



IBH Link UA Python / Methoden / Datenmodelle

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® ist ein eingetragenes Warenzeichen der Microsoft® Corporation. TeamViewer® ist ein eingetragenes Warenzeichen der TeamViewer AG, Göppingen. Simatic® S5, Step® 5, Simatic® S7, Step® 7, S7-200®, S7-300®, S7-400®, S7-1200®; S7-1500®, SiOME und GRAPH® 5 sind eingetragene Warenzeichen der Siemens Aktiengesellschaft, Berlin und München. Bildquelle: © Siemens AG 2001, Alle Rechte vorbehalten. Produktnamen sind Warenzeichen ihrer Hersteller.

Inhalt

Inha	alt	I
1	IBH	Link UA – Python / Methoden / Datenmodelle 1-1
	1.1	OPC UA Informationsmodell erstellen1-1
		Neuen Namespace festlegen 1-1
		Instanz erstellen (Add Instance / Object) 1-1
		Objekt Example eigefügt – OPC UA Attributes 1-1
		Variable einfügen 1-2
		Variable CounterVar einfügen 1-2
		Methode erstellen (Add Instance / Method) 1-2
		Input / Output Argumente festlegen 1-3
		Argument Data lype und Name andern 1-3
	1.2	IBH OPC UA Editor – Vorbereitung NodeSet einlesen1-4
		OPC-Variable CPU 15001-4
		1.2.1 NodeSet-Konfiguration hinzufügen 1-5
		Python-Modul hinzufügen1-5
		Python-Modulvorlage mit Variablen erstellen
		Python-iviodul mit im NodeSet definierten Variablen
	1.3	Vorhandene Funktionen in Python-Modulen1-6
		Modul init_opc 1-6
		Read / Write Modul 1-7
	1.4	Python-Modulvorlage überprüfen1-7
		Konfiguration an OPC UA Server übertragen
		IBH Link UA Browser-Fenster Siemens Slots1-8
		IBH OPC UA Editor (Fenster Server)
		NotSet -Konfiguration-Variable anzeigen
		UaExpert – Data Access View
	1.5	Python-Module anpassen
	1.6	Python-Moduldatel anpassen1-10
		Kopieren der Nodeld in die Windows-Zwischenablage
		Nodeld aus UaExpert-Attribute-Fenster kopieren
		Gelesen Variable in eine andere Variable schreiben 1-10
		Read/Write Module – Variable zuordnen
		#InData : Float 1-12
		#CounterVar : Int16 1-12
		#OutRes : Float 1-12
		#InVar : Int16 1-13
		Modul UserMethod 1-13
		1.6.1 Python-Module angepasst 1-14
	1.7	IBH OPC UA Editor (Fenster Server)1-14
		NotSet -Konfiguration-Variable anzeigen1-14
	1.8	Variable in UaExpert1-15
	1.9	Weiterführende Informationen1-16

Datei IBH Link UA – Python - Methoden - Datenmodelle.zip beinhaltet folgende Ordner / Dateien:

CPU 1500 TIA NodeSet	Ordner mit Bereitstellung der Datenbaustein Variablen der TIA V19 Programmen zur CPU 1500. IP-Adresse CPU 1500: 10.1.13.53
Workshop CPU 1500	IBH OPC UA Editor Konfigurations-Programm.
NodeSet.opu	IP-Adresse IBH Link UA: 10.1.13.53
SiOME Nodeset Manual Example.xml	Programm für den Siemens OPC UA Modeling Editor SiOME.
Python-Workshop.py	Python-Modul für den IBH OPC UA Editor.
ibhua.pyi	Python-Programm, um IBH OPC UA Editor Python- Module in Visual Studio Code fehlerfrei anzuzeigen.
Nodeset Manual CPU 1500.zip	gepackte Datei der oben aufgeführten Dateien.

1 IBH Link UA – Python / Methoden / Datenmodelle

In dem folgenden Beispiel wird ein einfaches OPC UA Informationsmodell mit dem Siemens OPC UA Modeling Editor (SiOME) erstellt. Zu der erstellte Nodset-Konfigurationsdatei wird mit Hilfe des IBH OPC UA Editors ein Python-Programm zugeordnet und in den IBH Link UA übertragen.

Die Methode (Python-Programm) liest einen Wert aus einer CPU 1500 modifiziert diesen und legt ihn in einem anderen Datenbaustein der CPU 1500 ab. Die Handhabung von Variablen wird an Beispielen gezeigt.

1.1 OPC UA Informationsmodell erstellen.

Select editing name

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



Neuen Namespace festlegen

Please add or choose a namespace



Um nicht mit dem hinterlegten Namespace der OPC Foundation (http://opcfoundation.org/ua/) zu arbeiten wird ein eigener Namespace festgelegt.

Der Namespace wird im **URI** (Uniform Resource Identifier) Standard, einem beliebigen Namen **(example)** mit der Domain **.com** festgelegt

= SIEMENS							
📑 🖻 🖬 🛛 🖶 🖶 🔍 🚺	amespaces: Editing (current) 1: http://example.c					
Information model http://example.com							
No details 🔹 No filter	OPC UA Attri	butes					
V DPC Root	Nodeld	ns=0;i=85					
Objects	NodeClass	Object					
 OP¢ Aliases 	BrowseName	Objects					
 Server 	DisplayName	Objects					
 OP¢ Types 	Description	null					
op¢ Views	WriteMask	0					
	UserWriteMask	0					
	BolePermissions						

Instanz erstellen (Add Instance / Object)

Mit einem Rechtsklick auf **Objects** und einem Klick auf den Befehl **Add Instance** im geöffneten Kontextmenü wird das Dialogfeld Add Instance geöffnet. Den Namen **Example** eingeben, **NodeClass Objects** und **Namspace http://example.com** auswählen.

Objekt Example eigefügt – OPC UA Attributes



Variable einfügen

Für das Beispiel werden vier (4) Variable eingefügt. Eine Variable (CounterVar) soll einen Wert (Int16) aus der CPU 1500 (ValueCounter / DB 5) kontinuierlich lesen. Das Python-Programm dividiert den Wert (Int16) und schreibt diesen Wert (Int16) in die zweite Variable (OutRes), der von dort an die Variable Valueln (Dataln / DB 10) gegeben wird. Der Variablen CounterVar (Int16) wird der Wert MaxValue aus der CPU 1500 (ValueCounter / DB 5) übergeben.

Der Variablen InData wird ein Wert (Float) vom Python-Programm zugewiesen. Die Variablen InVar (Int16) führt eine Berechnung durch.

Der Wert der Variable OutRes (Float) wird in die Variable RealData (Dataln / DB 10) geschrieben. Außerdem wird eine UserMethod mit einer Berechnung eingefügt.

Variable CounterVar einfügen



Methode erstellen (Add Instance / Method)

= SIEMENS 🌁 🞦 📕 📑 🖳 🖳 🛄 Namespaces: Editing (current) 1: http://examp Information model http://e No details • OPC Root Move Node ▼ OP¢ Objects OPC Aliases Add Instance 🔹 🧊 Example Delete Node CounterVar (Int16) Add Interfa Add Instance 💷 InVar (Float) - OutRes (Int16) Addin OutVar (Float) Name UserMethod Change Acce Server NodeClass Method Move to nam OPC Types OPC Views Bookmarks Object Namespace Copy Path to Variable ReferenceType Cancel 🗾 Ok

Mit einem Rechtsklick auf Objects und einem Klick auf den Befehl Add Instance im geöffneten Kontextmenü wird das Dialogfeld Add Instance geöffnet. Den Namen UserMethod eingeben, NodeClass Method und Namspace http://example.com auswählen und mit OK bestätigen.



Die Methode enthält Ordner, um Input und **Output Argumente**

Siemens OPC UA Modeling Editor 2.8.0

= SI

📑 📑 📑

No details

V OPC Root

EMENS

Information mode http://example.com -

U 🕕

Input / Output Argumente festlegen



Namespaces: Editing (current) 1: http://exa

Argument Attributes

Name

Attributes / References

Value

Mit einem Rechtsklick auf *InputArguments (Argumente)* und einem Klick auf den Befehl *Add New Arguments* im geöffneten Kontextmenü wird das *Arg1* eingefügt.

Durch erneutes Öffnen des Kontextmenü und Anklicken des Befehls *Add New Arguments* werden weitere Argumente eingefügt.

Auf die gleiche Weise können Argumente in dem Ordner *OutputArguments* erstellen werden.

Argument DataType und Name ändern

Mit einem Klick auf die Variable der Methode werden Argument Attributes der Variablen aufgelistet.



SIOME - OPC UA

Namespaces

Der **DataType** der Variablen des **InputArguments Arg1** (**Value1**) wird auf **Float** festgelegt und der Name auf **Value1** (**Arg1**) geändert.

Der **DataType** der Variablen des **InputArguments Arg2** (**Value2**) wird auf **Float** festgelegt und der Name auf **Value2** (**Arg2**) geändert.

Der **DataType** der Variablen des **OutputArguments Arg1** (**Result**) wird auf **Float** festgelegt und der Name auf **Result** (**Arg1**) geändert.



Für das Beispiel sind alle erforderlichen Festlegungen abgeschlossen. Mit Anklicken des Symbols **Save As** wird die Konfiguration als *.xml* Datei gespeichert. Die *.xml* Datei (*SiOME Nodeset Manual Example.xml*) ist bereit um in den IBH UA Editor als *NodeSet* eingelesen zu werden.

1.2 IBH OPC UA Editor – Vorbereitung NodeSet einlesen

- 1. IBH OPC UA Editor aufrufen.
- 2. Serververbindung erstellen. In dem Beispiel wird ein IBH Link UA QC genutzt.



Der OPC-Server soll Verbindung zu einer CPU 1500 IP-Adresse **10.0.13.90** aufnehmen. Alle Geräte sind im **SubNet IP:**



BH ORC UA BE

10.0.13.00/24 verbunden. Die IP-Adresse des IBH Link UA QC (steuerungsebene) ist 10.0.13.89. Als sicherheitsverfahren ist **None** und als Variablenformat ist **Klassisch** ausgewählt.

3. Verbindungseinstellungen zur Steuerung mit dem Namen CPU 1500,



dem Protokoll **S7-1500**



TCP /IP und der IP-Adresse 10.0.13.90 festlegen. Die Verbindung als erfolgreich testen.

- 4. Programmzuordnung CPU 1500 TIA NodeSet / Counter.
- Als OPC-Tag aus den Datenbaustein
 CounterValues (DB5) die Variablen
 ValueCounter und MaxValue auswählen.
- Als OPC-Tag aus den Datenbaustein *Dataln* (*DB10*) die Variable *Valueln* und die Variable *RealData* auswählen.



OPC-Variable CPU 1500

Workshop CPU 1500 NodeSet.opu - OPCUAEdit — — — — X										
Datei Bearbeiten Hilfe	<u>Datei</u> Bearbeiten Hiffe									
Projekt	Jekt 2 3 Image: Section of the section o									
	🗳 🗙 X b B ó b 🦯	Y								
	Name	Adresse	SPS-Typ	Länge	Herkunft	Zugriff	OPC-Typ	U	o	Kommentar
	CounterValues.MaxValue	DB5.DBW 2	Int	2	Programm	RW	Int16	0	0	maximaler Zählerstand PLC 1500
	CounterValues.ValueCounter	DB5.DBW 4	Int	2	Programm	RW	Int16	0	0	counter value PLC 1500
	Dataln.Valueln	DB10.DBW 2	Int	2	Programm	RW	Int16	0	0	
🙀 Projekt 🛛 🖓 Server 🛛 🖓 Zertifi	Dataln.RealData	DB10.DBD 4	Real	4	Programm	RW	Float	0	0	
										LIE NUM

1.2.1 NodeSet-Konfiguration hinzufügen



Mit einem Rechtsklick auf *IBH Link UA* und einem Klick auf den Befehl *NodeSet-Konfiguration hinzufügen…* im geöffneten Kontextmenü wird ein Dialogfeld zur Nodeset-Übernahme geöffnet.

In dem Beispiel soll erstellte Nodeset-Konfiguration aus dem SiOME-Editor erfolgen. Im Dialogfeld ist **Nein** zu bestätigen.



Ist **NodeSet** im rechten Fenster markiert, werden im linken Fenster **NodeSet** und **Python** angezeigt

Mit einem Rechtsklick auf **NodeSet** und einem Klick auf den Befehl **NodeSet hinzufügen...** im geöffneten Kontextmenü wird das Dialogfeld **Nodeset hizufügen** zur Auswahl der Nodeset-Konfiguration aus dem SiOME-Editor geöffnet. Die **.xml** Datei **SiOME Nodeset Manual Example** wurde übernommen.

Mit einem Rechtsklick auf **Python** und einem Klick auf den Befehl **Pythonmodul hinzu***fügen...* im geöffneten Kontextmenü wird das Dialogfeld zur Festlegung des Dateinamens und Speicherortes des Python-Moduls geöffnet. Die Eingabe ist mit **OK** zu bestätigen.



Python-Modulvorlage mit Variablen erstellen.

Mit einem Rechtsklick auf *Python-Modul.py* und einem Klick auf den Befehl *Python-*

Modulvorlage mit Variablen erstellen... im geöffneten Kontextmenü wird basierend auf der im SiOME OPC UA Modeling Editor erstellten Konfiguration erstellt.



Python-Modul hinzufügen



Das eingefügte Python-Modul wird angezeigt.



Die Modulvorlage muss entsprechend den Erfordernissen des Beispiels angepasst werden. Diese Python-Programmänderung kann direkt im IBH OPC UA Editor oder komfortabler in einem Python-Programmiersystem (z.B. Visual Studio Code von Microsoft[®]) durchgeführt werden.



Python-Modul mit im NodeSet definierten Variablen

Vorhandene Funktionen in Python-Modulen 1.3

import ibhua

Mit import ibhua wird das Python-Modul des IBH Link UA aufgerufen. Ein im IBH Link UA auszuführendes Python-Programm startet mit import ibhua.

Modul init_opc

def init opc():

Die Funktion "init_opc()" wird zur Initialisierung von dem IBH Link UA aufgerufen.

ns1 = ibhua.get namespace(name)

Die Funktion gibt die Namespacenummer zurück. Als Parameter (name) ist der Namespacename, der im Beispiel im SiOME festgelegt wurde, eingetragen.

ns1 = ibhua.get namespace("http://example.com")

ibhua.variable(ns,id,"read funktion","write funktion")

Die Funktion ermöglicht das Lesen und Schreiben von OPC-Variablen. Diese Variablen werden im **UaExpert** unter dem Namen des im Beispiel im SiOME festgelegt Namespacenamen aufgelistet. Es besteht keine Datenverbindung.

Parameter:

ns: Namespacenummer

Id : Nodename oder Numerischer ID

read funktion : Funktion die beim Lesen der Variablen aufgerufen wird. Die Funktion enthält ein Ausgangsparameter aber kein Eingangsparameter. write funktion : Funktion die beim Schreiben der Variablen aufgerufen wird Die Funktion enthält ein Eingangsparameter kein Ausgangsparameter.

Für alle im NodeSet vorhandenen Variablen werden diese Funktionen automatisch erstellt.

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(ns,id,"funktion")

Die Funktion ermöglicht das Festlegen von Methoden. Diese Variablen der Methoden werden im *UaExpert* unter dem Namen des im Beispiel im SiOME festgelegt Namespacenamen aufgelistet.

Für alle im NodeSet vorhandenen Methoden werden diese Funktionen automatisch erstellt.

Parameter:

ns : Namespacenummer

Id : Nodename oder Numerischer ID

funktion : Name der Methode die aufgerufen wird

ibhua.method(ns1,7004,"UserMethod ns1 7004")

return

Dies ist die Beendigung der oberhalb festgelegten Moduls.

Read / Write Modul

```
#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
```

Für jede im NodeSet vorhandene und im Python-Modul *init_opc()* deklarierte Variable wird automatisch ein *Read /Write* Modul erstellt. Jedes Modul hat als Überschrift eine Kommentarzeile mit dem *Variablennamen* und dem im SiOME festgelegten *Data-Type*.

1.4 Python-Modulvorlage überprüfen

Die Python-Modulvorlage kann nach Übertragung an den **OPC UA-Server** (IBH Link UA) mit dem IBH OPC UA Editor bzw. dem Programm **UaExpert** (OPC Unified Architecture Client) überprüft werden.

Konfiguration an OPC UA Server übertragen

Zur Anpassung der Python-Modulvorlage werden die **Nodeld** der OPC-Variablen benötigt. Diese können aus dem IBH OPC UA Editor (**OPC-Tags**) bzw. **UaExpert** Programm (OPC Unified Architecture Client) kopiert werden.

🗱 Workshop CPU 1500	NodeSet.opu - OPCUAEdit	Die Konfiguration muss hierzu an den OPC					
Datei Bearbeiten Hi	lfe	UA Server übertragen und der onlinezugriff					
🗋 💕 🗔 🖨 🔞							
Projekt		auf die CPU 1500 gewählteistet sein.					
🛃 📽 🗙 🖇 🗈 🛍	t û	🐱 Konfiguration zum Server übertragen — 🗆 🗙					
E	Rechtsklick	Name der Serververbindung Übertragung Status					
🔤 🚰 Variablen	Neue Serververbindung	IBH Link UA					
📖 🛄 NodeSets	Neue Steuerung	Markieren)					
	Einfügenanklicken)						
	Markierte Konfiguration zum OPC UA Server übertragen						
	Gesamte Konfiguration vom OPC UA Server einlesen	anklicken					
	Importieren	Start Schließen Hilfe					
	Exportieren						

Wird die Konfiguration aus dem IBH OPC UA Editor zum IBH Link UA (OPC UA Server) übertragen, werden die im OPC UA Server vorhandenen OPC-Tags (angezeigt im Fenster Siemens Slots).

IBH Link UA Browser-Fenster Siemens Slots



Der Inhalt des übertragenen *Nodesets* aus dem IBH OPC UA Editor wird im IBH Link UA Browserfenster *Nodeset_*angezeigt.

🗱 IBH Link UA - Nodeset	× +					
ightarrow C $ m G$	Q 10.0.13.89/?_=/de/opc_slots	${igardown}$	\mathbf{F}	٢	பி	≡
	OPC Server läuft Abmelden Passwort ändern		qu	ad-c	ore	
Netzwerk						
Sicherheit	Phython-Workshop.py SiOME Nodeset Workshop Example.xml					
Zartifikata	Lade XML Lade Python Alles löschen					

IBH OPC UA Editor (Fenster Server)



Da eine online Verbindung zur CPU 1500 besteht, werden die angezeigten Werte der Variablen laufend aktualisiert.

Das Python-Programm ist noch nicht angepasst, dadurch werden keine Werte an die Variablen übergeben.

NotSet -Konfiguration-Variable anzeigen

Die als **NotSet -Konfiguration** eingefügten Variablen werden im linken Server-Fenster unter UA Nodes / Objects unter dem Namen der **NotSet -Konfiguration** (**Example**) aufgelistet. Die Attribute der einzelnen Variablen werden nach dem Markieren im rechten Server-Fenster angezeigt.

🐱 Workshop CPU 1500 NodeSet.opu - OPCUAEdit — 🛛 🗙								
Datei Bearbeiten Hilfe								
		Nedeld						
		Node XMI	ns=14/i=6000					
Ċ		NamesnaceIndex	14					
🖃 📲 IBH Link UA		IdentifierType	Numeric					
🗄 🌆 CPU 1500		Identifier	6000					
CounterValues		Properties						
🛄 Dataln		NodeClass	Variable					
🖅 📖 🛄 Nodesets		BrowseName	InData					
💼 🛅 UA Nodes 🕽		DisplayName	InData					
- Objects		Description						
Client		WriteMask	0					
+ A DeviceSet		UserWriteMask	0					
Example		ValueRank	-1					
Counterivar		AccessLevel	CurrentRead,CurrentWrite					
		UserAccessLevel	CurrentRead,CurrentWrite					
InVar		MinimumSamplingInterval	0					
Q OutBes		Historizing	false					
		DataType						
		DataType	Float					
MQTI		Node XML	i=10					
		NamespaceIndex	0					
		IdentifierType	Numeric					
tier server		Identifier	10					
		Value						
		Value						
Projekt Server Zertifikate								
			UF NUM .:					

Die **Nodeld** der Variablen wird angezeigt.

Ein *Wert* (*Value*) wird erst nach Anpassung und Übertragung des Python-Programms zugeordnet und damit angezeigt werden.

UaExpert – Data Access View

📮 Unified Automation UaExpert - The OPC Unified Architecture Client - NewProject* 🛛 🚽 🗆 🗙								
<u>File View Server D</u> ocument Settings <u>H</u>	lelp							
🗋 🥟 🕞 🖉 🧿 🔶 🗕 🔅	🗙 🔦 🤰 🖹 🖹 🛄							
roject B × Data Access View								
✓	# Server Node Id	Display Name Value	Datatype So	ource Timestamp	Server Timestamp	Statuscode		
✓	1 IBHLinkUA NS14 Numeric 600	1 CounterVar 0	Int16 17	10:19.450	17:10:19.450	Good		
IBHLinkUA@ibhlinkua-005668 2	2 IBHLinkUA NS14 Numeric 600	0 InData 0	Float 17	10:19.450	17:10:19.450	Good		
✓	BHLinkUA NS14 Numeric 600	2 InVar 0	Int16 17	10:19.450	17:10:19.450	Good		
Data Access View 4	4IBHLinkUA NS14 Numeric 600	3 OutRes 0	Float 17	10:19.450	17:10:19.450	Good		
Address Space & x								
🚱 No Highlight	Call UserMethod or	Example ?	\times					
Root > Client > Client Name > DericeSet > Example > InData > InData > InVar > InputArguments > OutputArguments > MQTT > PLCs	Input Arguments Name Value Value1 Value2 Output Arguments Name Value Result rowse	DataType Descri Float Float DataType Descri Float Call Close	ption ption					

Der IBH Link UA muss im *UaExpert* als OPC UA Server angemeldet sein.

Die im Fenster **Nodeset** aufgelisteten Variablen werden im **UaExpert** (OPC UA Client) dargestellt.

1.5 Python-Module anpassen

Im IBH OPC UA Editor kann das Editieren direkt ausgeführt werden.



Mit der folgenden Windows Anpassung kann ein anderer Python-Editor (z.B. Visual Studio Code) geöffnet werden.

Anmerkung:

Um mit dem Befehl **Pythonmodul bearbeiten...** einen externer **Python-Editor** aufzurufen, muss dieser als Standard-App für den Dateityp **.py** festgelegt sein.

1.6 Python-Moduldatei anpassen.

Die im IBH OPC UA Editor erstellten Python-Module müssen angepasst werden.

Apps > Standard-Apps						
Einen Sta	andard für einen Dateityp oder einen Linktyp festlegen					
.py	٩					
×	Visual Studio Code Python-Quelldatei					

Anmerkung:

Um das Python-Modul in Visual Studio Code fehlerfrei anzuzeigen, ist die von IBHsoftec zur Verfügung gestellte Datei *ibhua.pyi* vor den Starten in das *Projektverzeichnis*, in dem sich die *.xml* Datei des IBH OPC UA Editors und das dazugehörende *Python-Modul .py* befindet abgelegt sein.

Kopieren der Nodeld in die Windows-Zwischenablage.



Werden die OPC-Tags angezeigt, wird mit einem Rechtsklick auf die Variable das Kontextmenü geöffnet.

Durch Klicken auf den Befehl **UA-Nodekennung kopieren** wird die ID in die Windows-Zwischenablage kopiert. Diese kann in die Python-Anweisung eingefügt werden.

Im Beispiel wurde die **Nodeld** der Variable **RealData** kopiert. Mit diesem Verfahren können alle in Python-Befehlen erforderlichen **Nodelds** kopiert und eingefügt werden.

Nodeld aus UaExpert-Attribute-Fenster kopieren.



Wurde die im OPC UA Editor erstellte OPC-Variablenkonfiguration auf den IBH Link UA übertragen, kann die *Nodeld* einer OPC-Variable im Fenster des *UaExpert*-Programms kopiert werden. Wenn eine Variable im *Address Space*-Fenster markiert ist, kann ihre *Nodeld* in das *Attributes*-Fenster kopiert werden.

Im Beispiel wurde die *Nodeld* der Variable *ValueCounter* markiert.

OPC-Variable aus CPU 1500 lesen

Die **CPU 1500** Variable (**DB5 – ValueCounter**) die als OPC-Tag festgelegt wurde, soll durch Anpassung der Python-Modulvorlage

def init_opc():

kontinuierlich gelesen werden. Da im IBH OPC UA Editor automatisch erstellen Python-Frame nur Read/Write Module für die Variablen erzeugt wurden ist im Pyton-Modul *init_opc* ist der Ort der Variablen, die zu monitoren ist, deklariert werden. Hierzu ist folgende Zeile hinzuzufügen:

<pre>def init_opc():</pre>					
<pre>ns1 = ibhua.get_namespace("http://example.com")</pre>					
ibhua.variable(ns1,6000,"Read_InData_ns1_6000","Write_InData_ns1_6000")	#Float				
<pre>ibhua.variable(ns1,6001,"Read_CounterVar_ns1_6001","Write_CounterVar_ns1_6001")</pre>	#Int16				
<pre>ibhua.variable(ns1,6002,"Read_OutRes_ns1_6002","Write_OutRes_ns1_6002")</pre>	#Float				
<pre>ibhua.variable(ns1,6003,"Read_InVar_ns1_6003","Write_InVar_ns1_6003")</pre>	#Int16				
ibhua.method(ns1,7004,"UserNethod_ns1_7004")					
(ibhua.monitor("ns=4;s=IBH Link UA.CPU 1500.Programs.CounterValues.ValueCounter","Monitor_var_Intl6_always",1000,0,0)					
return					

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

ibhua.monitor

aus *ibhua* soll etwas kontinuierlich betrachtet werden (Text ohne Leerzeichen).

"node"

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

```
ValueCounter",
```

direkt hinter dem Anführungszeichen (") wird der **Nodeld** der **CPU 1500** Variablen (**CounterValues [DB5] – ValueCounter**) eingetragen.

Der **Nodeld** wird aus dem **UaExpert** kopiert. Es ist sicherzustellen das keine zusätzliche Leerzeichen hinzugefügt oder entfernt werden.

"Monitor Funktion",

"Monitor_var_Int16_always", schreiben – ohne Leerzeichen.

Intervall,

1000 gibt an, dass der Wert der mit *nameset* angegebene Variable alle 1000 ms (1s) gelesen werden soll.

Triggermode,

0 = wird immer getriggert;	1=Trigger bei Wertänderung
2=Trigger bei ansteigender Flanke	3=Trigger bei abfallender Flanke

Deadband

"Deadband": bestimmt die minimale Wertänderung, die zum Trigger führt

Gelesen Variable in eine andere Variable schreiben

Die gelesene Variable (*ValueCounter*), soll durch Anpassung der Python-Modulvorlage durch zehn (10) geteilt werden und kontinuierlich (Monitor) in die Variable *Valueln* (*DatenIn [DB10]*) übertragen werden.

Ibhua.OPCWriteVar("node",var)

Das Pyton-Modul Monitor_var_Int16_always (var) : ist hinzuzufügen:

```
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):

Vorgabe des Namens des Python-Moduls der einen Werte kontinuierlich in die *CPU 1500* schreiben soll.

- myVar=var//10 der übernommene Wert wird durch zehn (10) geteilt und als myVar abgelegt.
- ibhua.OPCWriteVar in *ibhua* soll eine OPV-Variable geschrieben werden (Text ohne Leerzeichen).
- "node"

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

in Anführungszeichen ('') wird der **Nodeld** der **CPU 1500** Variablen (**Dataln [DB10] – Valueln**) eingetragen.

Die **Nodeld** der Variablen kann mit dem Befehl **UA-Nodekennung kopieren** direkt aus dem OPC UA-Editorfenster oder aus dem **Attributefenster** des UaExpert-Programms in die Windows-Zwischenablage kopiert werden, um diese in die Python-Anweisung einzufügen. Nach der **Nodeld** muss ein Anführungszeichen (<u>"</u>) stehen, gefolgt von einem Komma und einem Parameter.

• var

myVar Der Wert der Variablen wird in die *CPU 1500* Variable (*Dataln [DB10] – Valueln*) eingetragen.

Return

Ende des Python-Moduls.

Read/Write Module – Variable zuordnen

Den einzelnen Modulen werden im Beispiel Werte zugeordnet damit diese im **UaExpert** (OPC UA Client) angezeigt werden.

#InData : Float

Der Variablen InData ist der Wert 98765.4 (Data-Type Float) zugeordnet.

Modul angepasst	Modul automatisch erstellt
<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

Der Variablen *CounterVar* ist der Wert aus der *CPU 1500*, Datenbaustein *CounterValues* / *Struct [DB 5]*; Variable *MaxValue* (Data-Type Int16) zugeordnet.

Ibhua.OPCWriteVar("node")

- **ibhua.OPCWriteVar** in **ibhua** soll eine OPV-Variable geschrieben werden (Text ohne Leerzeichen).
- "node"

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

- Structure.MaxValue",
- in Anführungszeichen (") wird der **Nodeld** der der **CPU 1500** Variablen MaxValue eingetragen.

Die **Nodeld** der Variablen kann mit dem Befehl **UA-Nodekennung kopieren** direkt aus dem OPC UA-Editorfenster oder aus dem **Attributefenster** des UaExpert-Programms in die Windows-Zwischenablage kopiert werden.

Modul angepasst	Modul automatisch erstellt
<pre>#CounterVar : Int16 CounterVar_ns1_6001 = 0 def Read CounterVar ns1_6001():</pre>	<pre>#CounterVar : Int16 CounterVar_ns1_6001 = 0 def Read_CounterVar_ns1_6001(): return CounterVar_ns1_6001</pre>
CounterVar_ns1_6001=ibhua.OPCReadVar("ns=4;s=IBH Link UA.CPU 1500 return CounterVar_ns1_6001	Programs.CounterValues.MaxValue")

#OutRes : Float

Wird der Variablen **OutRes** ein Wert zugeordnet, dies kann durch ändern des Wertes (**Value**) im Fenster Data Access View im **UaEpert** verursacht werden, wird mit dem

Schreibbefehl *Write_OutRes* der Inhalt der Variablen *OutVar* in die Variable *RealData* (*CPU 1500, Dataln [DB 10]*), übertragen.

Dies geschieht nur, wenn der Wert in der Variablen der **OutRes** sich ändert. Mit anderen Worten: Wird der Wert **OutRes** im Attribut-Fenster des **UaEpert** verändert, erscheint der geänderte Wert in der Variablen **RealData (CPU 1500, DataIn [DB 10])**.

Die **Nodeld** der Variablen kann mit dem Befehl **UA-Nodekennung kopieren** direkt aus dem **OPC UA-Editorfenster** oder aus dem **Attributefenster** des **UaExpert**-Programms in die Windows-Zwischenablage kopiert werden, um diese in die Python-Anweisung einzufügen. Nach der **Nodeld** muss ein Anführungszeichen (") stehen, gefolgt von einem Komma und einem Parameter.

Modul angepasst	Modul automatisch erstellt
<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 ibhua.OPCWriteVar("ns=4;s=IBH Link UA.CPU 15 return</pre>	<pre>def Write_OutRes_ns1_6002(value): global OutRes_ns1_6002 OutRes_ns1_6002 = value return 500.Programs.DataIn.RealData",value)</pre>

#InVar : Int16

Der Variablen InVar wird durch eine einfache Rechenaufgabe gebildet (Data-Type Int16).

Modul angepasst	Modul automatisch erstellt
<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>

Modul UserMethod

In der eingefügten Methode sollen die beiden *InputArguments Value1* und *Value2* addiert werde. Das Ergebnis soll unter *InputArguments Result* vorliegen.

Modul angepasst	Modul automatisch erstellt
<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>

Mit der Befehlszeile **Result=Value1+Value2** wird die Addition ausgeführt. Befehlszeile **return Result** bringt das Ergebnis zur Anzeige.

1.6.1 Python-Module angepasst

CounterVar_ns1_6001 = value	
return	
<pre>def Monitor_var_Intl6_always(var):</pre>	
myVar=var//10 ibbus OBCUsiteVer/"restanceIBU Link UA CBU 1500 Decembra DeteIn Uslanta" relies	
return	
#OutRes : Float	
OutRes_ns1_6002 = 0.0	
def Read_OutRes_ns1_6002():	
return OutRes_ns1_6002	
<pre>def Write_OutRes_ns1_6002(value):</pre>	
global OutRes ns1_6002	
return	
#InVer : Int16	
InVar ns1 6003 = 0	
def Read_InVar_ns1_6003():	
x=20	
y=25	
Z=500 Invitor and 6002=virtual	
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 nsl 7004(Value1, Value2):	
return Result	
aei init_opc(): 	
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
<pre>ibhua.variable(ns1,6003,"Read_InVar_ns1_6003","Write_InVar_ns1_6003")</pre>	#Int16
<pre>ibhua.method(ns1,7004,"UserMethod_ns1_7004")</pre>	
ibhua.monitor("ns=4;s=IBH Link UA.CPU 1500.Programs.CounterValues.ValueCounter","	Monitor_var_Int16_always",1000,0,0)
return	

1.7 IBH OPC UA Editor (Fenster Server)



Da eine online Verbindung zur CPU 1500 besteht, werden die angezeigten Werte der Variablen laufend aktualisiert.

Das Python-Programm ist angepasst und übertragen. Die Werte der Variablen werden laufend aktualisiert.

NotSet Variable anzeigen

Die als **NotSet -Konfiguration** eingefügten Variablen werden im rechten Server-Fenster unter dem Namen der **NotSet -Konfiguration** (**Example**) aufgelistet. Die Attribute der einzelnen Variablen werden nach dem Markieren im rechten Server-Fenster angezeigt.

Das Python-Programm ist angepasst und übertragen. Die *Werte* der Variablen werden laufend aktualisiert.

🗯 Workshop CPU 1500 NodeSet.opu - OPCUAEdit	:					—		×
Datei Bearbeiten Hilfe								
🗋 💕 🛃 🕼 🞯								
Server 🗸 🗸	×	Name	Datentyp	Status	Zugriff	Wert	Noder	ame
¢	_	🚺 CounterVar	Int16	ОК	RW	8000	6001	
🖃 📢 IBH Link UA		🚺 InData	Float	ОК	RW	98765.4	6000	
		📘 InVar	Int16	OK	RW	545	6003	
🛄 CounterValues		OutRes	Float	OK	RW		6002	
🛄 Dataln						<u>ا</u> ا	Nerte	
🖕 🛄 Nodesets						werde	n aktuali	siert
SiOME Nodeset Workshop Example.x	ml							
Example Markieren								
UserMethod								
Python-Workshop.py								
Rojekt 🖾 Server 🧟 Zertifikate								
							UF I	NUM;

1.8 Variable in UaExpert

Unified Automation UaExpert - The OPC U	nified Ar	rchitecture Clien	t - NewProject*							×
<u>File View Server D</u> ocument <u>S</u> ettings	<u>H</u> elp									
🗋 💋 🕞 🗭 🧿 🔶 🗖	> 🗙	🔌 🚨 📗	e 🛛 🖵							
Project 🗗 🗙	Data /	Access View								8
🛩 🎵 Project	#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Status	code
✓	1	IBHLinkUA@	NS4 String IBH Link UA.CPU	MaxValue	9500	Int16	13:10:09.623	13:10:10.716	Good	1
🚫 IBHLinkUA@ibhlinkua-005668	2	IBHLinkUA@	NS4 String IBH Link UA.CPU	ValueCounter	5056	Int16	13:11:43.258	13:11:43.508	Good	
✓	3	IBHLinkUA@	NS4 String IBH Link UA.CPU	RealData	1234.5	Float	13:10:16.757	13:10:17.719	Good	
📁 Data Access View	4	IBHLinkUA@	NS4 String IBH Link UA.CPU	Valueln	432	Int16	13:11:42.739	13:11:42.757	Good	
Address Space & X	5	IBHLinkUA@	NS14 Numeric 6001	CounterVar	9500	Int16	12:17:06.207	12:17:06.207	Good	
A No Highlight	6	IBHLinkUA@	NS14 Numeric 6000	InData	98765.4	Float	12:17:06.207	12:17:06.207	Good	
Root	7	IBHLinkUA@	NS14 Numeric 6003	InVar	545	Int16	12:17:06.208	12:17:06.208	Good	
Y Directs	8	IBHLinkUA@	NS14 Numeric 6002	OutRes	0	Float	12:17:06.208	12:17:06.208	Good	
> A Client			4							
> 📥 DeviceSet		dr	ag & drop							
V 📥 Example			7							
CounterVar			/							
> 🔲 InData 🧧	Rebro	owse	/							
> 💷 InVar 📃 🛄	Call	Click								
👌 💷 OutRes 🔶 📊 🖓 👘										
✓ =♥ UserMethod		/ /	📕 Call UserMethod on Examp	le ?	\times					
InputArguments										
OutputArguments										
> 👶 MQTT			Input Arguments							
> 뤚 Modbus			Name Value	DataType Desc	ription					
✓ □ PLCs		/	Value1 4711.123	Float						
✓ CPU 1500	1		Value2 4712.456 (enter)	Float						
> DeviceHealth	1		Output Arguments							
DeviceRevision	1		Name Value	DataType Des	ription					
HardwareRevision	/		Decult 0423 58	Float						
Manufacturer			Result 9425.50	Hour	_					
Model			Result							
🗸 👶 Programs			Succeeded							
✓			(confirm)							
> 🖾 MaxValue			<u> </u>	Call Clo	ose					
🔉 🍩 ValueCounter		C								
Y 🛅 DataIn										
> 💷 RealData										
🔰 🔍 Valueln										
> 🖾 SupportedTypes										
RevisionCounter										
 SerialNumber SoftwarePowieion 										
Softwarercevision										
/ 🚚 IdSKS										

Valueln

Dat	a Access View					
#	Display Nan	ne Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	MaxValue	9500	Int16	13:10:09.623	13:10:10.716	Good
2	ValueCounter	3387	Int16	14:30:31.366	14:30:31.395	Good
3	RealData	1234.5	Float	13:10:16.757	13:10:17.719	Good
4	Valueln	360	Int16	14:30:31.369	14:30:31.395	Good
Data #	Access View Server			Node ld		Display Name
1	IBHLinkUA@	NS4 String IBH L	lues.MaxValue	/laxValue		
2	IBHLinkUA@	NS4 String IBH L	ink UA.CPU 1	500.Programs.CounterVa	lues.ValueCounter	/alueCounter
3	IBHLinkUA@	NS4IStringIIBH L	ink UA.CPU 1	500. Programs. Dataln. Rea	IData F	RealData

	IDI II Coluli A @	NC4ICtain all DLLLinds LLA_CDLL1500 Data and an Data la Malvada
4	IBHI INKUAW	NS4ISTRINGUBELLINK UA.CPU LOUU.Programs.Datain.valuein
	ier ien noor te m	ris gound for the officer of the officer officer of the officer of

OPC variable InVar

Dat	a 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							8
#	Server	Node Id	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		

OPC Variable CounterVar

Data	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							0
#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
				_				

1.9 Weiterführende Informationen

Das Python Modul, bestückt mit OPC-Variablen wird von dem im IBH Link UA vorhandenen Python-Interpreter abgearbeitet. Selbstverständlich sind komplexere Aufgaben mit der Programmiersprache Python realisierbar, Maschinenparameter und daten zu analysiert, um Vorhersagen und Empfehlungen für die Optimierung der Maschinenleistung zu treffen. Weiterführende Informationen über Python/Methoden/Datenmodelle stehen im IBHsoftec WIKI bereit: https://wiki.ibhsoftec.com/de/IBH_Link_UA:Python/Methoden/Datenmodelle

lue assigned to OutRes