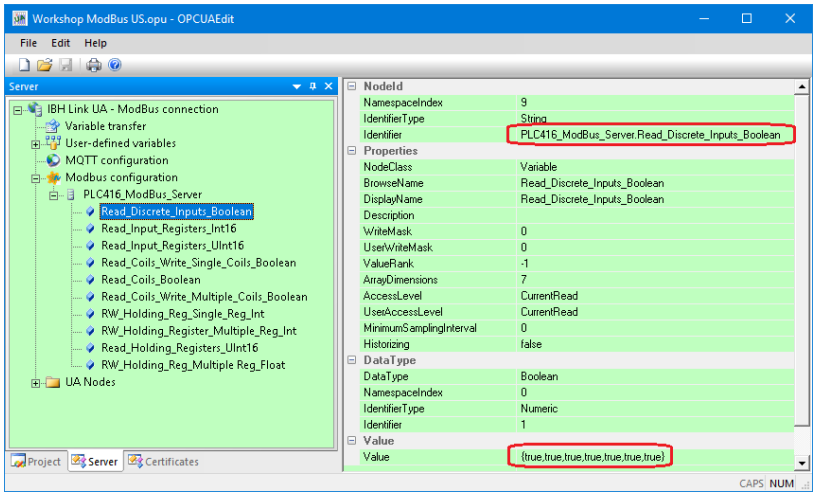
The transmitted Modbus configuration is displayed under ***Diagnostics*** in the IBH Link UA Browser window.



1.7 IBH OPC UA Editor Server Window

A ***Modbus configuration*** successfully transmitted to the OPC UA server can be displayed online in the server window.

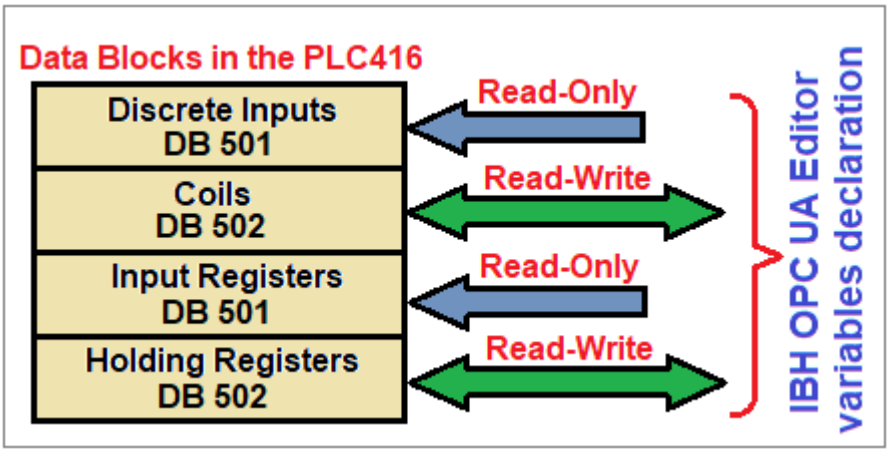
The variables (***Modbus configuration, Modbus device, OPC tag***) are listed in the left-hand server window. Clicking on a variable displays the variable definitions with the status in the right-hand server window. The status of this OPC tag is constantly being updated.



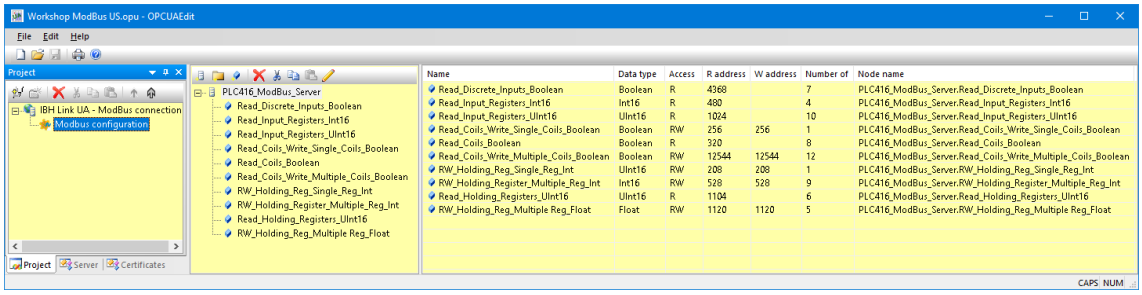
1.8 Modbus connection - examples

The IBH SoftPLC PLC416 has the option of a Modbus connection. In the example, variables are defined as OPC tags. This Modbus configuration is transmitted to the IBH Link UA and the variables are displayed in the ***UAExpert client program***.

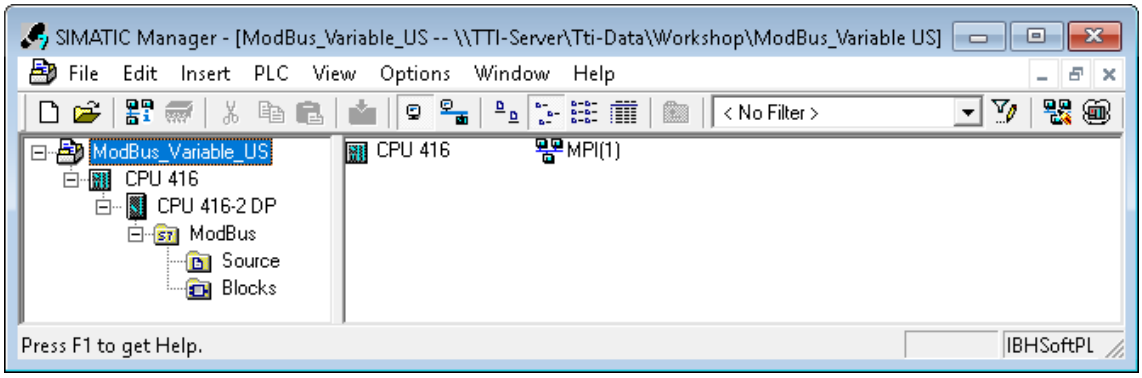
The data blocks DB501 (**DBIn - R**ead) and DB502 (**DBOut - R**ead / **W**rite) are created in the PLC416.



The OPC tags from the IBH UA Editor Program ***Workshop ModBus US.opu*** are used. The file must be opened with the IBH UA Editor and the Modbus configuration must be transferred to the IBH Link UA.

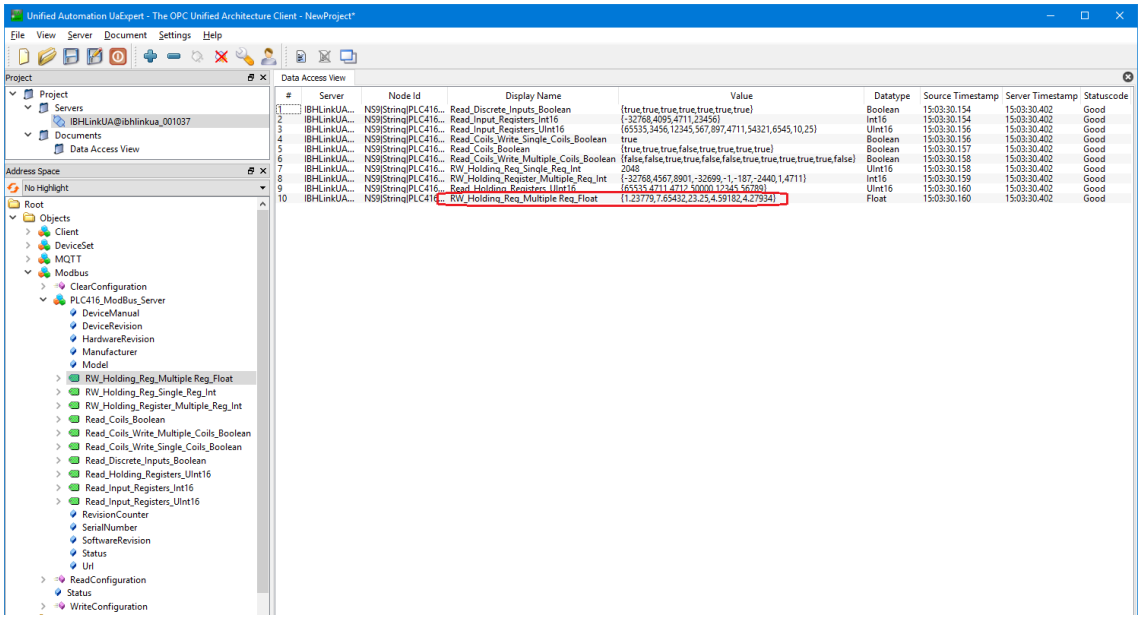


The STEP 7 PLC program must be opened and the blocks with the system data must be loaded into the PLC416 soft PLC.



1.9 Unified Automation UaExpert - The OPC Unified Architecture Client

The ***UaExpert program window*** lists the ***OPC tags*** transmitted by the IBH OPC UA Editor and the corresponding ***UA nodes***.



The values of the variables can be changed via the diagnostics of the PLC416.

Example 1: **Read_Discrete_Inputs_Boolean,** (Read only, bit access, data type Boolean). Initial word address 0111_{hex} = 273_{dec},
Bit Address 1110_{hex} = 4368_{dec} - word access - DB501 - DW 546 (byte 546). 7 bits defined as OPC tags.

Example 2: **Read_Input_Registers_Int16,** (Read only), All data types except Boolean (Int16), Start Word Address $01E0_{hex}$ = 480_{dec} Word Access - DB501 - DW 960 to DW 966.
4 integer numbers defined as OPC tags.

Example 3: **Read_Input_Registers_UInt16,** (Read only), All data types except Boolean (UInt16 - unsigned integer number), start word address 400_{hex} = 1024_{dec} word access - DB501 - DW 2048 to DW 2066. Defines 10 unsigned integer numbers as OPC tags.

Example 4: **Read_Coils_Write_Single_Coils_Boolean,** (Read-Write, Bit Access, Data Type Boolean), Start Word Address 0010hex = 16dec Bit Address 100_{hex} = 256_{dec} - Word Access - DB502 - DW 32 (Byte 32). 1 bit defined as OPC tag.

Example 5: **Read_Coils_Boolean,** (read only, bit access, Boolean data type), start word address 0014_{hex} = 20_{dec} Bit address 140_{hex} = 256_{dec}

Word access - DB502 - DW 40 (byte 40). 8 bits defined as OPC tags.

Example 6: **Read_Coils_Write_Multiple_Coils_Boolean,** (Read-Write, Bit Access, Data Type Boolean), Start Word Address 0310_{hex} = 784_{dec}, bit address 3100_{hex} = 12544_{dec} - word access - DB502 - DW 1568. 12 bits defined as OPC tags.

Example 7: **RW_Holdinq_Reg_Single_Reg_Int,** (read-write, data type INT16 integer number), start-word address $00D0_{hex}$ = 208_{dec}, word access - DB502 - DW 416. One (1) integer numbers defined as OPC tag.

Example 8: **RW_Holdinq_Register_Multiple_Reg_Int,** (read-write, data type INT16 integer number), start-word address 0210_{hex} = 528_{dec}, word access - DB502 - DW 1056 to DW 1072. 9 integer numbers defined as OPC tags.

Example 9: **Read_Holding_Register_UInt16,** (read-write, data type UINT16 unsigned integer number), start word address 0250_{hex} = 1104_{dec}, word access - DB502 - DW 2208 to DW 2218. 6 unsigned integer numbers defined as OPC tags.

Example 10: **RW_Holding_Reg_Multiple_Reg_Float,** (read-write, data type FLOAT floating point number), start word address 0460_{hex} = 1120_{dec}, 2-word access - DB502 - DW 2240 to DW 2258. 5 Floating point numbers defined as OPC tags.