



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

Windows[®] is a registered trademark of Microsoft[®] Corporation. TeamViewer[®] is a registered trademark of TeamViewer AG, Göppingen. Simatic[®] S5, Step[®] 5, Simatic[®] S7, Step[®] 7, S7-200[®], S7-300[®], S7-400[®], S7-1200[®]; S7-1500[®] and GRAPH[®] 5 are registered trademarks of Siemens Aktiengesellschaft, Berlin and Munich. Image source: [©] Siemens AG 2001, All rights reserved. Product names are trademarks of their respective owners.

Contents

Со	nten	tsI
1	IBH	I Link UA – Python / Methods / Data Models 1-1
	1.1	Creating an OPC UA information model1-1
		Set new namespace 1-1
		Creating an instance (Add Instance / Object)1-1
		Example object added – OPC UA Attributes
		Insert variable
		Insert variable Countervar
		Set input/output arguments
		Argument DataType and change Name
	1.2	IBH OPC UA Editor – Read in NodeSet preparation1-4
		OPC variable CPU 1500 1-5
		1.2.1 Add NodeSet configuration 1-5
		Add Python module1-6
		Create Python module template with variables
		Python module with variables defined in NodeSet
	1.3	Existing functions in Python modules1-7
		Module init_opc1-7
		Read/Write module 1-7
	1.4	Representation in the Unified Automation UaExpert program1-7
	1.5	Customize Python modules1-9
	1.6	Customize Python module file1-10
		Copy Nodeld to the Windows clipboard1-10
		Copy Nodeld from UaExpert window attributes
		OPC variable Read from the CPU 1500
		Read Variable write to another variable.
		#InData : Float
		#CounterVar : Int16
		#OutRes : Float 1-13
		#InVar : Int16 1-13
		UserMethod module1-13
		1.6.1 Customized Python modules 1-14
	1.7	Variable representation in UaExpert1-14
		OPC variables 1-15
	1.8	Additional information1-16
		Configuration exampleII

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.

lease add or choose a namespace	 Select editing namespace (expand/select) 	In order not to work with the stor Foundation (http://opcfoundation namespace is defined.	ed namespac n.org/ua/), a se	e of the OPC eparate
Add Namespace		The namespace is defined in the Identifier) standard, an arbitrary domain <i>.com</i> .	e URI (Uniforn name (examp	n Resource cle) with the
http://example		🧱 Siemens OPC UA Modeling Editor 2.8.0		
Version 1.00		= SIEMENS		
		📑 🔁 🖬 🛛 🖶 🖻 🔍 🛄 🛛 Nar	mespaces: Editing (c	urrent) 1: http://example.com
PublicationDate	÷	Information model http://example.com -		
2024-01-08T13:43:57+01:00		No details 👻 No filter 👻	OPC UA Attrib	outes
		▼ OPC Root	Nodeld	ns=0;i=85
	antim	Objects	NodeClass	Object
	Comm	 OP¢ Aliases 	BrowseName	Objects
	Cancel OK	 Server 	DisplayName	Objects
		► OPC Types	Description	null
		OP¢ Views	WriteMask	0
			UserWriteMask	0
			RolePermissions	

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.



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 *Valueln* (*Dataln / 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** (*Dataln / DB 10*). A *UserMethod* with a calculation is also *inserted*.

Insert variable CounterVar.



IBH Link UA Workshop

OPC Views

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 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).

*

Save as



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.

IBH OPC UA Editor – Read in NodeSet preparation. 1.2

Name of the server connection

Name

Port URL Inverse connection Security settings Security policity Message mode

Login Session name

Server address

Authentication

Host name / Address

1. Open IBH OPC UA Editor.

M Untitled - OPCUAEdit File Edit Help 🗋 💕 🛃 🖨 🎯

💅 🖆 🗙 👗 🛍 🛝 🛧 🖗

Terma

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*.

IBH Link UA

10.1.13.53

None None

Anonymous

opc.tcp://10.1.13.53:48010 N&o





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.

🥨 Select program			×				
CPU 1500 TIA Nodeset							
PFad: A:\NodeSet Manual CPU 1500\CPU 1500 TIA Nodeset\							
OK Cancel	(Help					

S7-1500 TCP/IP

OPC variable CPU 1500

🐱 Nodeset manual CPU 1500.opu - OPCUAEdit — 📃 🔍										×			
File Edit Help													
🗋 💕 🔙 🕼 🎯													
Project 👻 🕈 🗙	🖃 🗋 Variables												
IBH Link UA IBH Link UA	Contervalues [DB 5] CounterValues [DB 5] MinValue: Int //minimum counter reading PLC 1500 MinValue: Int //minimum counter reading PLC 1500 MinValue: Int //counter value PLC 1500 OPC-Tag) OneValue: Int //counter Value PLC 1500 OneValue: Int //counter Value / devided (Int) OneValue: Int //counter Value / devided (Int) RealData: Real //Data in (Real) Seneric												
	🗳 🗙 X b B ě b /	2 19											
	Name Address PLC type Length Origin Access OPC type L U Comment												
PLC 1500	CounterValues.MaxValue DB5.DBW 2 Int 2 Program RW Int16 0 0 maximum countr									unter rea	ading PL	C 1500	
OPC tags	CounterValues.ValueCounter	DB5.DBW 4	Int	2	Program	RW	Int16	0	0	counter value	PLC 150	00	
	Dataln.Valueln DB10.DBW 0 Int 2 Program RW Int16 0 0 counter value / devided (Int)												
Roject Server Server Certifi	📘 Datain.RealData	DB10.DBW 2	Real	4	Program	RW	Float	0	0	Data in (Real)			
												CAPS	NUM

1.2.1 Add NodeSet configuration.

M Untitled - OPCUAEdit	Right-clicking on IBH Link UA and clicking on the Add NodeSet	
<u>F</u> ile <u>E</u> dit <u>H</u> elp	Configuration command. A dialog	
🗋 💕 🛃 🕼 🞯		box opens.
Project		In the example, the node set
学 💣 🗙 🔉 🖻 🛍 🛧 🏟		 configuration should be created from the <i>SiOME</i> editor.
E-11 IBH Link UA		IBH OPC UA Editor
Variables New server co	onnection	Take over current Nodeset configuration from server IBH Link
Add CNC stat	ion	confirm
is inserted Add Rockwell	configuration	Yes No
Add Nodeset	configuration	Confirm No in the dialog box.
Delete	Click	
Cut		
Export		4
Untitled - OPCUAEdit		If <i>NodeSet</i> is selected in the right
File Edit Help		window, <i>NodeSet</i> and <i>Python</i> are
Project V A X ModeSets		displayed in the left window.
Python Update all)	Right-clicking on <i>NodeSet</i> and
BH Link UA	Click	clicking on the Add NodeSet
Add nodese	t	command in the opened context
Edit nodese	t	menu. In the Add Nodeset dialog box
Add Python Edit Python	select the Nodeset configuration from	
Create Pytho	on Mo	
Create Pyth	Don More Look jn: DodeSet Manual CPI	2U 1500 🗸 🚱 🎓 📴 🖽 🕈
The .xml file SiOME Nodes	A Name	Date modified Type Attributes Size
	Home SiOME Nodeset Wo	leSet 11-Mar-2410:13 File folder D orkshop Example.xml 03-Mar-2418:48 XML File N 14 KB
adopted.	Home SiOME Nodeset Wo	leSet 11-Mar-24 10:13 File folder D orkshop Example.xml 0.047-24 1848 XML File N 14 KB (mark)

Add Python module.



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.



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
ld :	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.

Page 1-8

IBH Link UA - Python / Methods / Data Models

🚺 Untitled - OPCUAEdit		The command Transfer Selected Configuration	on to	
<u>F</u> ile <u>E</u> dit <u>H</u> elp		OPC UA Server opens the Transfer configu	ratio	n to
🗋 💕 🔒 🖨 📀		the server dialog box. Select the server and the	n clic	k
Project				ĸ
🖗 💣 🗙 🎖 🖻 🖻	† ŵ	Start.		
BH Link UA	ight click)	🐱 Transver configuration to the server 🛛 🚽		\times
📄 🛄 CPU 1500 🍎				
🔤 🖓 🖓 🖓 🖓	New server connection	Name of the server connection Transfer Status		
NodeSets	New control	IBH Link UA 0 %		
-	Insert	_ V(mark)		
	Transfer celected configuration to the OPC IIA Server			
	Read complete configuration from OPC UA Server			>
		confirm		
	Import			1
	Export	Start Close	Help	

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*.

USH I	IBH Lin	k UA -	Nodese	et .	×	+								~				
$\leftarrow \rightarrow$	С	습		0 8	▲ 10.	1.13	1.53/?_=	=/nod	eset				ふ	☆	◙	۲	பி	≡
	MC P	0		OP) serve	risı	running		L	ogout	Upd	ate pas:	sword		qL	iad-c	core	
Net	wor	k																
Sec	urit	у		PI Si	nython OME N	-Wo ode	orkshop set Wo	o.py orksho	op Ex	ample	.xml							
Cer	tific	ate	es :	L	oad XM	L	load P _i	/thon	Del	lete all								
No	lese	t											=	_	 		_	

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.

Apps > Default apps Set a default for a file type or link type		Note: To open the Python module file in Visual Studio Code without errors, the file <i>ibhua.pyi</i> provided by
.py Q Visual Studio Code Python-Quelldatei	Ø	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 Nodeld 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 *Nodeld* of the *ValueCounter* variable was copied. This procedure can be used to copy all *Nodeld* required in Python commands.

Copy Nodeld from UaExpert window attributes.

Unified Automation UaExpert - The OPC Unified Automation UaExpert -					
<u>File View Server D</u> ocument <u>Settings</u>	Help				
🗋 🥟 🕞 🗭 🧿 🔶 🗖	🛛 🗙 🔦 🧏 🖹 🖹				
Project 🗗 🛪	Attributes				đΧ
✓ 📁 Project	🗲 🧹 🤖 🛞				C
✓	Attribute	Value			
IBHLinkUA@ibhlinkua-005668	× Nodeld	ns=4:s=IBH Link LIA CPU 1500 Progra	ams CounterValue	s ValueCo	unter
Documents	NamespaceIndex	4	anis.countervalue		unter
Data Accord View	IdentifierType	String	Expand All		
Data Access view	Identifier	IBH Link UA.CPU 1500.Programs.0	Cellanse All	unter	
Address Space 🗗 🗙	NodeClass	Variable	Conapse All	click	
	BrowseName	4. "ValueCounter"	Copy Value 🚄		
🖌 No Hignlight 🗸 🗸	DisplayName	"", "ValueCounter"			
D Root	Description	"en", "counter value PLC 1500 "			
🗸 🚞 Objects	Value				
> 🚓 Client	SourceTimestamp	11-Mar-24 16:22:42.221			
A Madhur	SourcePicoseconds	0			
> 🙀 Modbus	ServerTimestamp	11-Mar-24 16:22:42.223			
V D PLCs	ServerPicoseconds	0			
🗸 😽 CPU 1500	StatusCode	Good (0x0000000)			
> 🕘 DeviceHealth	Value	7823			
DeviceManual	✓ DataType	Int16			
Model	NamespaceIndex	0			
	IdentifierType	Numeric			
	Identifier	4 [Int16]			
Countervalues	ValueRank	-1 (Scalar)			
✓ □ Structure	ArrayDimensions	BadAttributeldInvalid (0x80350000)			
(markieren 💊 🔌 MaxValue	AccessLevel	CurrentRead, CurrentWrite			
ValueCounter	UserAccessLevel	CurrentRead, CurrentWrite			
✓	AccessLevelEx	CurrentRead, CurrentWrite			
> 🔲 RealData	MinimumSamplingInterval	50			
> Valuelo	Historizing	talse			
> 🔤 Valdeni	WriteMask	0			
> upported types	UserWriteMask	U De d'Attribute I d'avail d'Avances (Compassione)			
RevisionCounter	KolePermissions	BadAttributeidinvalid (0x80350000)			
SerialNumber	UserKolePermissions	BadAttributeidinvalid (0x80350000)			
SoftwareRevision	ACCESSIVESTRICTIONS	DauAttributeidinvalid (UX80350000)			

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

In the example, the *Nodeld* 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:

<pre>def init opc():</pre>	
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
<pre>ibhua.variable(ns1,6002,"Read_OutRes_ns1_6002","Write_OutRes_ns1_6002")</pre>	#Float
ibhua.variable(ns1,6003,"Read_InVar_ns1_6003","Write_InVar_ns1_6003")	#Int16
<pre>ibhua.method(ns1,7004,"UserNethod_ns1_7004")</pre>	
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 **Nodeld** of the CPU 1500 Variable (**CounterValues [DB5] – ValueCounter**) is entered.

The **Nodeld** 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 **Nodeld**, a quotation mark (") must be placed, followed by a comma and further parameters.

٠	"Monitor function	on",	
	"Monitor	_var_Int16_always",	write – without spaces.
•	Interval,		
	1000	indicates that the value of th	e 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 *Valueln (Dataln [DB10]*).

Ibhua.OPCWriteVar("node",var)

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

```
def Monitor_var_Intl6_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 Nodeld of the CPU 1500 variable is enclosed in quotation marks (") (DataIn
[DB10] - ValueIn).
```

The **Nodeld** 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 **Nodeld**, 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 (Dataln
	[DB10] – Valueln).

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		
<pre>#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</pre>	<pre>#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</pre>		

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

The **Nodeld** 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
<pre>#CounterVar : Int16</pre>	#CounterVar : Int16
CounterVar_ns1_6001 = 0	CounterVar_ns1_6001 = 0
def Read_CounterVar_ns1_6001():	def Read_CounterVar_ns1_6001():
CounterVar_ns1_6001=ibhua.OPCReadVar("ns=4;s=IBH Link	return CounterVar_ns1_6001
return CounterVar_ns1_6001	x UA.SoftPLC416.Programs.CounterValues.Structure.MaxValue")

Module adapted (no change)			Module created automatically
	<pre>def Write_CounterVar_ns1_6001(value): global CounterVar_ns1_6001 CounterVar_ns1_6001 = value return</pre>		<pre>def Write_CounterVar_ns1_6001(value): global CounterVar_ns1_6001 CounterVar_ns1_6001 = value return</pre>

#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, Dataln [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 **Nodeld** 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 **Nodeld**, a quotation mark (") must be placed, followed by a comma and the variable name (**value**).

Module adapted	Module created automatically		
<pre>#OutRes : Float OutRes_ns1_6002 = 0.0 def Read_OutRes_ns1_6002(): return OutRes_ns1_6002</pre>	<pre>#OutRes : Float OutRes_ns1_6002 = 0.0 def Read_OutRes_ns1_6002(): return OutRes_ns1_6002</pre>		
	<pre>def Write_OutRes_ns1_6002(value): global OutRes_ns1_6002</pre>		
def Write_OutRes_ns1_6002(value): global OutRes_ns1_6002	OutRes_ns1_6002 = value return		
ibhua.OPCWriteVar("ns=4;s=IBH Link UA.CPU 1500.Programs.DataIn.RealData return			

#InVar : Int16

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

Module adapted	Module created automatically
<pre>#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</pre>	<pre>#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</pre>

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		
<pre>#In Value1 : Float, Value2 : Float #Out Result : Float def UserMethod_ns1_7004(Value1, Value2): Result=Value1+Value2 return Result</pre>	<pre>#In Value1 : Float, Value2 : Float #Out Result : Float def UserMethod_ns1_7004(Value1, Value2): return 0.0</pre>		

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

▲ 🛛 🗙 🖇 哈	
import ibhua	
<pre>def Monitor_var_Int16_always(var): myVar=var//10 ibhua.OPCWriteVar("ns=4;s=IBH Link UA.CPU 1500.Programs.DataIn.ValueIn",myVar) return</pre>	
<pre>#InData : Float InData_ns1_6000 = 0.0 def Read_InData_ns1_6000(): InData_ns1_6000=98765.4 return InData_ns1_6000</pre>	
<pre>def Write_InData_ns1_6000(value): global InData_ns1_6000 InData_ns1_6000 = value return</pre>	
<pre>#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.CounterVar_ return_CounterVar_ns1_6001</pre>	alues.MaxValue")
<pre>def Write_CounterVar_ns1_6001(value): global CounterVar_ns1_6001 CounterVar_ns1_6001 = value return</pre>	
#OutRes : Float	
OutRes_ns1_6002 = 0.0 def Read_OutRes_ns1_6002(): return_OutRes_ns1_6002	
<pre>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</pre>	
<pre>#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</pre>	
<pre>def Write_InVar_ns1_6003(value): global InVar_ns1_6003 InVar_ns1_6003 = value return</pre>	
<pre>#In Value1 : Float, Value2 : Float #Out Result : Float def UserMethod_ns1_7004(Value1, Value2):</pre>	
<pre>def init_opc(): ns1 = ibhua.get_namespace("http://example.com") ibhua.variable(ns1,6000,"Read_InData_ns1_6000","Write_InData_ns1_6000") ibhua.variable(ns1,6001,"Read_CounterVar_ns1_6001","Write_CounterVar_ns1_6001") ibhua.variable(ns1,6002,"Read_OutRes_ns1_6002,""Write_OutRes_ns1_6002") ibhua.variable(ns1,6003,"Read_InVar_ns1_6003","Write_InVar_ns1_6003") ibhua.method(ns1,7004,"UserNethod_ns1_7004") ibhua.monitor("ns=4;s=IBH_Link_UA.CPU_1500.Programs.CounterValues.ValueCounter","Mon return</pre>	<pre>#Float #Int16 #Float #Int16 nitor_var_Int16_always",1000,0,0)</pre>

1.7 Variable representation in UaExpert

OPC variable InVar

Data	Data Access View							8
#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	IBHLinkUA@ibhlinkua-005668	NS14 Numeric 6003	InVar	545	Int16	12:17:06.208	12:17:06.208	Good
		InV	ar = 20 + 25 + 500	•				
OPC variable InData								
Data /	Access View							0
#	Server	Node ld	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	IBHLinkUA@ibhlinkua-005668	NS14 Numeric 6000	InData	98765.4	Float	12:17:06.207	12:17:06.207	Good
					assigned t	o the InData		

1

Good

OPC variable CounterVar

IBHLinkUA@ibhlinkua-005668 NS14|Numeric|6002 OutRes

Data A	Access View							8
#	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 / Co	unterValues [DB5] N	/laxValue	
OPC variable OutRes								
Data Access View								
#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode

1234.5. Float

12:17:06.208

ue assigned to OutRes

12:17:06.208

OPC variables

🖉 Unified Automation UaExpert - The OPC Unified Architecture Client - NewProject* - 🗆 X											
<u>File View Server D</u> ocument <u>S</u> ettings <u>H</u> elp											
D 💋 🕞 🗭 🧿 💠 🖛 🖄 💥 🤮 🖹 🖹 🔟											
Project & ×	Data A	ccess View									0
✓	#	Server	Node	e ld	Display Name	Value	Datatype	Source Timestamp	Server Timestam	np Statusc	ode
✓	1	IBHLinkUA@	NS4 String IBH L	ink UA.CPU	MaxValue	9500	Int16	13:10:09.623	13:10:10.716	Good	
IBHLinkUA@ibhlinkua-005668	2	IBHLinkUA@	NS4 String IBH L	ink UA.CPU	ValueCounter	5056	Int16	13:11:43.258	13:11:43.508	Good	
Documents Data Access View	3	IBHLinkUA@	NS4 String IBH L	ink UA.CPU	RealData	1234.5	Float	13:10:16.757	13:10:17.719	Good	_
	4	IBHLinkUA@	NS4 String IBH L	.ink UA.CPU	Valueln	432	Int16	13:11:42.739	13:11:42.757	Good	
Address Space 🗗 🗙	5	IBHLinkUA@	NS14 Numeric 6	001	CounterVar	9500	Int16	12:17:06.207	12:17:06.207	Good	
😏 No Highlight 🗸 🗸	0	IBHLinkUA@	NS14 Numeric 6	000	InData	98/65.4	Float	12:17:06.207	12:17:06.207	Good	
Contemporaries and the second	6	IBHLINKUA@	NS14 Numeric 0	003	invar OutBaa	0	Int Io	12:17:06:208	12:17:00.208	Good	
V 🗀 Objects	•	IDFILINKUA@	NS 14 Numeric 0	002	Outres	0	FIDAL	12:17:00.206	12:17:00.200	0000	
> 💫 Client		dr	ag & drop								
> 🙀 Deviceset			1								
			/								
> InData	Rebro	wse									
> 🕥 InVar 📃	Call										
OutRes											
✓ =♥ UserMethod		/ T	📕 Call UserMet	hod on Exampl	le ?	×					
InputArguments											
OutputArguments		Toronk Augusto			_						
> 💑 MQTT			Input Argumen	its	Dete Tree Dee	- in the -					
> 🙀 Modbus			Name value	(enter)	DataType Des	cription					
✓ ▲ CPU 1500			Value1 4/11.123	(enter)	Float						
> DeviceHealth	/		Value2 4712.456		Float						
DeviceManual			Output Argume	ents							
DeviceRevision			Name Value		DataType Des	cription					
HardwareRevision	/	1	Result 9423.58		Float						
Manufacturer			Result								
V A Bragrams			Succeeded								
V Counted/aluer											
A maxValue Confirm											
> ValueCounter											
Ver Dataln								~			
> RealData Data Access View							1				
> Caluein		# Disp	lay Name	Value	Datatype	Source	e Timestar	np Server Tin	nestamp S	tatuscod	e
 RevisionCounter 	1	MaxV	alue	9500	Int16	13:10:0	9.623	13:10:10.7	'16 Go	od	
SerialNumber SoftwareRevision	2	Value	Counter	3387	Int16	14:30:3	31.366	14:30:31 3	95 Go	od	
> 💑 Tasks	2	RealD	ata	1224 5	Float	12,10,1	6 757	12,10,17.7	10 6-	od	
		Valuel		1204.0	Lat 16	14-20-2	1 260	14-20-21-2	05 00	ou a	Ē
	4	valuel	n	300	INTIO	14:30:3	01.209	14:30:31.3	90 60	od	

Data A	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	Valueln

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.