IBHsoftec

TTi
Trans-Tech International
Ingenieurbüro für Technologie Transfer
Dipl.-Ing. B. Peter Schulz-Heise

IBH Link UA

Python / Methods / Data Models

Version 1.0

IBHsoftec GmbH
Turmstr. 77
64760 Oberzent / Beerfelden
Tel.: +49 6068 3001
Fax: +49 6068 3074
info@ibhsoftec.com
www.ibhsoftec.com

TTi Ingenieurbüro für
Technologie Transfer
Dipl. Ing. B. Peter Schulz-Heise
Tel.: +49 6061 3382
Fax: +49 6061 71162
tti@ schulz-heise.com
www.schulz-heise.com

Contents

Configuration example

PLC program (TIA) / SiOME-, IBH OPC UA editor-, Python-files

PLC program	**Project: CPU 1500 TIA NodeSet** TIA Portal V18; CPU 1500.
Siemens SiOME file	**SiOME Nodeset Workshop Example.xml**
IBH OPC UA editor file	**Workshop CPU 1500 NodeSet.opu**
Python project file	**Python-Workshop.py**
External Python editor activation file	**ibhua.pyi**
All files	**NodeSet Manual CPU 1500 Example.zip**

1 IBH Link UA – Python / Methods / Data Models

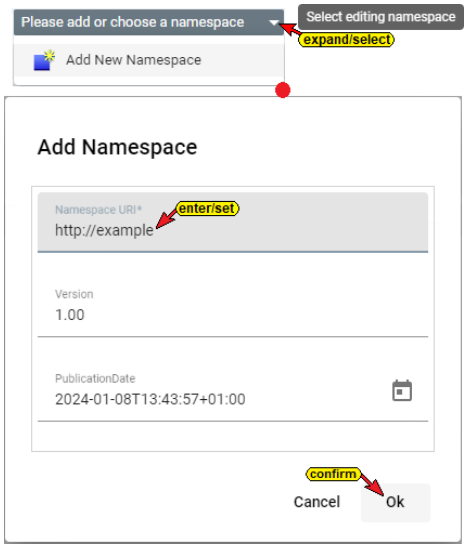
In the following example, an **OPC UA information model** is created using the Siemens OPC UA Modeling Editor (***SiOME***). A Python program is assigned to the created Nodset configuration file using the IBH OPC UA editor and transferred to the IBH Link UA.

The method (Python program) reads a value from a CPU 1500, modifies it, and stores it in another data block of the CPU 1500. The handling of variables is shown using examples.

1.1 Creating an OPC UA information model.

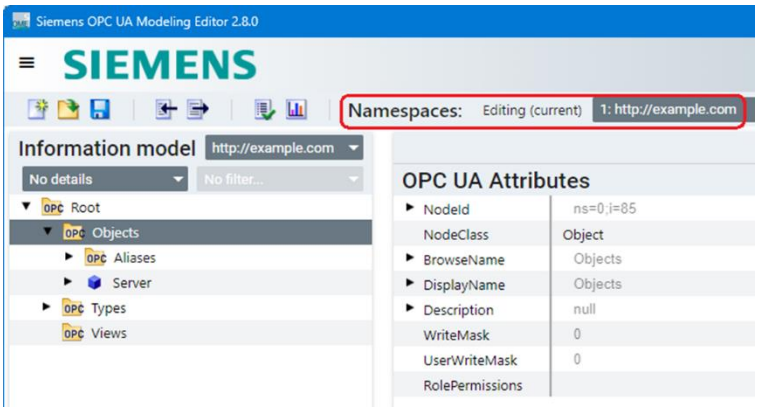
Start Siemens OPC UA Modeling Editor (SiOME V2.8.0).

Set new namespace.



In order not to work with the stored namespace of the OPC Foundation (http://opcfoundation.org/ua/), a separate namespace is defined.

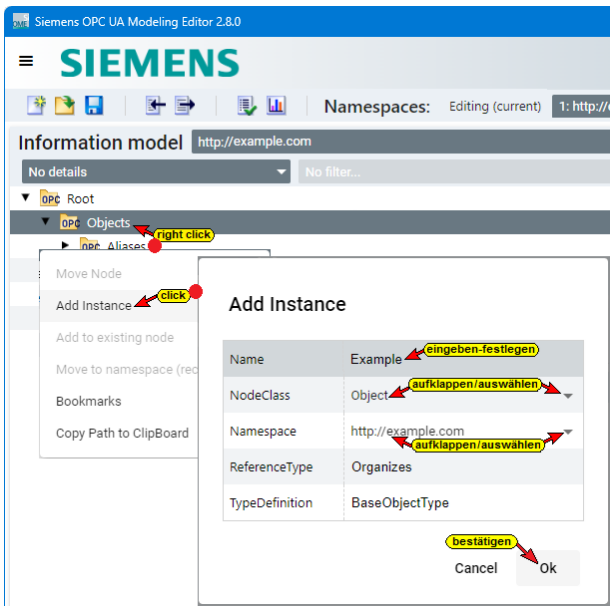
The namespace is defined in the ***URI*** (Uniform Resource Identifier) standard, an arbitrary name ***(example)*** with the domain ***.com***.

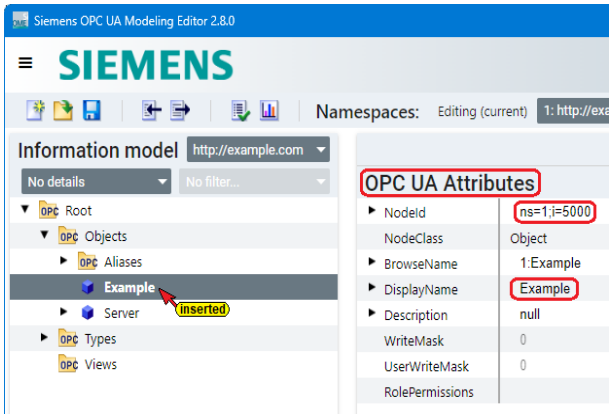


Creating an instance (Add Instance / Object).

Right-clicking ***Objects*** and clicking the ***Add Instance*** command in the opened context menu opens the ***Add Instance*** dialog box. Enter the name ***Example***, select ***NodeClass Objects*** and ***Namespace http://example.com***.

Example object added – OPC UA Attributes.





To identify the ***Example*** object, it is important that the OPC UA attribute ***NoteId*** is displayed numerically (***ns=1;i=5000***). This makes the assignment in the Python program in IBH Link UA easier.

Insert variable.

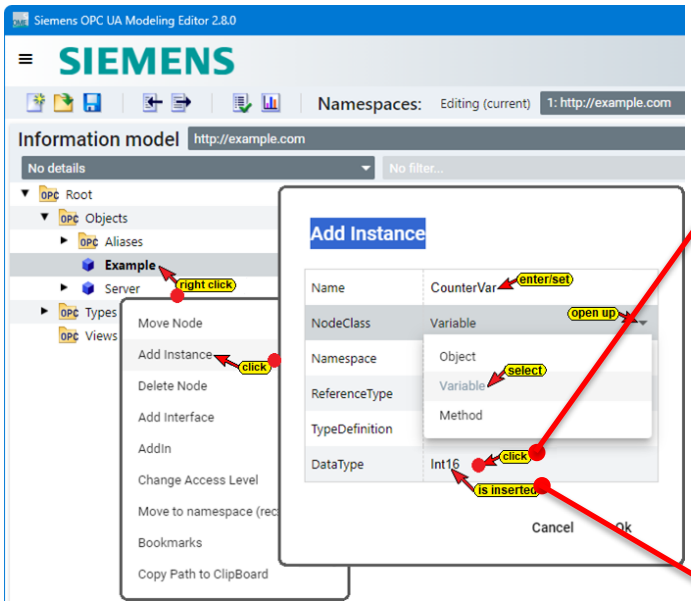
Four (4) variables are included for the example. The variable (***CounterVar)*** continuously reads a value (***Int16***) from CPU 1500 (***CounterValues / DB 5***). The Python program divides the value (***Int16)*** and writes this value (***Int16)*** into the second variable (***OutRes***), which is given to the variable ***ValueIn*** (***DataIn / DB 10***).

The value of ***MaxValue*** from CPU 1500 (***CounterValues / DB 5***) is transferred to the variable ***CounterVar*** (***Int16***).

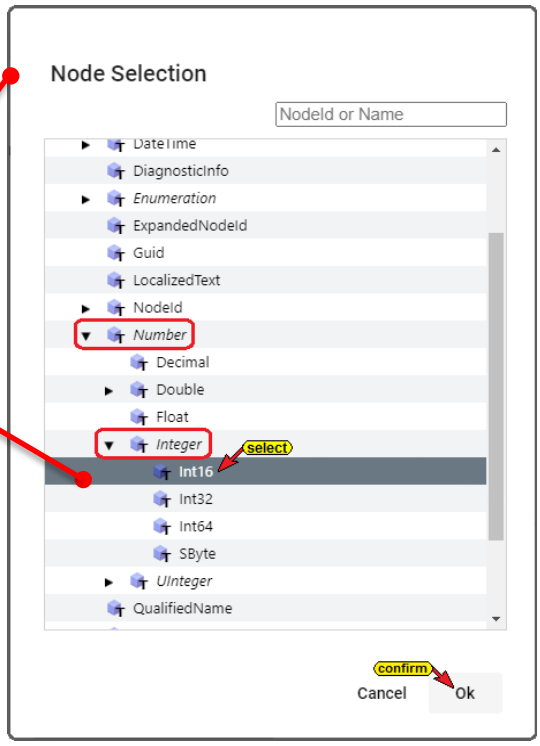
The ***InData*** variable is assigned a value (***Float***) by the Python program. The variable ***InVar*** (***Int16***) performs a calculation.

The value of the variable ***OutRes*** (***Float***) is transferred to the variable **RealData** (***DataIn / DB 10***). A ***UserMethod*** with a calculation is also ***inserted***.

Insert variable CounterVar.

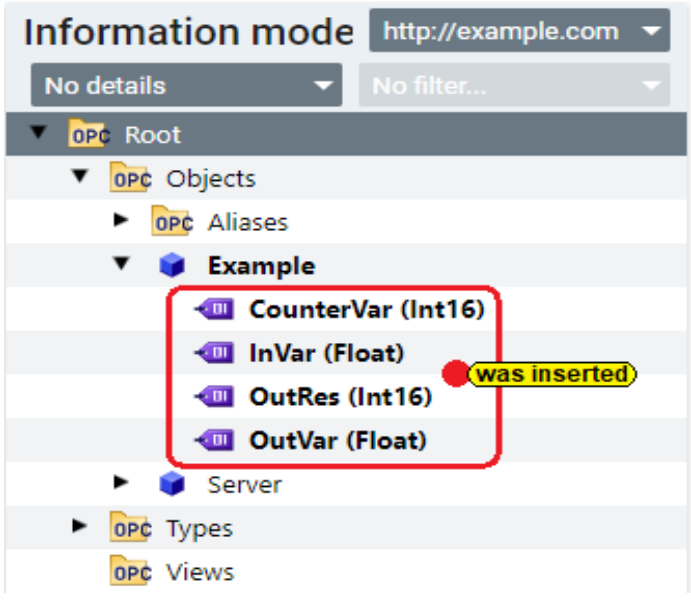


Right-clicking on ***Example*** and clicking on the ***Add Instance*** command in the opened context menu opens the ***Add Instance*** dialog box.



Enter the name ***CounterVar***, select ***NodeClass*** variable and click ***DataType***. The ***Node Selection*** dialog box opens.

In the ***Node Selection*** dialog box, click ***Number / Integer / Int16*** to set the data format for the ***CounterVar*** variable. Close the dialog boxes by clicking ***OK***.

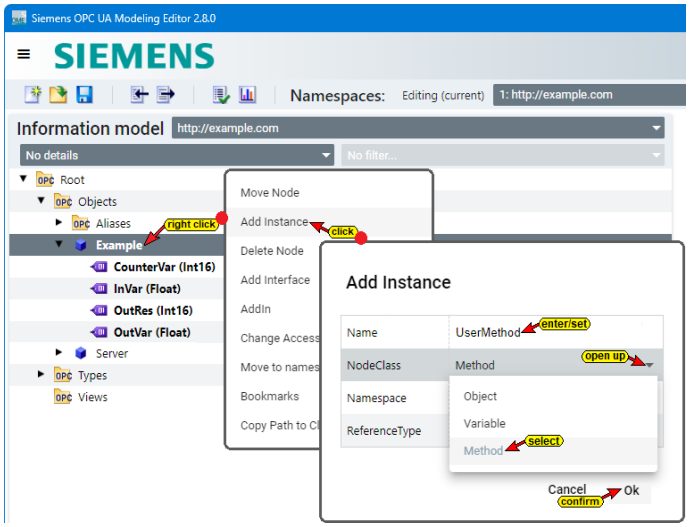


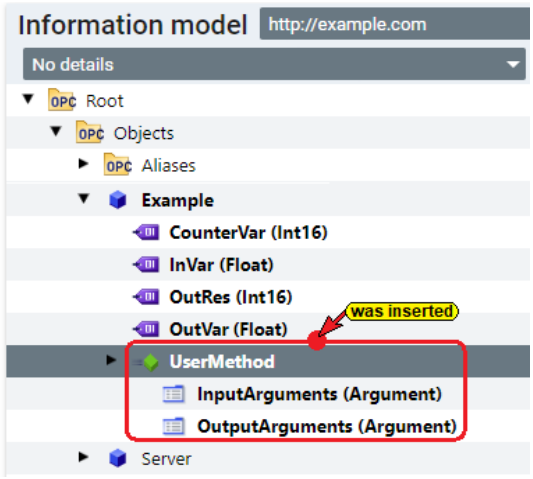
In the same way, create the variable ***OutRes*** with the data format ***Int16***. The variable ***InVar*** and ***OutVar*** are created with the data format ***Float***.

The variables are inserted.

Create method (Add Instance / Method).

Right-clicking on ***Objects*** and clicking the ***Add Instance*** command in the opened context menu opens the **Add Instance** dialog box. Enter the name ***UserMethod***, select NodeClass ***Method*** and ***Namespace http://example.com*** and confirm with ***OK***.





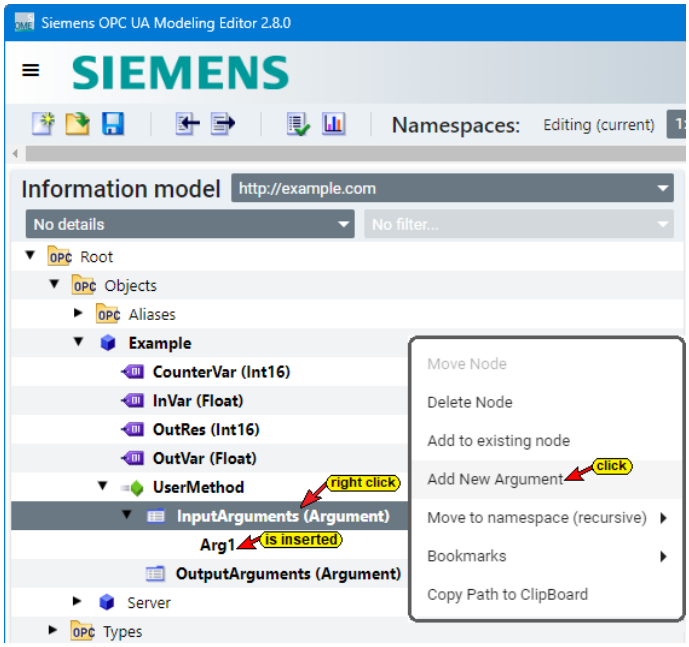
The method is inserted.

The method contains folders to set ***input*** and ***output arguments***.

Set input/output arguments.

Right-clicking on ***InputArguments*** and clicking the ***Add New Arguments*** command in the opened context menu inserts the ***Arg1***.

By reopening the context menu and clicking the **Add New Arguments** command, additional arguments will be added.



In the same way, arguments can be created in the ***OutputArguments*** folder.

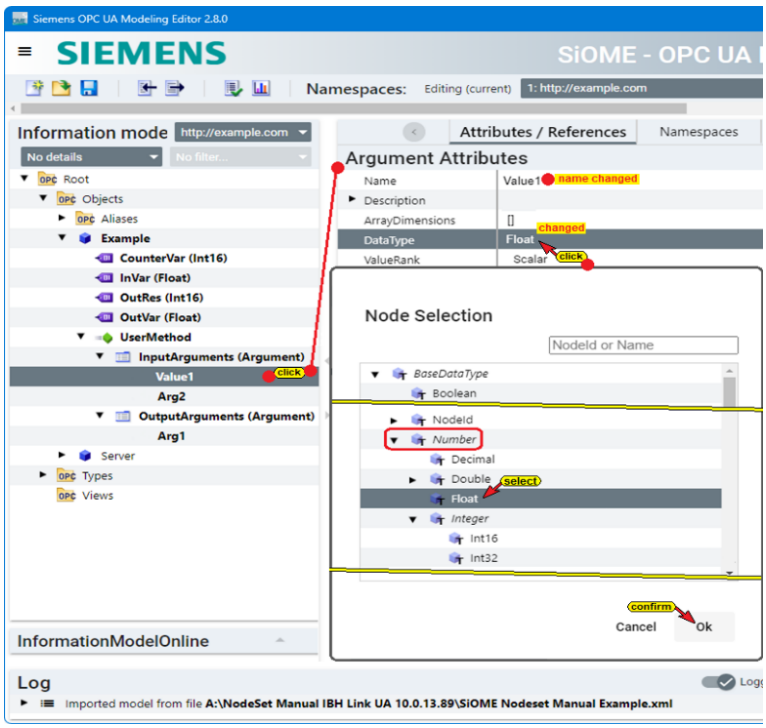
Argument DataType and change Name.

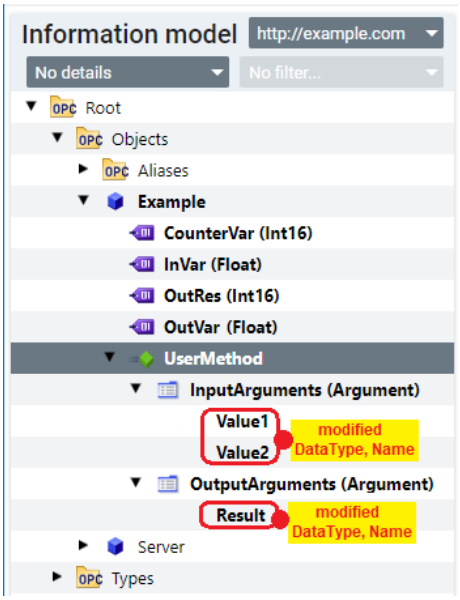
By clicking on the method variable, argument attributes of the variable are listed.

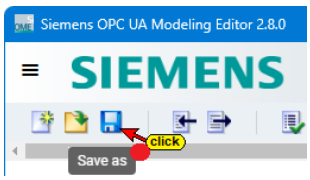
The ***DataType*** of the variable of the ***InputArgument Arg1 (Value1)*** is set to ***Float*** and the name is changed to Value1 (Arg1).

The ***DataType*** of the variable of the ***InputArgument Arg2 (Value2)*** is set to ***Float*** and the name is changed to Value2 (Arg2).

The ***DataType*** of the variable of the ***OutputArgument Arg1 (Result)*** is set to ***Float*** and the name is changed to Result (Arg1).



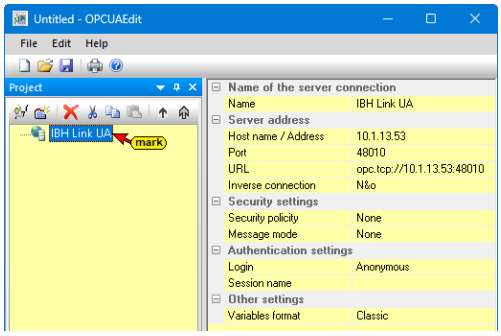




All required specifications have been completed for the example. By clicking the ***Save As*** symbol, the configuration is saved as an ***.xml*** file. The ***.xml*** file (***SiOME Nodeset Manual Example.xml***) is ready to be read into the IBH UA Editor as a ***NodeSet***.

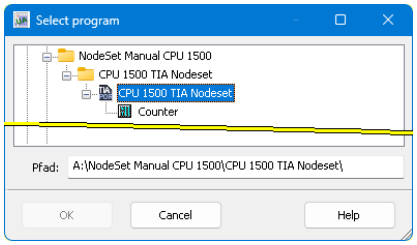
1.2 IBH OPC UA Editor – Read in NodeSet preparation.

1. Open IBH OPC UA Editor.
2. Create server connection. In the example an IBH Link UA ***QC*** is used. The OPC server should connect to a ***CPU 1500*** IP address ***10.1.13.49***. All devices are connected in ***SubNet IP: 10.1.13.0/24***. The IP address of the ***IBH Link UA QC*** (control plane) is ***10.1.13.53***. The security method selected is ***None*** and the variable format is ***Classic***.

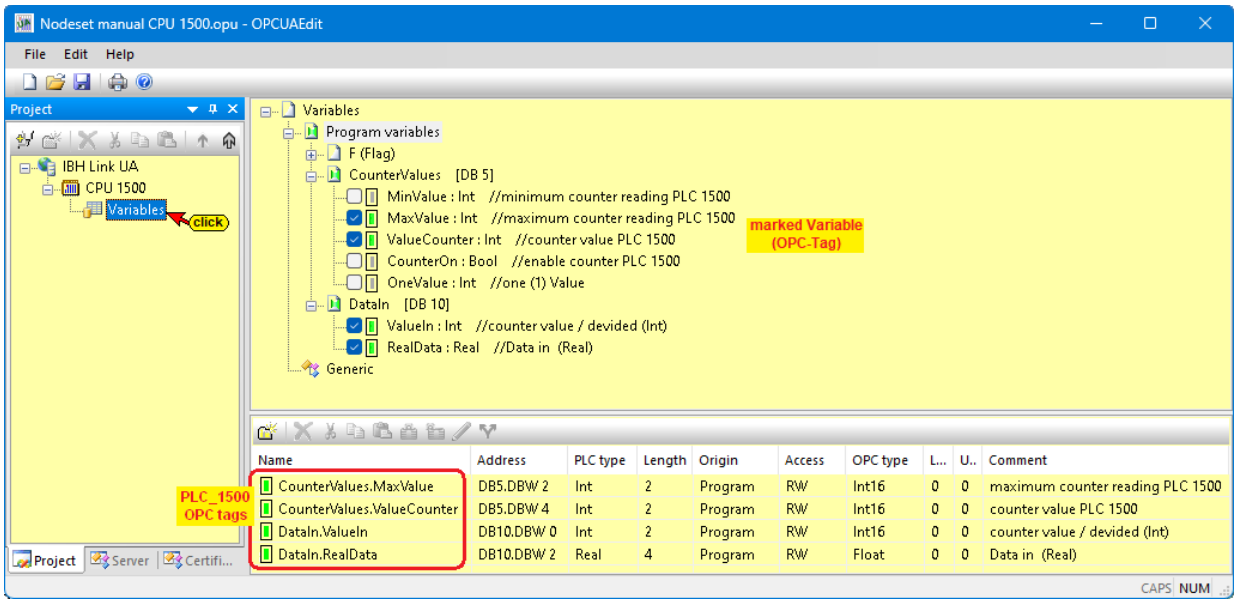


3. Specify connection settings for the controller with the name ***CPU 1500***, the protocol ***S7-1500 TCP /IP*** and the IP address ***10.1.13.49***. Test the connection as successful.

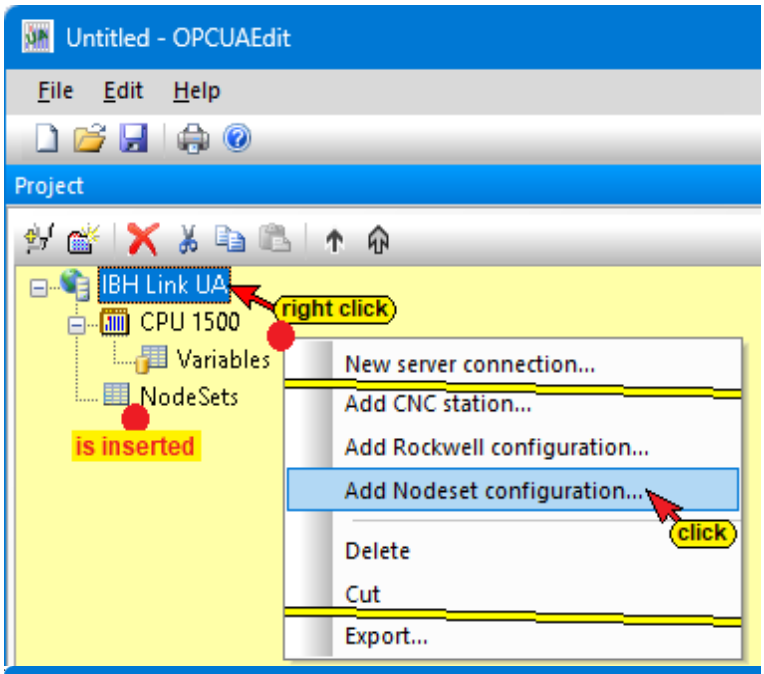
4. Program assignment CPU 1500 TIA NodeSet / ***Counter***.
5. Select the variables ***ValueCounter*** and ***MaxValue*** from the data block ***CounterValues (DB5)*** as the OPC tag.
6. Select the ***ValueIn*** variable and the ***RealData*** variable from the data block ***DataIn (DB10)*** as the OPC tag.



OPC variable CPU 1500

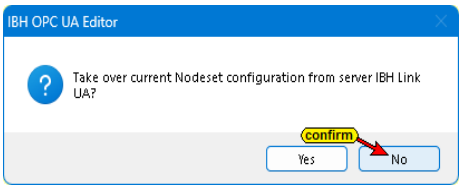


1.2.1 Add NodeSet configuration.

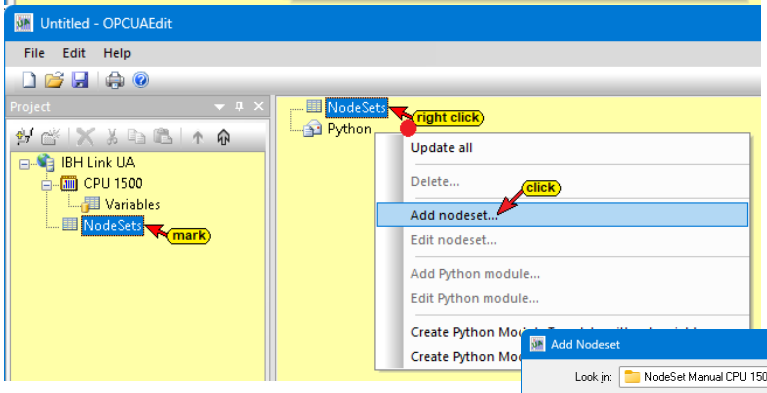


Right-clicking on **IBH Link UA** and clicking on the ***Add NodeSet Configuration…*** command. A dialog box opens.

In the example, the node set configuration should be created from the ***SiOME*** editor.

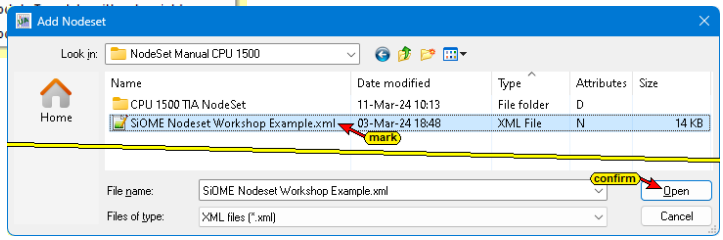


Confirm ***No*** in the dialog box.



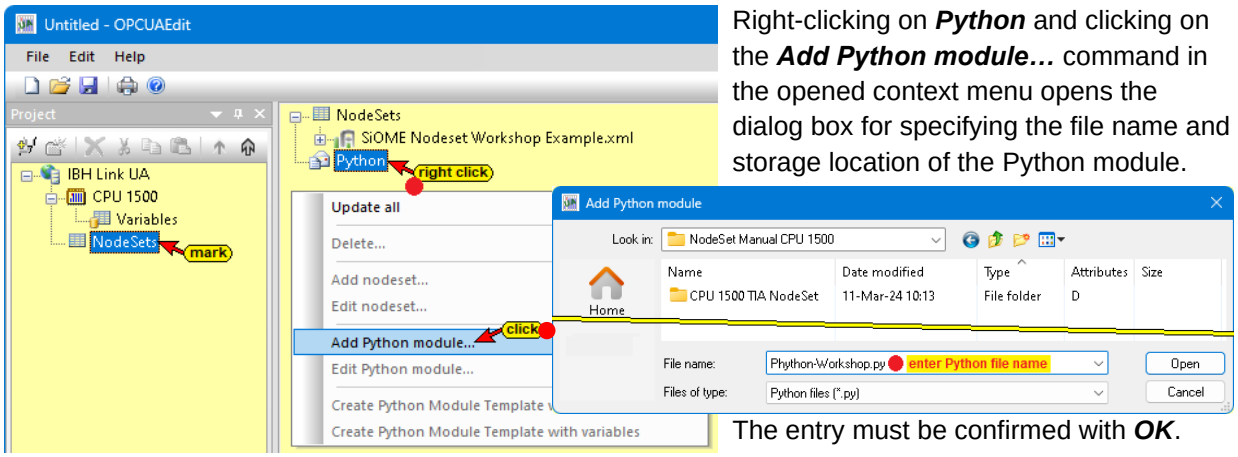
If ***NodeSet*** is selected in the right window, ***NodeSet*** and ***Python*** are displayed in the left window.

Right-clicking on ***NodeSet*** and clicking on the ***Add NodeSet…*** command in the opened context menu. In the ***Add Nodeset*** dialog box select the ***Nodeset*** configuration from the ***SiOME*** editor.



The ***.xml*** file SiOME Nodeset Manual Example has been adopted.

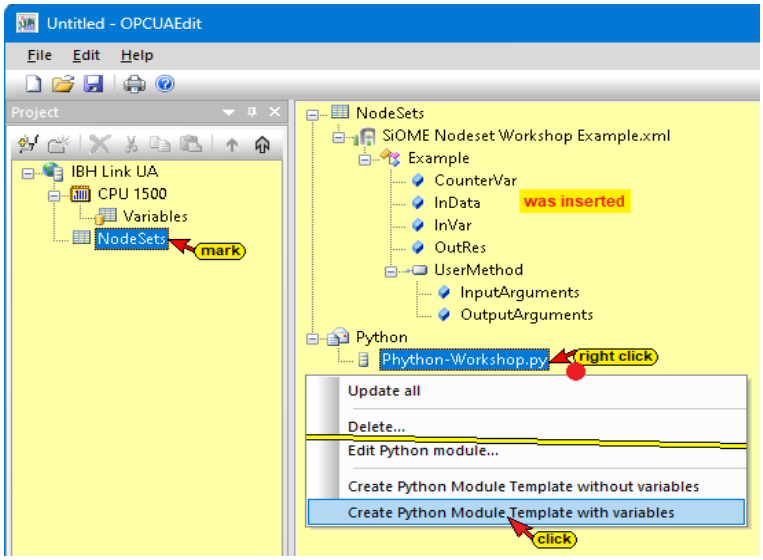
Add Python module.



Right-clicking on ***Python*** and clicking on the ***Add Python module…*** command in the opened context menu opens the dialog box for specifying the file name and storage location of the Python module.

The entry must be confirmed with ***OK***.

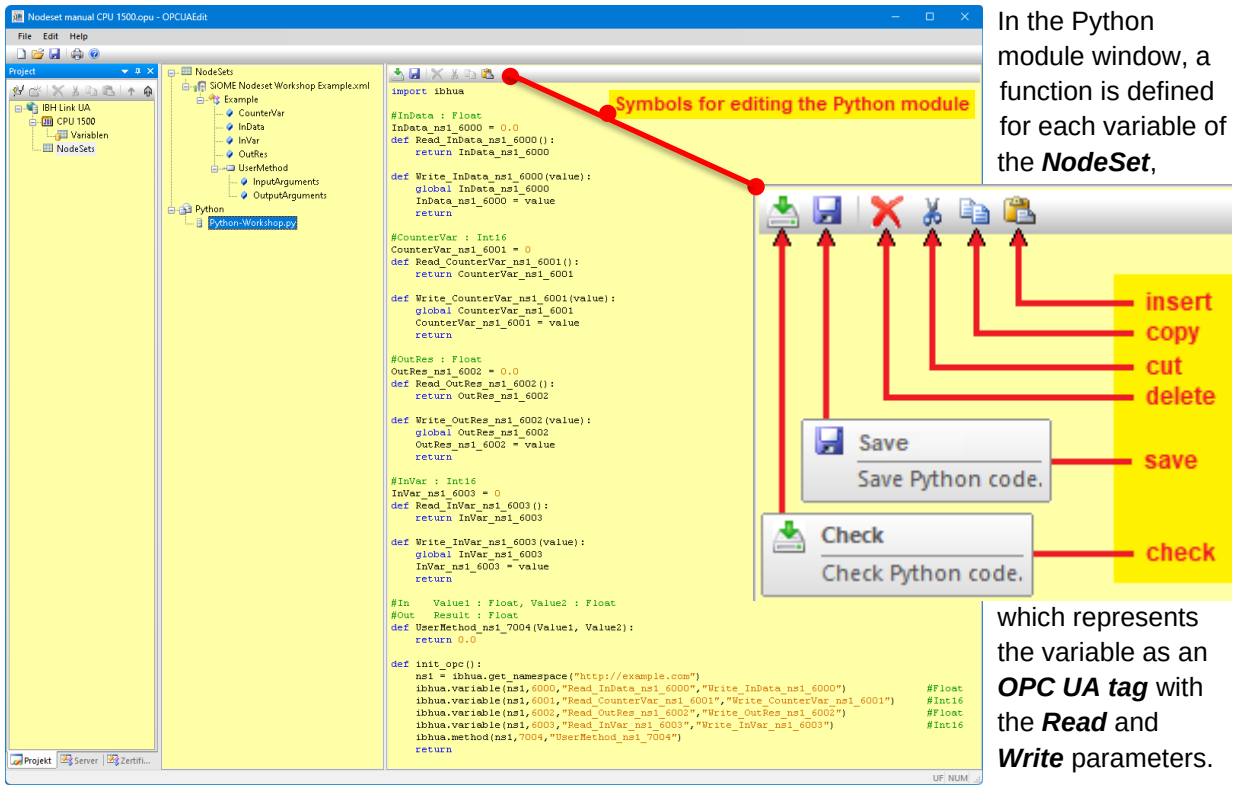
Create Python module template with variables.



Right-click on ***Python-Modul.py*** and a click on the ***Create Python module template with variables…*** command. The configuration is created based on the configuration created in the ***SiOME OPC UA Modeling Editor***.

The ***module template*** must be customized according to the requirements of the example. This ***Python program change*** can be carried out directly in the IBH OPC UA Editor or, more conveniently, in a ***Python programming system*** (e.g. Visual Studio Code from Microsoft®).

Python module with variables defined in NodeSet.



In the Python module window, a function is defined for each variable of the ***NodeSet***, which represents the variable as an ***OPC UA tag*** with the ***Read*** and ***Write*** parameters.

1.3 Existing functions in Python modules

```
import ibhua
```

With ***import ibhua*** the Python module of the IBH Link UA is called. A Python program to be executed in IBH Link UA starts with ***import ibhua***.

Module init_opc

```
def init_opc():
```

The ***"init_opc()"*** function is called for initialization of the IBH Link UA.

- **`ns1 = ibhua.get_namespace(name)`**

  ```
  ns1 = ibhua.get_namespace("http://example.com")
  ```

  The function returns the namespace number. As parameter ***(name)*** is the namespace name that was defined in the example in SiOME is entered.

- **`ibhua.variable(ns,id,"read function","write function")`**

  ```
  ibhua.variable(ns1,6000,"Read_InData_ns1_6000","Write_InData_ns1_6000")          #Float
  ibhua.variable(ns1,6001,"Read_CounterVar_ns1_6001","Write_CounterVar_ns1_6001")  #Int16
  ibhua.variable(ns1,6002,"Read_OutRes_ns1_6002","Write_OutRes_ns1_6002")          #Float
  ibhua.variable(ns1,6003,"Read_InVar_ns1_6003","Write_InVar_ns1_6003")            #Int16
  ```

  The function enables reading and writing of OPC variables. These variables are visibel in the ***UaExpert*** under the name of the example in the SiOME set namespace names listed. There is no data connection.

Parameter:

ns: namespace number
Id : Node name or numeric ID
read function: Function that is called when reading the variable. The function contains an output parameter but no input parameter.
write function: Function that is called when writing the variable. The function contains an input parameter and no output parameter.

These functions are automatically created for all variables present in the NodeSet.

- **`ibhua.method(ns,id,"function")`**

  ```
  ibhua.method(ns1,7004,"UserMethod_ns1_7004")
  ```

  The function allows setting methods. These variables are visibel in the ***UaExpert*** under the name of the example in the SiOME set namespace names listed. These functions are used for all methods present in the NodeSet created automatically.

Parameter:

ns : namespace number
Id : Node name or numeric ID
function: Name of the method that is called

- **`returns`**

  This is the termination of the module specified above.

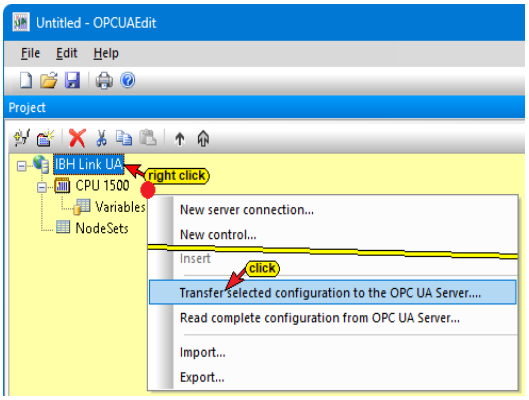
Read/Write module.

```
#InVar : Int16
InVar_ns1_6003 = 0
def Read_InVar_ns1_6003():
    return InVar_ns1_6003

def Write_InVar_ns1_6003(value):
    global InVar_ns1_6003
    InVar_ns1_6003 = value
    return
```

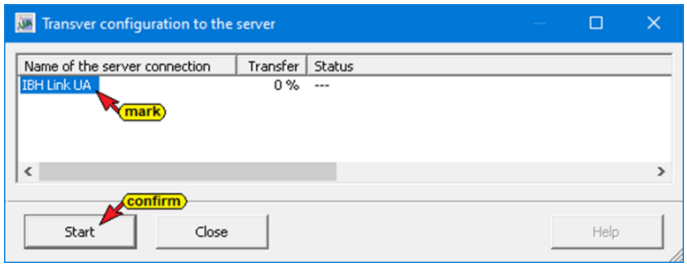
A ***Read/Write*** module is automatically created for each variable present in the NodeSet and declared in the Python module. Each module has a comment line as a heading with the variable name and the ***data type*** defined in SiOME.

1.4 Representation in the Unified Automation UaExpert program

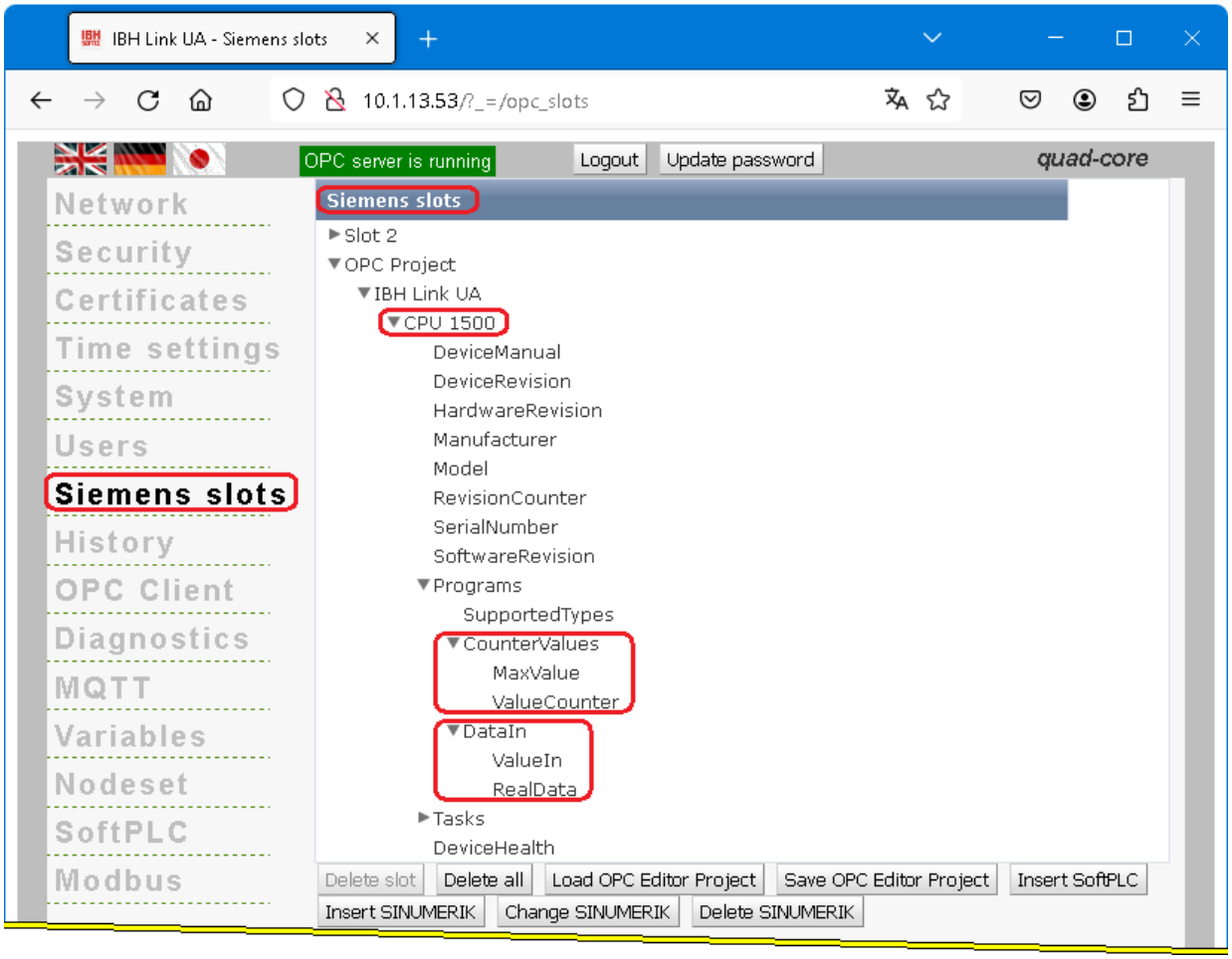
The Python module template can be checked using the ***UaExpert*** program (OPC Unified Architecture Client). To open ***UaExpert***, the configuration must be transferred to the OPC UA server.



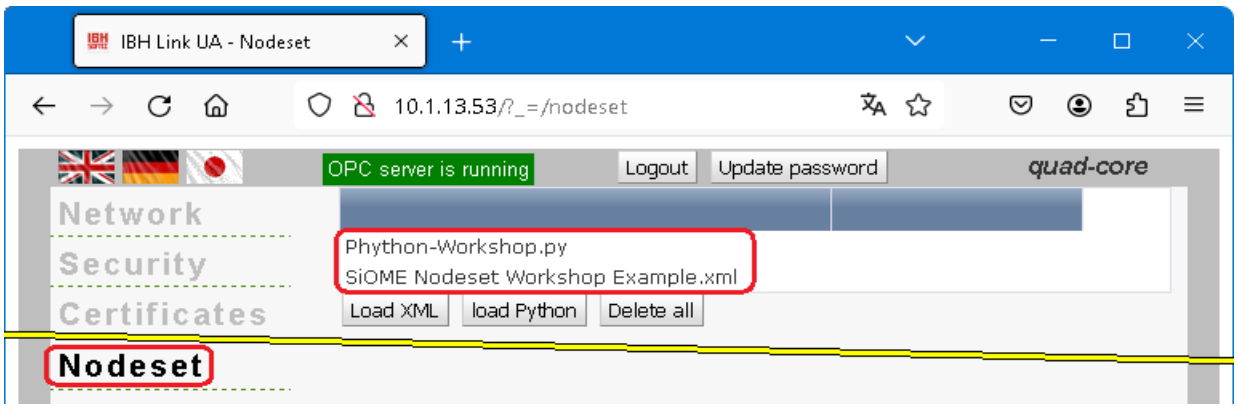
The command ***Transfer Selected Configuration to OPC UA Server…*** opens the ***Transfer configuration to the server*** dialog box. Select the server and then click Start.



If the configuration is transferred from the IBH OPC UA Editor to the IBH Link UA (OPC UA Server), the OPC tags present in the OPC UA server (displayed in the Siemens Slots window).

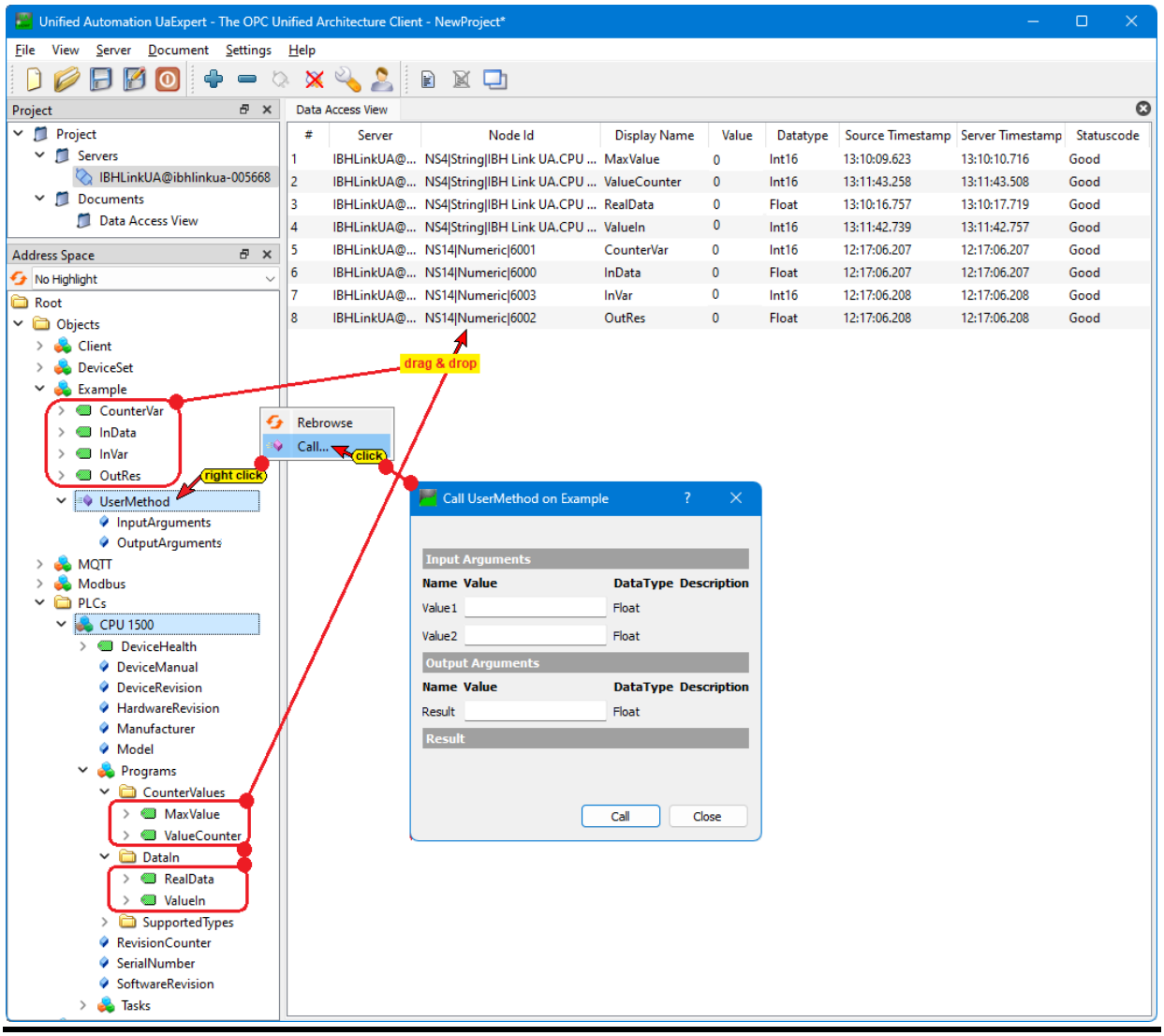


The contents of the transferred ***Nodeset*** and from the IBH OPC UA Editor is displayed in the IBH Link UA browser window ***Nodeset.***

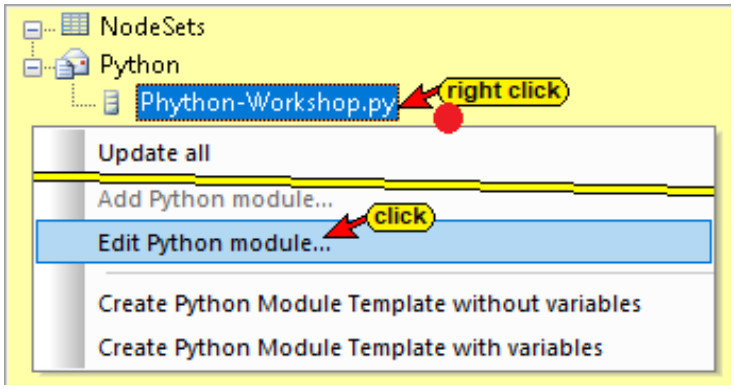


To display OPC UA Tags, the IBH Link UA must be registered in ***UaExpert*** as an OPC UA server.

If there is a connection between IBH Link UA and PLC CPUs, the OPC tags of the CPUs declared in the IBH OPC UA Editor also have the status ***Good***. Existing values are displayed and can be changed. The variables declared in the Read/Write Python module now have the status ***Good.***



1.5 Customize Python modules.

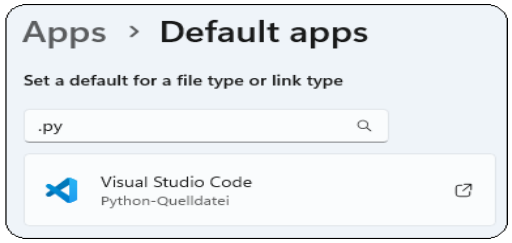


Editing can be carried out directly in the IBH OPC UA Editor.

With the following Windows adjustment, another Python editor (e.g. Visual Studio Code) can be opened.

Note:

To use the ***Edit Python Module…*** command to call an **external Python editor**, it must be set as the default app for the ***.py*** file type.



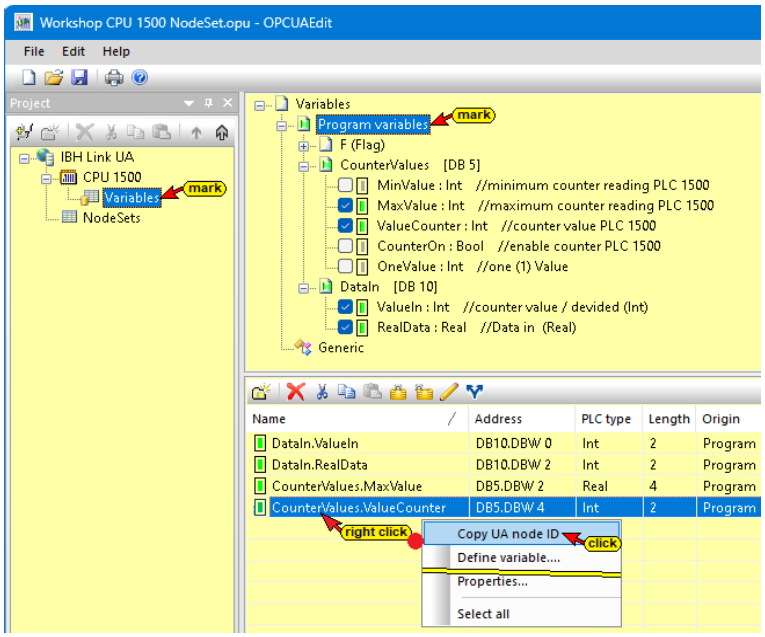
Note:
To open the Python module file in Visual Studio Code without errors, the file ***ibhua.pyi*** provided by IBHsoftec and the Python - IBH OPC UA editor file must be placed in the same folder.

1.6 Customize Python module file.

The Python modules created in the IBH OPC UA Editor must be adapted.

Copy NodeId to the Windows clipboard.

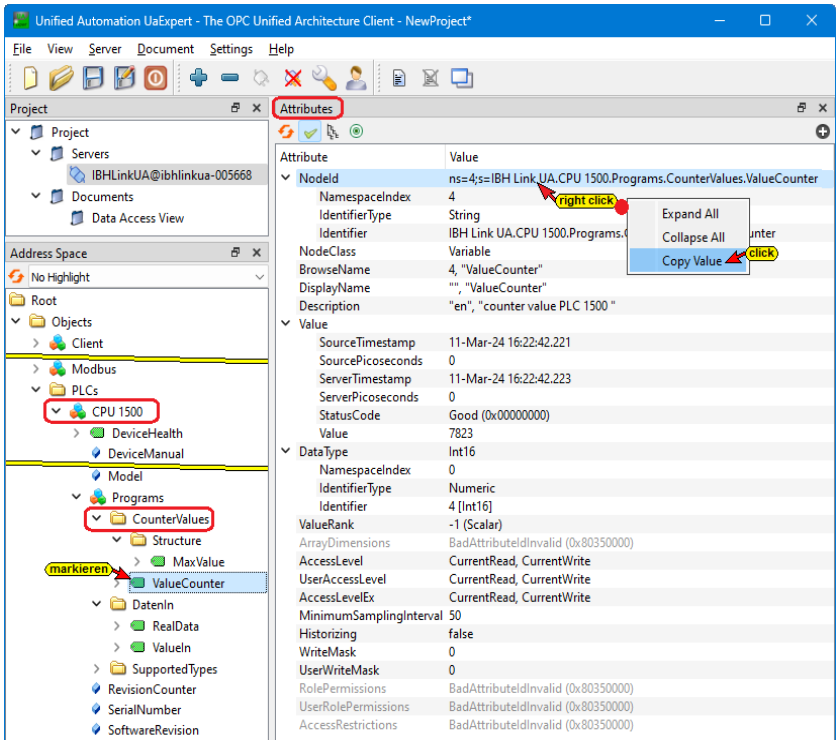
If ***Variables*** and ***Program variables*** are marked OPC tags are displayed. Right-clicking on the variable opens the context menu.



Clicking the command ***Copy UA node ID*** copies the ***ID*** to the Windows clipboard. That may be pasted into the Python statement.

In the example, the ***NodeId*** of the ***ValueCounter*** variable was copied. This procedure can be used to copy all ***NodeId*** required in Python commands.

Copy NodeId from UaExpert window attributes.



If the OPC variable configuration created in the OPC UA Editor was transferred to the IBH Link UA, the ***NodeId*** of an OPC variable can be copied in the window of the ***UaExpert*** program. If a variable is marked in the ***AddressSpace*** window, its ***NodeId*** can be copied in the Attributes window.

In the example, the ***NodeId*** of the ***ValueCounter*** variable was marked.

OPC variable Read from the CPU 1500

```
def init_opc():
```

The CPU 1500 variable (***DB5 – ValueCounter***), set as an OPC tag, should be read continuously by adapting the Python module template. Since only read/write modules for the variables were created in the IBH OPC UA editor automatically, the location of the variable that is to be monitored is declared in the Python module ***init_opc***. The following line should be added:

```
def init_opc():
    ns1 = ibhua.get_namespace("http://example.com")
    ibhua.variable(ns1,6000,"Read_InData_ns1_6000","Write_InData_ns1_6000")            #Float
    ibhua.variable(ns1,6001,"Read_CounterVar_ns1_6001","Write_CounterVar_ns1_6001")    #Int16
    ibhua.variable(ns1,6002,"Read_OutRes_ns1_6002","Write_OutRes_ns1_6002")            #Float
    ibhua.variable(ns1,6003,"Read_InVar_ns1_6003","Write_InVar_ns1_6003")              #Int16
    ibhua.method(ns1,7004,"UserMethod_ns1_7004")
    ibhua.monitor("ns=4;s=IBH Link UA.CPU 1500.Programs.CounterValues.ValueCounter","Monitor_var_Int16_always",1000,0,0)
    return
```

inserted

Ibhua.monitor("node", "Monitor Funktion", Intervall, Triggermode, Deadband)

- **`ibhua.monitor`**
  from ibhua something should be viewed continuously (text without spaces).
- **`"node"`**
  `"ns=4;s=IBH Link UA.SoftPLC416.Programs.CounterValues.ValueCounter",`
  directly after the quotation mark (**"**) the ***NodeId*** of the CPU 1500 Variable (***CounterValues [DB5] – ValueCounter***) is entered.

The ***NodeId*** of the variable can be copied directly from the OPC UA editor window using the command ***Copy UA node ID*** or copied from the ***Attributes*** window of the ***UaExpert*** program to the Windows clipboard to paste it into the Python statement. It is important to ensure that no additional spaces are added or removed. After the ***NodeId***, a quotation mark (**"**) must be placed, followed by a comma and further parameters.

- **`"Monitor function",`**
  `"Monitor_var_Int16_always",` write – without spaces.
- **`Interval,`**
  `1000` indicates that the value of the variable specified with ***nameset*** is all 1000 ms (1s) should be read.
- **`Trigger mode,`**
  `0` = is always triggered; **`1`** =Trigger when value changes
  `2` =Trigger on rising edge **`3`** =Trigger on falling edge
- **`Dead band`** determines the minimum value change that leads to the trigger.

Read variable write to another variable.

The read variable (***ValueCounter***) should be divided by ten (10) by adapting the Python module template and continuously transferred (Monitor) into the variable ***ValueIn (DataIn [DB10]***).

Ibhua.OPCWriteVar("node",var)

The Python module **`Monitor_var_Int16_always(var):`** is to be added:

```
def Monitor_var_Int16_always(var):
    myVar=var//10
    ibhua.OPCWriteVar("ns=4;s=IBH Link UA.CPU 1500.Programs.DataIn.ValueIn",myVar)
    return
```

`def Monitor_var_Int16_always(var):`
Defining the name of the Python module writing values continuously in the CPU 1500.

`myVar=var//10`
the adopted value is divided by ten (10) and stored as ***myVar***.

ibhua.OPCWriteVar

An OPV variable should be written in ibhua (text without spaces).

"node"

"ns=4;s=IBH Link UA.CPU 1500.Programs.DataIn.ValueIn",

The ***NodeId*** of the CPU 1500 variable is enclosed in quotation marks (") (***DataIn [DB10] – ValueIn***).

The ***NodeId*** of the variable can be copied directly from the OPC UA editor window using the ***UA node ID…*** command or copied from the ***Attributes*** window of the ***UaExpert*** program to the Windows clipboard to paste it into the Python statement. After the ***NodeId***, a quotation mark (") must be placed, followed by a comma and a parameter.

var

myVar The value of the variable is saved in the CPU 1500 variable (***DataIn [DB10] – ValueIn***).

Returns End of Python module.

Read/Write Module – Assign variable.

In the example, values are assigned to the individual modules so that they are available to OPC UA Clients (displayed in the ***UaExpert*** [OPC UA Client]).

#InData : Float

The value 98765.4 (data type float) is assigned to the ***InData*** variable.

Module adapted	Module created automatically

```
#InData : Float
InData_ns1_6000 = 0.0
def Read_InData_ns1_6000():
    InData_ns1_6000=98765.4
    return InData_ns1_6000

def Write_InData_ns1_6000(value):
    global InData_ns1_6000
    InData_ns1_6000 = value
    return
```

```
#InData : Float
InData_ns1_6000 = 0.0
def Read_InData_ns1_6000():
    return InData_ns1_6000

def Write_InData_ns1_6000(value):
    global InData_ns1_6000
    InData_ns1_6000 = value
    return
```

#CounterVar : Int16

The value of the variable ***MaxValue*** [CPU 1500, data block ***CounterValues*** (DB 5) (data type Int16)] is assigned to the variable ***CounterVar***.

Ibhua.OPCWriteVar("node")

- **ibhua.OPCWriteVar**

  An OPV variable should be written in ***ibhua*** (text without spaces).

- **"node"**

  "ns=4;s=IBH Link UA. CPU 1500.Programs.CounterValues. MaxValue",

  The ***NodeId*** of the CPU 1500 variable ***MaxValue*** is entered in quotation marks (").

The ***NodeId*** of the variable can be copied directly from the OPC UA editor or copied from the Attributes window of the ***UaExpert*** program to paste it into the Python statement.

Module adapted	Module created automatically

```
#CounterVar : Int16
CounterVar_ns1_6001 = 0
def Read_CounterVar_ns1_6001():
    CounterVar_ns1_6001=ibhua.OPCReadVar("ns=4;s=IBH Link UA.SoftPLC416.Programs.CounterValues.Structure.MaxValue")
    return CounterVar_ns1_6001
```

```
#CounterVar : Int16
CounterVar_ns1_6001 = 0
def Read_CounterVar_ns1_6001():
    return CounterVar_ns1_6001
```

Module adapted (no change)	Module created automatically

`def Write_CounterVar_ns1_6001(value):`
`  global CounterVar_ns1_6001`
`  CounterVar_ns1_6001 = value`
`  return`
	`def Write_CounterVar_ns1_6001(value):`
	`  global CounterVar_ns1_6001`
	`  CounterVar_ns1_6001 = value`
	`  return`

#OutRes : Float

If a value is assigned to the ***OutRes*** variable. This can be caused by changing the value in the Data Access View window in the ***UaExpert***. The ***Write_OutRes*** write command is used to write the contents of the ***OutVar*** variable into the ***RealData*** variable (***CPU 1500, DataIn [DB 10]***).

This only happens when the value in the ***OutRes*** variable changes.

In other words: If the ***OutRes*** value is changed in the ***UaExpert Data Access View***, the changed value appears in the ***RealData*** variable (***CPU 1500, DataIn [DB 10]***).

The ***NodeId*** of the variable can be copied directly from the OPC UA editor window using the ***UA node ID…*** command or copied from the ***Attributes*** window of the ***UaExpert*** to the Windows clipboard to paste it into the Python statement. After the ***NodeId***, a quotation mark (**"**) must be placed, followed by a comma and the variable name (***value***).

Module adapted	Module created automatically

`#OutRes : Float`
`OutRes_ns1_6002 = 0.0`
`def Read_OutRes_ns1_6002():`
`  return OutRes_ns1_6002`
 	
`def Write_OutRes_ns1_6002(value):`
`  global OutRes_ns1_6002`
`  ibhua.OPCWriteVar("ns=4;s=IBH Link UA.CPU 1500.Programs.DataIn.RealData",value)`
`  return`
	`#OutRes : Float`
	`OutRes_ns1_6002 = 0.0`
	`def Read_OutRes_ns1_6002():`
	`  return OutRes_ns1_6002`
	
	`def Write_OutRes_ns1_6002(value):`
	`  global OutRes_ns1_6002`
	`  OutRes_ns1_6002 = value`
	`  return`

#InVar : Int16

The variable ***InVar*** is created by a simple calculation task (data type Int16).

Module adapted	Module created automatically

`#InVar : Int16`
`InVar_ns1_6003 = 0`
`def Read_InVar_ns1_6003():`
`  x=20`
`  y=25`
`  z=500`
`  InVar_ns1_6003=x+y+z`
`  return InVar_ns1_6003`
 	
`def Write_InVar_ns1_6003(value):`
`  global InVar_ns1_6003`
`  InVar_ns1_6003 = value`
`  return`
	`#InVar : Int16`
	`InVar_ns1_6003 = 0`
	`def Read_InVar_ns1_6003():`
	`  return InVar_ns1_6003`
	
	`def Write_InVar_ns1_6003(value):`
	`  global InVar_ns1_6003`
	`  InVar_ns1_6003 = value`
	`  return`

UserMethod module

The two ***InputArguments Value1*** and ***Value2*** should be added in the inserted method. The result should be available under ***InputArguments Result***.

Module adapted	Module created automatically

`#In  Value1 : Float, Value2 : Float`
`#Out  Result : Float`
`def UserMethod_ns1_7004(Value1, Value2):`
`  Result=Value1+Value2`
`  return Result`
	`#In  Value1 : Float, Value2 : Float`
	`#Out  Result : Float`
	`def UserMethod_ns1_7004(Value1, Value2):`
	`  return 0.0`

The addition is carried out with the command line **`Result=Value1+Value2`**. Command line **`return Result`** displays the result.

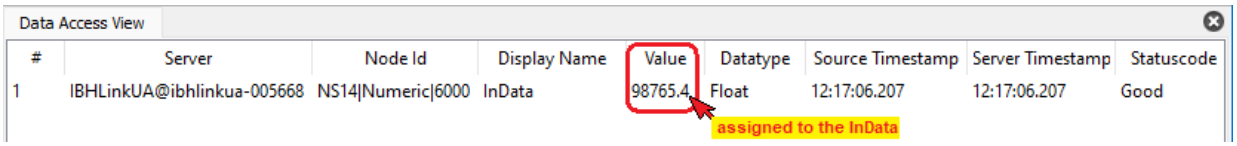
1.6.1 Customized Python modules

```
import ibhua

def Monitor_var_Int16_always(var):
    myVar=var//10
    ibhua.OPCWriteVar("ns=4;s=IBH Link UA.CPU 1500.Programs.DataIn.ValueIn",myVar)
    return

#InData : Float
InData_ns1_6000 = 0.0
def Read_InData_ns1_6000():
    InData_ns1_6000=98765.4
    return InData_ns1_6000

def Write_InData_ns1_6000(value):
    global InData_ns1_6000
    InData_ns1_6000 = value
    return

#CounterVar : Int16
CounterVar_ns1_6001 = 0
def Read_CounterVar_ns1_6001():
    CounterVar_ns1_6001=ibhua.OPCReadVar("ns=4;s=IBH Link UA.CPU 1500.Programs.CounterValues.MaxValue")
    return CounterVar_ns1_6001

def Write_CounterVar_ns1_6001(value):
    global CounterVar_ns1_6001
    CounterVar_ns1_6001 = value
    return

#OutRes : Float
OutRes_ns1_6002 = 0.0
def Read_OutRes_ns1_6002():
    return OutRes_ns1_6002

def Write_OutRes_ns1_6002(value):
    global OutRes_ns1_6002
    ibhua.OPCWriteVar("ns=4;s=IBH Link UA.CPU 1500.Programs.DataIn.RealData",value)
    return

#InVar : Int16
InVar_ns1_6003 = 0
def Read_InVar_ns1_6003():
    x=20
    y=25
    z=500
    InVar_ns1_6003=x+y+z
    return InVar_ns1_6003

def Write_InVar_ns1_6003(value):
    global InVar_ns1_6003
    InVar_ns1_6003 = value
    return

#In    Value1 : Float, Value2 : Float
#Out   Result : Float
def UserMethod_ns1_7004(Value1, Value2):
    Result=Value1+Value2
    return Result

def init_opc():
    ns1 = ibhua.get_namespace("http://example.com")
    ibhua.variable(ns1,6000,"Read_InData_ns1_6000","Write_InData_ns1_6000")              #Float
    ibhua.variable(ns1,6001,"Read_CounterVar_ns1_6001","Write_CounterVar_ns1_6001")      #Int16
    ibhua.variable(ns1,6002,"Read_OutRes_ns1_6002","Write_OutRes_ns1_6002")              #Float
    ibhua.variable(ns1,6003,"Read_InVar_ns1_6003","Write_InVar_ns1_6003")                #Int16
    ibhua.method(ns1,7004,"UserMethod_ns1_7004")
    ibhua.monitor("ns=4;s=IBH Link UA.CPU 1500.Programs.CounterValues.ValueCounter","Monitor_var_Int16_always",1000,0,0)
    return
```

1.7 Variable representation in UaExpert

OPC variable InVar

OPC variable InData



OPC variable CounterVar

Data Access View

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	IBHLinkUA@ibhlinkua-005668	NS14|Numeric|6001	CounterVar	9500	Int16	12:17:06.207	12:17:06.207	Good

CPU 1500 / CounterValues [DB5] MaxValue

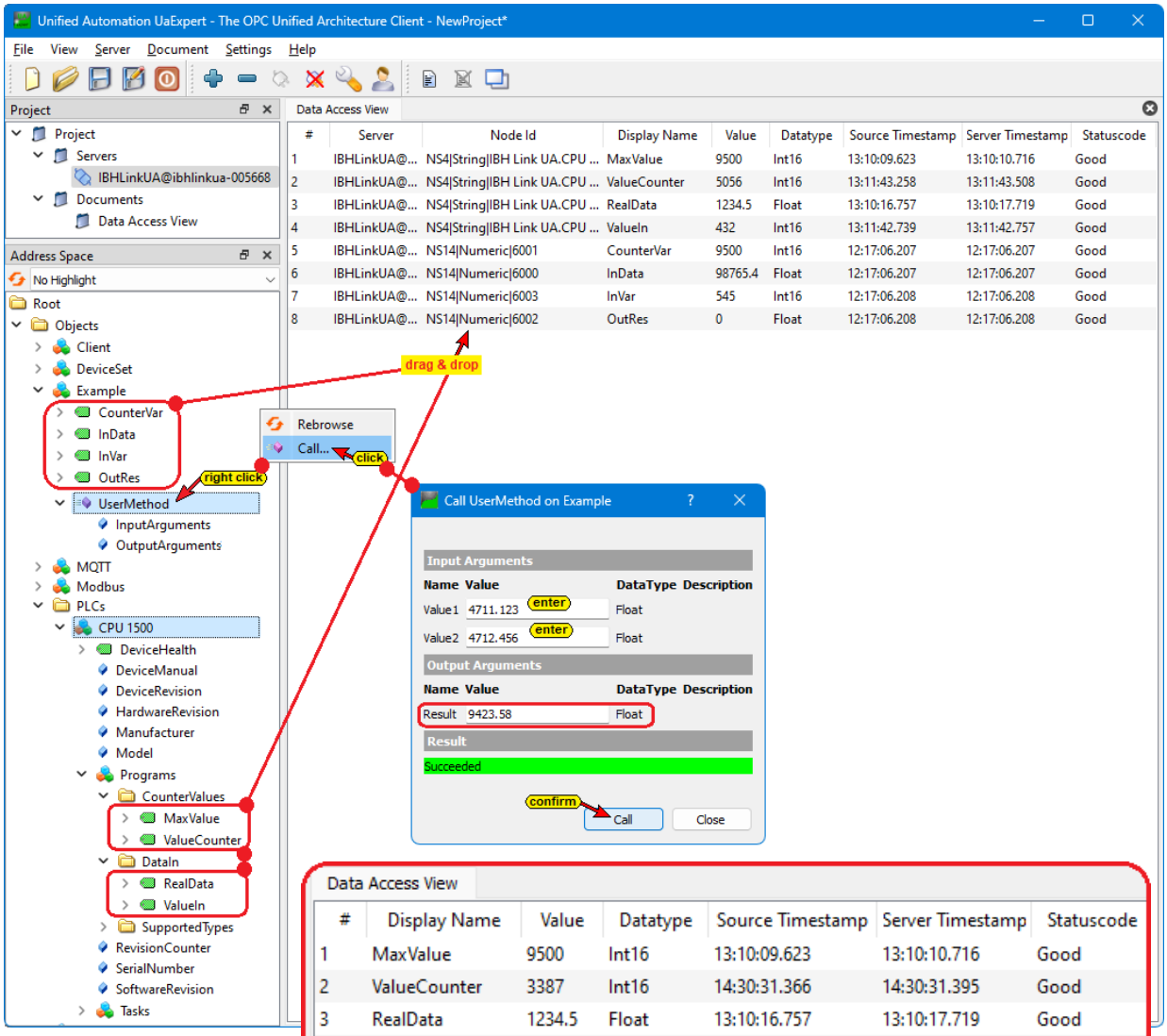
OPC variable OutRes

Data Access View

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	IBHLinkUA@ibhlinkua-005668	NS14|Numeric|6002	OutRes	1234.5	Float	12:17:06.208	12:17:06.208	Good

value assigned to OutRes

OPC variables



Data Access View

#	Server	Node Id	Display Name
1	IBHLinkUA@...	NS4|String|IBH Link UA.CPU 1500.Programs.CounterValues.MaxValue	MaxValue
2	IBHLinkUA@...	NS4|String|IBH Link UA.CPU 1500.Programs.CounterValues.ValueCounter	ValueCounter
3	IBHLinkUA@...	NS4|String|IBH Link UA.CPU 1500.Programs.DataIn.RealData	RealData
4	IBHLinkUA@...	NS4|String|IBH Link UA.CPU 1500.Programs.DataIn.ValueIn	ValueIn

1.8 Additional information

The Python module, equipped with OPC variables, is processed by the Python interpreter available in the IBH Link UA. Of course, more complex tasks can be implemented using the Python programming language, analyzing machine parameters and data to make predictions and recommendations for optimizing machine performance. Further information about Python/methods/data models is available in the IBHsoftec WIKI:

https://wiki.ibhsoftec.com/de/IBH_Link_UA:Python/Methoden/Datenmodelle

The packed file ***IBH Link UA - Python - Methods - Data Models.zip*** contains the following folders / files:

CPU 1500 TIA NodeSet	folder with TIA V18 programs Provision of the data block variables CPU 1500.
Workshop CPU 1500 NodeSet.opu	IBH OPC UA Editor configuration program.
SiOME Nodeset Manual Example.xml	Program for Siemens OPC UA Modeling Editor SiOME.
Python-Workshop.py	Python module for the IBH OPC UA editor.
ibhua.pyi	Python program for IBH OPC UA Editor Python modules to display in Microsoft Visual Studio error-free.
Nodeset Manual CPU 1500 Example.zip	zipped file of the files listed above.