



TTS

SISTEMA DI RACCOLTA DATI

Server OPC UA

08.10.2024



Server OPC UA	1
Generalità	3
Machine.....	4
Fanuc_FOCAS.....	5
Heidenhain	10
MTConnect	13
EUROMAP63	15
Mitsubishi	18
FAGOR	20
FPConnect.....	23
OpenConnect	25
HTTP.....	28
MQTT	30
OPC DA	32
OPC UA	34
PartProgram.....	37
File	39
DBTable	42
BiometricReader	45
PLC.....	47
SocketTCP	51
TTSObject.....	53
Connection	54
Digital I/O	55
RS232C_1	61
RS232C_2.....	64
Analog Input.....	67
Barcode	69
Watch	70
VAC.....	71
VSC.....	73

Generalità

Il **TTSMAN** è un framework multiplatforma che, dotato dei protocolli di comunicazione delle macchine di produzione, permette di gestire in modo bidirezionale i dati di campo con un MES (o gestionale).

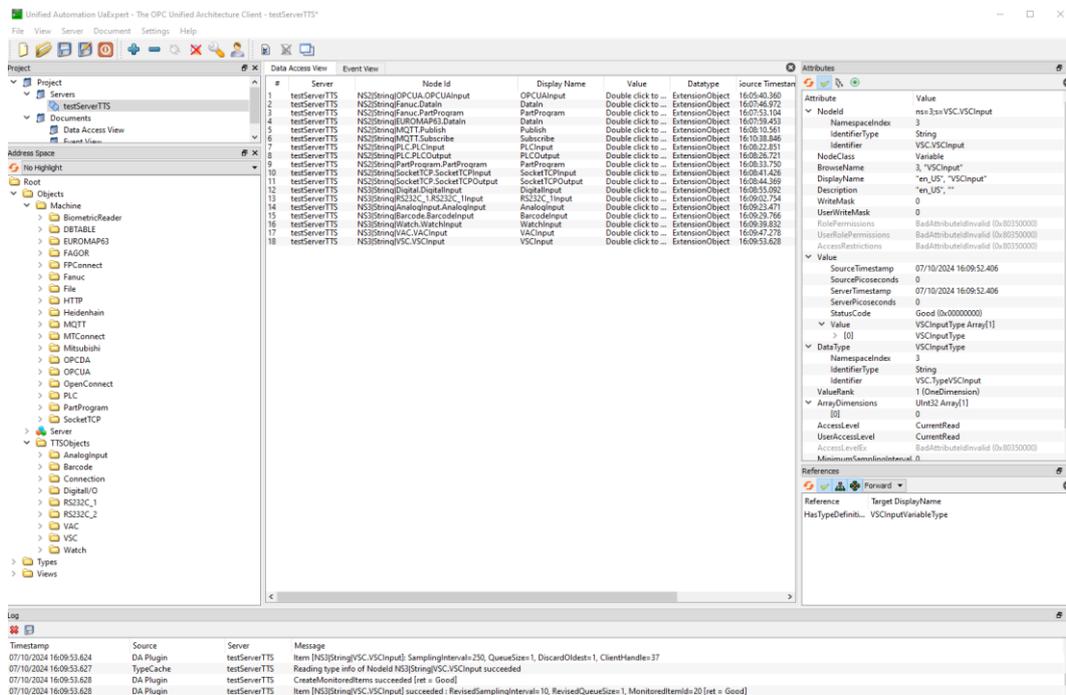
Il trasferimento può avvenire tramite il **Server OPC UA** integrato, in questo caso il TTSMAN diventa un Gateway OPC UA per tutti i dispositivi di campo configurati e trasforma i dati in variabili OPC UA.

L'Url è del tipo: **opc.tcp://IP_Server:48410** (ad esempio: **opc.tcp://127.0.0.1:48410**).

L'impianto configurato:



viene trasposto in OPC UA in due oggetti: **Machine** (dispositivi dotati di un protocollo) e **TTSObject** (dispositivi della PHSNET).



Machine

Nel TTSMSAN un oggetto Machine gestisce lo scambio dati con un dispositivo di fabbrica tramite il suo protocollo di comunicazione.

Nel Server OPCUA ad ogni Machine (Fanuc, Heidenhain, ecc.) sono associate delle variabili complesse (DataIn, MacrodataWrite, ecc.) che danno sinteticamente i valori relativi a tutti i dispositivi configurati (FA100, FAN200, ecc.) e delle variabili semplici (o primitive) per il singolo dispositivo (FAN100_COUNTER, FAN100_MACRODATA, ecc.).

- ▼ **Fanuc**
 - > DataIn
 - > MachineFanuc
 - > MacrodataWrite
 - > PartProgram
 - > PartRequired

- ▼ **MachineFanuc**
 - > FAN100
 - > FAN200

- ▼ **FAN100**
 - > FAN100_ALARM
 - > FAN100_ALARMMessage
 - > FAN100_COUNTER
 - > FAN100_GetStatus
 - > FAN100_MACRODATA
 - > FAN100_MAINPROGRAM
 - > FAN100_MacrodataValueWrite
 - > FAN100_OperatorMessage
 - > FAN100_PARTCOUNT
 - > FAN100_PROGRAM
 - > FAN100_PartRequiredValue
 - > FAN100_PathPartProgram
 - > FAN100_ResultMacrodataWrite
 - > FAN100_ResultPartProgram
 - > FAN100_ResultPartRequired
 - > FAN100_STATUS
 - > FAN100_SYSINFO
 - > FAN100_SetFanuc
 - > FAN100_TABLE
 - > FAN100_TOTALPARTCOUNT

Un esempio di variabile complessa è il seguente:

Attribute	Value
▼ NodeId	ns=2;s=Fanuc.DataIn
NamespaceIndex	2
IdentifierType	String
Identifier	Fanuc.DataIn
NodeClass	Variable
BrowseName	2, "DataIn"
DisplayName	"en_US", "DataIn"
Description	"en_US", "CNC Fanuc Variables"
WriteMask	0
UserWriteMask	0
RolePermissions	BadAttributeValueInvalid (0x80350000)
UserRolePermissions	BadAttributeValueInvalid (0x80350000)
AccessRestrictions	BadAttributeValueInvalid (0x80350000)
▼ Value	
SourceTimestamp	25/01/2022 07:34:03.035
SourcePicoseconds	0
ServerTimestamp	25/01/2022 07:34:03.035
ServerPicoseconds	0
StatusCode	Good (0x00000000)
Value	DBTABLEReadType Array[2]
> [0]	FanucDataInType
> [1]	FanucDataInType
▼ DataType	FanucDataInType
NamespaceIndex	2
IdentifierType	String
Identifier	Fanuc.TypeDataIn
ValueRank	1 (OneDimension)
▼ ArrayDimensions	UInt32 Array[1]
[0]	1
AccessLevel	CurrentRead
UserAccessLevel	CurrentRead
AccessLevelEx	BadAttributeValueInvalid (0x80350000)
MinimumSamplingInterval	0
Historizing	false

Name	Value
▼ [0]	FanucDataInType Array[2]
TypeMachine	FAN200
DescriptionMachine	Fanuc 2
> SYSINFO	String Array[1]
> STATUS	FanucChannelType Array[1]
> PROGRAM	FanucChannelType Array[1]
> MAINPROGRAM	FanucChannelType Array[1]
> TOTALPARTCOUNT	FanucChannelType Array[1]
> PARTCOUNT	FanucChannelType Array[1]
> MACRODATA	FanucChannelType Array[1]
> TABLE	FanucChannelType Array[1]
> COUNTER	FanucChannelType Array[1]
> ALARM	FanucValueType Array[1]
> ALARMMessage	FanucAlarmMessageType Array[1]
> OperatorMessage	FanucOperatorMessageType Array[1]
▼ [1]	FanucDataInType
TypeMachine	FAN100
DescriptionMachine	Fanuc1
> SYSINFO	String Array[1]
> STATUS	FanucChannelType Array[1]
> PROGRAM	FanucChannelType Array[1]
> MAINPROGRAM	FanucChannelType Array[1]
> TOTALPARTCOUNT	FanucChannelType Array[1]
> PARTCOUNT	FanucChannelType Array[1]
> MACRODATA	FanucChannelType Array[1]
> TABLE	FanucChannelType Array[1]
> COUNTER	FanucChannelType Array[1]
> ALARM	FanucValueType Array[1]
> ALARMMessage	FanucAlarmMessageType Array[1]
> OperatorMessage	FanucOperatorMessageType Array[1]

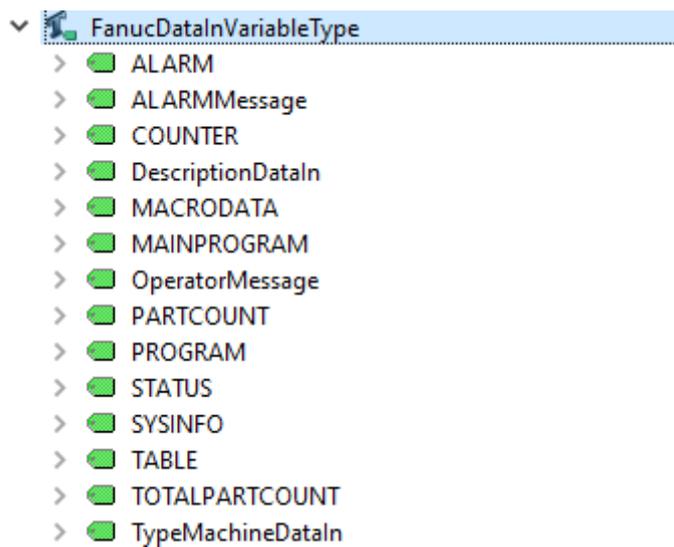
Fanuc_FOCAS

Nel TTSMAN, per i CN FNUC, c'è il driver basato sulle librerie FOCAS che prevede:

- la lettura dei dati macchina;
- il download e l'upload dei partprogram;
- la scrittura dei pezzi da produrre;
- la lettura e la scrittura delle macro.

Nel Server OPC UA abbiamo le variabili:

- ns=2;s=Fanuc.DataIn (lettura dei dati macchina)



Poiché per i CN FANUC ogni canale ha il suo set di variabili, è definito il tipo Channel:

```
"<opc:StructuredType BaseType=\"ua:ExtensionObject\" Name=\"FanucChannelType\"/>\r\n";  
"<opc:Field TypeName=\"opc:Int16\" Name=\"Channel\"/>\r\n";  
"<opc:Field TypeName=\"opc:Int32\" Name=\"NoOfValue\"/>\r\n";  
"<opc:Field LengthField=\"NoOfValue\" TypeName=\"opc:String\" Name=\"Value\"/>\r\n";  
"</opc:StructuredType>\r\n";
```

quindi ogni dato (STATUS, PROGRAM, ecc.) è un array di ChannelType e ad ogni canale è associato un array di stringhe.

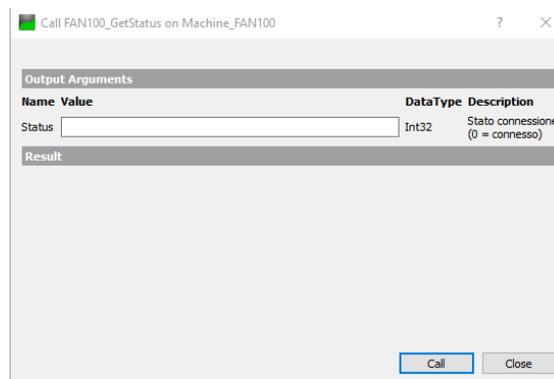
A screenshot of a software interface showing a tree view of variables. The root node is 'FanucDataInType', which is expanded to show a list of sub-variables: TypeMachine (FAN200), DescriptionMachine (Fanuc 2), SYSINFO (String Array[1]), STATUS (FanucChannelType Array[1]), PROGRAM (FanucChannelType Array[1]), MAINPROGRAM (FanucChannelType Array[1]), TOTALPARTCOUNT (FanucChannelType Array[1]), PARTCOUNT (FanucChannelType Array[1]), and MACRODATA (FanucChannelType Array[1]). Each sub-variable is further expanded to show its internal structure, including Channel and Value arrays.

Ci sono poi le variabili semplici per ogni Machine, ad esempio:

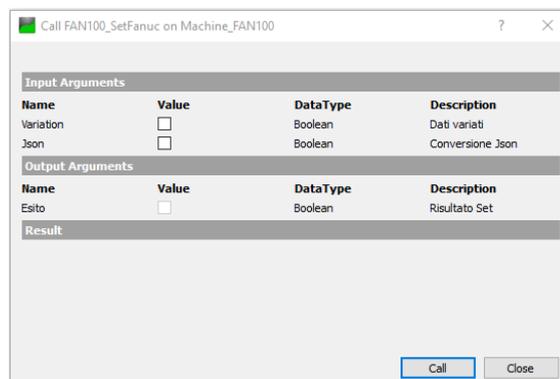
- ns=2;s=Fanuc.FAN100_SYSINFO
- ns=2;s=Fanuc.FAN100_STATUS
- ns=2;s=Fanuc.FAN100_PROGRAM
- ns=2;s=Fanuc.FAN100_MAINPROGRAM
- ns=2;s=Fanuc.FAN100_TOTALPARTCOUNT
- ns=2;s=Fanuc.FAN100_PARTCOUNT
- ns=2;s=Fanuc.FAN100_MACRODATA
- ns=2;s=Fanuc.FAN100_TABLE
- ns=2;s=Fanuc.FAN100_ALARM
- ns=2;s=Fanuc.FAN100_ALARMMessage
- ns=2;s=Fanuc.FAN100_OperatorMessage

I metodi:

- **ns=2;s=Fanuc.FAN100_GetStatus**



- **ns=2;s=Fanuc.FAN100_SetFanuc**



- ns=2;s=Fanuc.PartProgram (trasmissione dei partprogram)

The screenshot shows a tree view on the left and a table on the right. The tree view is expanded to show 'FanucPartProgramVariableType' with sub-items: 'ActPartProgram', 'DescriptionPartProgram', 'PathPartProgram', 'ResultPartProgram', and 'TypeMachinePartProgram'. The table on the right is titled 'FanucPartProgramType Array[2]' and contains two rows of data, each representing a machine type.

FanucPartProgramType Array[2]	
▼ [0]	FanucPartProgramType
TypeMachine	FAN100
DescriptionMachine	Fanuc1
PathPartProgram	
ActPartProgram	false
ResultPartProgram	false
▼ [1]	FanucPartProgramType
TypeMachine	FAN200
DescriptionMachine	Fanuc 2
PathPartProgram	
ActPartProgram	false
ResultPartProgram	false

Per inviare il partprogram bisogna:

- valorizzare il campo **PathPartProgram** (ad es. C:\\TEMP\\KCICCIO.CNC|1|D);
- settare il campo **ActPartProgram**;

il risultato si avrà nel campo **ResultPartProgram**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=Fanuc.FAN100_PathPartProgram;
- ns=2;s=Fanuc.FAN100_ResultPartProgram.

- ns=2;s=Fanuc.PartRequired (pezzi da produrre)

▼		FanucPartrequiredVariableType
>		ActPartRequired
>		DescriptionPartRequired
>		PartRequiredValue
>		ResultPartRequired
>		TypeMachinePartRequired
▼		FanucPartRequiredType Array[2]
▼	[0]	FanucPartRequiredType
	TypeMachine	FAN200
	DescriptionMachine	Fanuc 2
▼	PartRequiredValue	FanucValueType
	Channel	0
	Value	
	ActPartRequired	false
	ResultPartRequired	false
▼	[1]	FanucPartRequiredType
	TypeMachine	FAN100
	DescriptionMachine	Fanuc1
▼	PartRequiredValue	FanucValueType
	Channel	0
	Value	
	ActPartRequired	false
	ResultPartRequired	false

Per inviare il valore dei pezzi da produrre bisogna:

- valorizzare il campo **PartRequiredValue**, valorizzando **Channel** e **Value**;
- settare il campo **ActPartRequired**;

il risultato si avrà nel campo **ResultPartRequired**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=Fanuc.FAN100_PartRequiredValue;
- ns=2;s=Fanuc.FAN100_ResultPartRequired.

- ns=2;s=Fanuc.MacrodataWrite (Scrittura di una macro)

▼		FanucMacrodataWriteVariableType
>		ActMacrodataWrite
>		AddressMacrodata
>		DescriptionMacrodata
>		MacrodataValue
>		ResultMacrodataWrite
>		TypeMachineMacrodata
▼		FanucMacrodataWriteType Array[2]
▼	[0]	FanucMacrodataWriteType
	TypeMachine	FAN200
	DescriptionMachine	Fanuc 2
	AddressMacrodata	0
▼	MacrodataValue	FanucValueType
	Channel	0
	Value	
	ActMacrodataWrite	false
	ResultMacrodataWrite	false
▼	[1]	FanucMacrodataWriteType
	TypeMachine	FAN100
	DescriptionMachine	Fanuc1
	AddressMacrodata	0
▼	MacrodataValue	FanucValueType
	Channel	0
	Value	
	ActMacrodataWrite	false
	ResultMacrodataWrite	false

Per inviare il valore della macro bisogna:

- valorizzare il campo **AddressMacrodata**;
- valorizzare il campo **MacrodataValue**, valorizzando **Channel** e **Value**;
- settare il campo **ActMacrodataWrite**;

il risultato si avrà nel campo **ResultMacrodataWrite**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=Fanuc.FAN100_MacrodataValueWrite, nel formato **Value|Channel|MacroAddress**;
- ns=2;s=Fanuc.FAN100_ResultMacrodataWrite.

Heidenhain

Nel TTSMAN, per i CN Heidenhain, c'è il driver che prevede:

- la lettura dei dati macchina;
- il download e l'upload dei partprogram;

Nel Server OPC UA abbiamo le variabili:

- **ns=2;s=Heidenhain.DataIn (lettura dei dati macchina)**

The screenshot shows a tree view of OPC UA variables. The root node is 'HeidenhainDataInVariableType'. It contains several sub-variables: DescriptionDataIn, DncMode, ErrorMessage, ExecutionMessage, ExecutionMode, ProgramEvent, ProgramName, ProgramState, StateEvent, and TypeMachineDataIn. Below this, a table shows the structure of the 'HeidenhainDataIn Type Array[3]' variable. The first element of the array is expanded to show its fields: TypeMachine (MA200), DescriptionMachine (Heidenhain Fresa 2), DncMode (String Array[1]), ExecutionMode (String Array[1]), ProgramState (String Array[1]), ProgramName (String Array[1]), StateEvent (String Array[1]), ProgramEvent (String Array[1]), ExecutionMessage (String Array[1]), and ErrorMessage (String Array[1]).

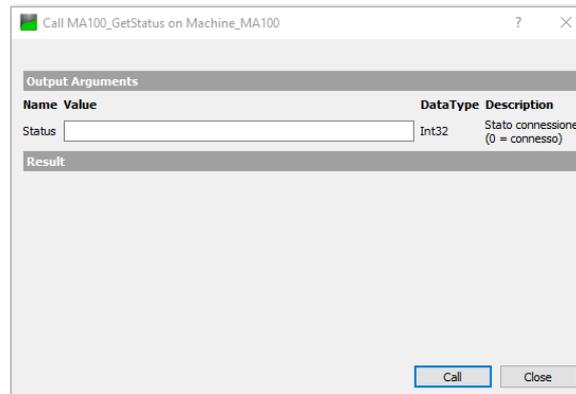
HeidenhainDataIn Type Array[3]	
▼ [0]	HeidenhainDataIn Type
TypeMachine	MA200
DescriptionMachine	Heidenhain Fresa 2
▼ DncMode	String Array[1]
[0]	
▼ ExecutionMode	String Array[1]
[0]	
▼ ProgramState	String Array[1]
[0]	
▼ ProgramName	String Array[1]
[0]	
▼ StateEvent	String Array[1]
[0]	
▼ ProgramEvent	String Array[1]
[0]	
▼ ExecutionMessage	String Array[1]
[0]	
▼ ErrorMessage	String Array[1]
[0]	

Ci sono poi le variabili semplici per ogni Machine, ad esempio:

- ns=2;s=Heidenhain.MA100_DncMode;
- ns=2;s=Heidenhain.MA100_ExecutionMode;
- ns=2;s=Heidenhain.MA100_ProgramState;
- ns=2;s=Heidenhain.MA100_ProgramName;
- ns=2;s=Heidenhain.MA100_StateEvent;
- ns=2;s=Heidenhain.MA100_ProgramEvent;
- ns=2;s=Heidenhain.MA100_ExecutionMessage;
- ns=2;s=Heidenhain.MA100_ErrorMessage.

Il metodo:

- ns=2;s=Heidenhain.MA100_GetStatus



- ns=2;s=Heidenhain.PartProgram (trasmissione dei partprogram)

HeidenhainPartProgramVariableType

- > ActPartProgram
- > DescriptionPartProgram
- > PathDestination
- > PathSource
- > ResultPartProgram
- > TypeMachinePartProgram

HeidenhainPartProgramType Array[3]

Index	TypeMachine	DescriptionMachine	PathSource	PathDestination	ActPartProgram	ResultPartProgram
[0]	MA300	Heidenhain Stampo 3			false	false
[1]	MA100	Heidenhain Tornio 1			false	false
[2]	MA200	Heidenhain Fresa 2			false	false

Per inviare il partprogram bisogna:

- valorizzare il campo **PathSource** (ad es. c:\script.txt);
- valorizzare il campo **PathDestination** (ad es. TNC:\work\prg1);
- settare il campo **ActPartProgram**;

il risultato si avrà nel campo **ResultPartProgram**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=Heidenhain.MA100_PathPartProgram, nel formato **PathSource||PathDestination**;
- ns=2;s=Heidenhain.MA100_ResultPartProgram.

MTCConnect

Nel TTSMAN, per i CN MTCConnect, c'è il driver che prevede:

- la lettura dei dati macchina;
- il download e l'upload dei partprogram;

Nel Server OPC UA abbiamo le variabili:

- **ns=2;s=MTCConnect.DataIn (lettura dei dati macchina)**

The screenshot shows a tree view of OPC UA variables. The root node is 'MTCConnectDataInVariableType'. It has three children: 'DescriptionDataIn', 'TypeMachineDataIn', and 'VariableMTC'. The 'VariableMTC' node is expanded to show an array of 'MTCConnectDataInType' objects. The array has two elements, indexed [0] and [1].

Index	Type	Value
[0]	TypeMachine	MTC200
[0]	DescriptionMachine	MTCConnect PC Laboratorio
[0]	VariableMTC	String Array[0]
[1]	TypeMachine	MTC100
[1]	DescriptionMachine	MTCConnect My PC
[1]	VariableMTC	String Array[0]

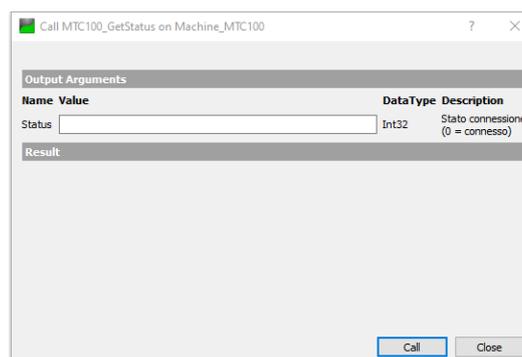
Il campo VariableMTC contiene le variabili MTCConnect selezionate.

C'è poi la variabile semplice per ogni Machine, ad esempio:

- **ns=2;s=MTCConnect.MTC100_VariableMTC.**

Il metodo:

- **ns=2;s=MTCConnect.MTC100_GetStatus**



- ns=2;s=MTConnect.PartProgram (trasmissione dei partprogram)

The screenshot shows a tree view of the `MTConnectPartProgramVariableType` object. It contains several sub-objects: `ActPartProgram`, `DescriptionPartProgram`, `PathDestination`, `PathSource`, `ResultPartProgram`, and `TypeMachinePartProgram`. Below this, an array of `MTConnectPartProgramType` objects is shown, with two elements: `[0]` and `[1]`. Each element contains fields for `TypeMachine`, `DescriptionMachine`, `PathSource`, `PathDestination`, `ActPartProgram`, and `ResultPartProgram`.

MTConnectPartProgramVariableType	
>	ActPartProgram
>	DescriptionPartProgram
>	PathDestination
>	PathSource
>	ResultPartProgram
>	TypeMachinePartProgram

MTConnectPartProgramType Array[2]	
▼ [0]	MTConnectPartProgramType
TypeMachine	MTC200
DescriptionMachine	MTConnect PC Laboratorio
PathSource	
PathDestination	
ActPartProgram	false
ResultPartProgram	false
▼ [1]	MTConnectPartProgramType
TypeMachine	MTC100
DescriptionMachine	MTConnect My PC
PathSource	
PathDestination	
ActPartProgram	false
ResultPartProgram	false

Per inviare il partprogram bisogna:

- valorizzare il campo **PathSource**;
- valorizzare il campo **PathDestination**;
- settare il campo **ActPartProgram**;

il risultato si avrà nel campo **ResultPartProgram**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=MTConnect.MTC100_PathPartProgram, nel formato **PathSource||PathDestination**;
- ns=2;s=MTConnect.MTC100_ResultPartProgram.

EUROMAP63

Nel TTSMAN, per l'EUROMAP63, tramite i JobFile:

- la lettura dei dati macchina;
- l'update delle variabili;

Nel Server OPC UA abbiamo le variabili

- ns=2;s=EUROMAP63.DataIn (lettura dei dati macchina)

EUROMAP63DataInVariableType	
> DescriptionDataIn	
> TypeMachineDataIn	
> VariableEuro	
EUROMAP63DataInType Array[2]	
[0] EUROMAP63DataInType	
TypeMachine	EMAP200
DescriptionMachine	Pressa
VariableEuro	String Array[3]
[0]	DATE=
[1]	TIME=
[2]	COUNT=
[1] EUROMAP63DataInType	
TypeMachine	EMAP100
DescriptionMachine	Prova Euromap
VariableEuro	String Array[3]
[0]	DATE=
[1]	TIME=
[2]	COUNT=

Il campo VariableEURO contiene le variabili previste nel JobFile.

C'è poi la variabile semplice per ogni Machine, ad esempio:

- ns=2;s=EUROMAP63.EMAP100_VariableEuro.

- ns=2;s=EUROMAP63.WriteJob (trasmissione dei parametri)

▼		EUROMAP63WriteJobVariableType
	>	ActWriteJob
	>	DescriptionWriteJob
	>	ResultWriteJob
	>	TypeMachineWriteJob
	>	WriteJob
▼		EUROMAP63WriteJobType Array[2]
	▼ [0]	EUROMAP63WriteJobType
		TypeMachine EMAP100
		DescriptionMachine Prova Euomap
		EUROMAP63WriteJob String Array[0]
		ActWriteJob false
		ResultWriteJob false
	▼ [1]	EUROMAP63WriteJobType
		TypeMachine EMAP200
		DescriptionMachine Pressa
		EUROMAP63WriteJob String Array[0]
		ActWriteJob false
		ResultWriteJob false
▼		EUROMAP63WriteJobType Array[2]
	▼ [0]	EUROMAP63WriteJobType
		TypeMachine EMAP100
		DescriptionMachine Prova Euomap
		EUROMAP63WriteJob String Array[0]
		ActWriteJob false
		ResultWriteJob false
	▼ [1]	EUROMAP63WriteJobType
		TypeMachine EMAP200
		DescriptionMachine Pressa
		EUROMAP63WriteJob String Array[0]
		ActWriteJob false
		ResultWriteJob false

Per inviare i parametri bisogna:

- valorizzare il campo **EUROMAP63WriteJob** con le stringhe nel formato **Set**;
- settare il campo **ActWriteJob**;

il risultato si avrà nel campo **ResultWriteJob**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=EUROMAP63.EMAP100_VariableEuro;
- ns=2;s=EUROMAP63.EMAP100_ResultWriteJob.

I metodi:

■ **ns=2;s=EUROMAP63.EMAP100_GetStatus**

Output Arguments			
Name	Value	Data Type	Description
Status	<input type="text"/>	Int32	Stato connessione (0 = connesso)

Result

Call Close

■ **ns=2;s=EUROMAP63.EMAP100_Abort**

Output Arguments			
Name	Value	Data Type	Description
Abort	<input type="checkbox"/>	Boolean	Abort

Result

Call Close

■ **ns=2;s=EUROMAP63.EMAP100_SetPassive**

Input Arguments			
Name	Value	Data Type	Description
Passive	<input type="checkbox"/>	Boolean	Set Passive FTP

Output Arguments			
Name	Value	Data Type	Description
Esito	<input type="checkbox"/>	Boolean	Risultato Set

Result

Call Close

Mitsubishi

Nel TTSMAN, per i CN Mitsubishi, c'è il driver che prevede:

- la lettura dei dati macchina;
- il download e l'upload dei partprogram.

Nel Server OPC UA abbiamo le variabili:

- **ns=2;s=Mitsubishi.DataIn (lettura dei dati macchina)**

The screenshot shows a tree view of OPC UA variables. The root node is 'MitsubishiDataInVariableType'. It contains several sub-nodes: 'DescriptionDataIn', 'PARTCOUNT', 'PROGRAM', 'STATUS', and 'TypeMachineDataIn'. Below this, there is a 'MitsubishiDataInType Array[1]' node, which is expanded to show an array element '[0]' of type 'MitsubishiDataInType'. This array element contains the following variables: 'TypeMachine' (value: MIT100), 'DescriptionMachine' (value: Prova Mitsubishi), 'STATUS', 'PROGRAM', and 'PARTCOUNT'.

Ci sono poi le variabili semplici per ogni Machine, ad esempio:

- ns=2;s=Mitsubishi.MIT100_STATUS;
- ns=2;s=Mitsubishi.MIT100_PROGRAM;
- ns=2;s=Mitsubishi.MIT100_PARTCOUNT.

Il metodo:

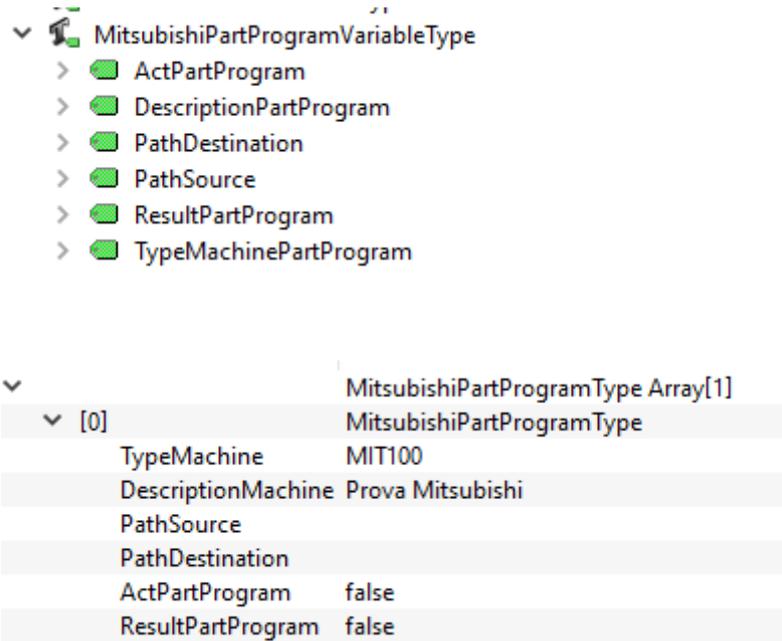
- **ns=2;s=Mitsubishi.MIT100_GetStatus**

The screenshot shows a dialog box titled 'Call MIT100_GetStatus on Machine_MIT100'. It has a 'Call' button and a 'Close' button. The dialog displays the 'Output Arguments' section with a table:

Name	Value	Data Type	Description
Status	<input type="text"/>	Int32	Stato connessione (0 = connesso)

Below the table is a 'Result' section, which is currently empty.

- ns=2;s=Mitsubishi.PartProgram (trasmissione dei partprogram)



Per inviare il partprogram bisogna:

- valorizzare il campo **PathSource**;
- valorizzare il campo **PathDestination** nel formato **PathDestination|D(U)**;
- settare il campo **ActPartProgram**;

il risultato si avrà nel campo **ResultPartProgram**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=Mitsubishi.MIT100_PathPartProgram, nel formato **PathSource||PathDestination|D(U)**;
- ns=2;s=Mitsubishi.MIT100_ResultPartProgram.

FAGOR

Nel TTSMAN, per i CN Mitsubishi, c'è il driver che prevede:

- la lettura dei dati macchina;
- il download e l'upload dei partprogram;
- la scrittura di variabili.

Nel Server OPC UA abbiamo le variabili:

- **ns=2;s=FAGOR.DataIn (lettura dei dati macchina)**

The screenshot shows a tree view of OPC UA variables. The root node is 'FAGORDataInVariableType', which is expanded to show several sub-variables: DescriptionDataIn, ERROR, MODE, PARTCOUNT, PARTPROGRAM, PARTTIME, and TypeMachineDataIn. Below this, a table shows the structure of the 'FAGORDataInType Array[1]' variable. The table has two columns: 'Name' and 'Value'. The rows are: TypeMachine (FAG100), DescriptionMachine (Test Fagor), MODE, PARTPROGRAM, PARTCOUNT, PARTTIME, and ERROR.

Name	Value
TypeMachine	FAG100
DescriptionMachine	Test Fagor
MODE	
PARTPROGRAM	
PARTCOUNT	
PARTTIME	
ERROR	

Ci sono poi le variabili semplici per ogni Machine, ad esempio:

- ns=2;s=FAGOR.FAG100_MODE;
- ns=2;s=FAGOR.FAG100_PARTPROGRAM;
- ns=2;s=FAGOR.FAG100_PARTCOUNT;
- ns=2;s=FAGOR.FAG100_PARTTIME;
- ns=2;s=FAGOR.FAG100_ERROR.

Il metodo:

- **ns=2;s=Mitsubishi.MIT100_GetStatus**

The screenshot shows a dialog box titled 'Call FAG100_GetStatus on Machine_FAG100'. It has a 'Call' button and a 'Close' button. The dialog is divided into 'Output Arguments' and 'Result' sections. The 'Output Arguments' section contains a table with the following data:

Name	Value	Data Type	Description
Status	<input type="text"/>	Int32	Stato connessione (0 = connesso)

- ns=2;s=FAGOR.PartProgram (trasmissione dei partprogram)

▼	FAGORPartProgramVariableType
>	ActPartProgram
>	DescriptionPartProgram
>	PathDestination
>	PathSource
>	ResultPartProgram
>	TypeMachinePartProgram
▼	FAGORPartProgramType Array[1]
▼	[0] FAGORPartProgramType
	TypeMachine FAG100
	DescriptionMachine Test Fagor
	PathSource
	PathDestination
	ActPartProgram false
	ResultPartProgram false

Per inviare il partprogram bisogna:

- valorizzare il campo **PathSource**;
- valorizzare il campo **PathDestination** nel formato **PathDestination|D(U)**;
- settare il campo **ActPartProgram**;

il risultato si avrà nel campo **ResultPartProgram**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=FAGOR.FAG100_PathPartProgram, nel formato **PathSource||PathDestination|D(U)**;
- ns=2;s=FAGOR.FAG100_ResultPartProgram.

- ns=2;s=FAGOR.Variable (scrittura di una variabile)

▼	FAGORVariableVariableType
>	ActVariable
>	DescriptionVariable
>	ResultVariable
>	TypeMachineVariable
>	VarVariable
▼	FAGORVariableType Array[1]
▼	[0] FAGORVariableType
	TypeMachine FAG100
	DescriptionMachine Test Fagor
	▼ Variable String Array[1]
	[0]
	ActVariable false
	ResultVariable false

Per scrivere le variabili bisogna:

- valorizzare il campo **Variable** nel formato **Nome=Valore**;
- settare il campo **ActVariable**;

il risultato si avrà nel campo **ResultVariable**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=FAGOR.FAG100_Variable, nel formato **Nome=Valore**;
- ns=2;s=FAGOR.FAG100_ResultVariable.

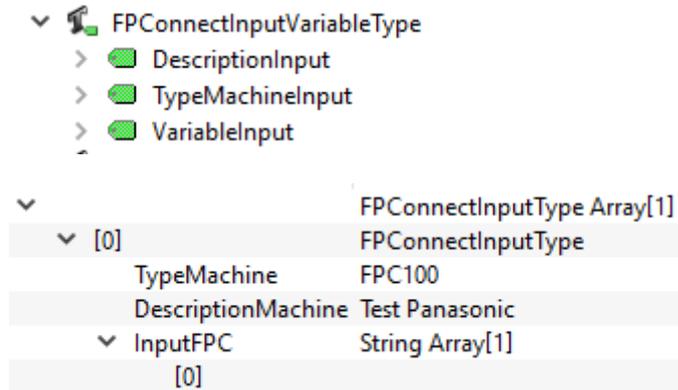
FPCConnect

Nel TTSMAN, per i Robot Panasonic, c'è il driver che prevede:

- la lettura di variabili;
- la scrittura di variabili.

Nel Server OPC UA abbiamo le variabili:

- **ns=2;s=FPCConnect.Input (lettura variabili)**

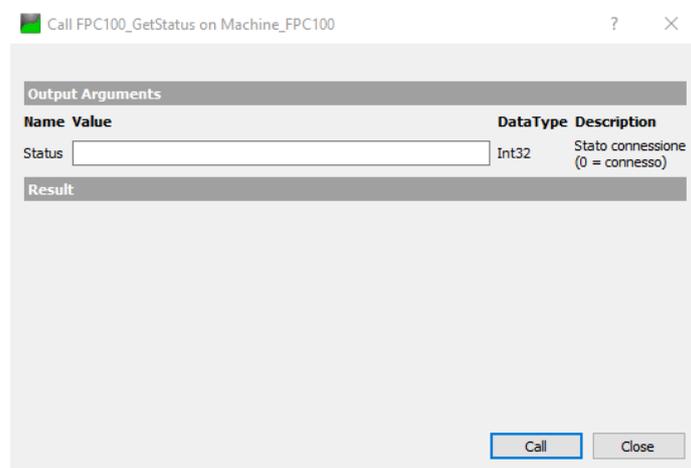


C'è poi la variabile semplice per ogni Machine, ad esempio:

- ns=2;s=FPCConnect.FPC100_InputFPC.

Il metodo:

- ns=2;s=FPCConnect.FPC100_GetStatus



- ns=2;s=FPCConnect.Output (scrittura variabili)

The screenshot shows a tree view of the FPCConnectOutputVariableType object. The root node is expanded to show its children: ActOutput, DescriptionOutput, ResultOutput, TypeMachineOutput, and VariableOutput. The TypeMachineOutput node is further expanded to show its properties: TypeMachine (FPC100), DescriptionMachine (Test Panasonic), and OutputFPC (String Array[1]). The OutputFPC node is expanded to show its elements: [0], ActOutput (false), and ResultOutput (false).

▼	FPCConnectOutputVariableType	
>	ActOutput	
>	DescriptionOutput	
>	ResultOutput	
>	TypeMachineOutput	
>	VariableOutput	
▼	FPCConnectOutputType Array[1]	
▼	[0]	FPCConnectOutputType
	TypeMachine	FPC100
	DescriptionMachine	Test Panasonic
▼	OutputFPC	String Array[1]
	[0]	
	ActOutput	false
	ResultOutput	false

Per scrivere le variabili bisogna:

- valorizzare il campo **OutputFPC** nel formato **Nome=Valore**;
- settare il campo **ActOutput**;

il risultato si avrà nel campo **ResultOutput**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=FPCConnect.FPC100_OutputFPC, nel formato **Nome=Valore**;
- ns=2;s=FPCConnect.FPC100_ResultOutput.

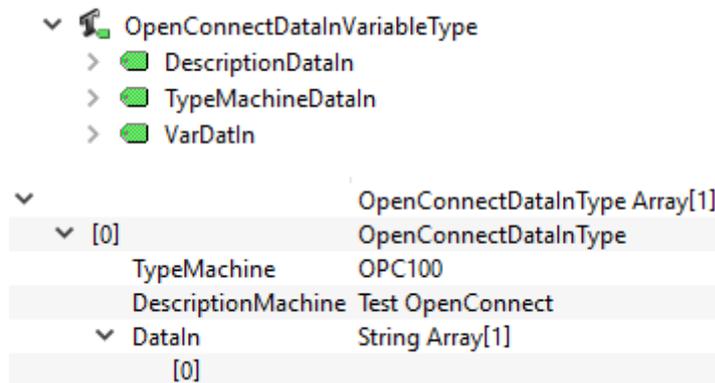
OpenConnect

Nel TTSMAN, per i CN OSAI, c'è il driver che prevede:

- la lettura dei dati macchina;
- il download e l'upload dei partprogram;
- la scrittura di variabili.

Nel Server OPC UA abbiamo le variabili:

- **ns=2;s=OpenConnect.DataIn (lettura dei dati macchina)**

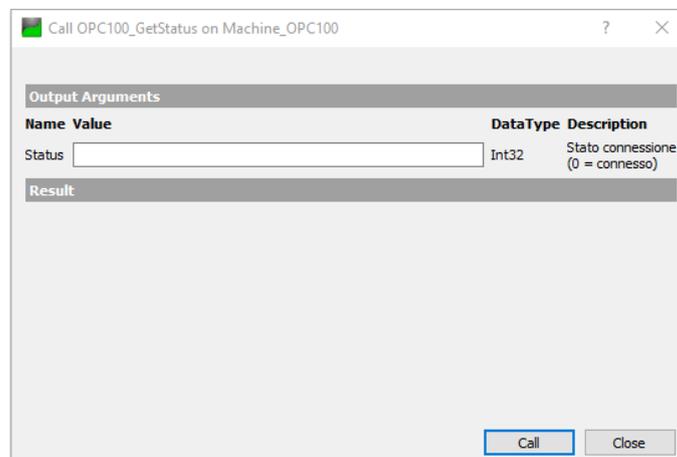


C'è poi la variabile semplice per ogni Machine, ad esempio:

- ns=2;s=OpenConnect.OPC100_DataIn.

Il metodo:

- ns=2;s=OpenConnect.OPC100_GetStatus



- ns=2;s=OpenConnect.PartProgram (trasmissione dei partprogram)

OpenConnectPartProgramVariableType	
> ActPartProgram	
> DescriptionPartProgram	
> PathDestination	
> PathSource	
> ResultPartProgram	
> TypeMachinePartProgram	
OpenConnectPartProgramType Array[1]	
> [0]	OpenConnectPartProgramType
TypeMachine	OPC100
DescriptionMachine	Test OpenConnect
PathSource	
PathDestination	
ActPartProgram	false
ResultPartProgram	false

Per inviare il partprogram bisogna:

- valorizzare il campo **PathSource**;
- valorizzare il campo **PathDestination**;
- settare il campo **ActPartProgram**;

il risultato si avrà nel campo **ResultPartProgram**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=OpenConnect.OPC100_PathPartProgram, nel formato **PathSource|PathDestination**;
- ns=2;s=OpenConnect.OPC100_ResultPartProgram.

- ns=2;s=OpenConnect.Variable (scrittura di una variabile)

OpenConnectVariableVariableType	
> ActVariable	
> DescriptionVariable	
> ResultVariable	
> TypeMachineVariable	
> VarVariable	
OpenConnectVariableType Array[1]	
> [0]	OpenConnectVariableType
TypeMachine	OPC100
DescriptionMachine	Test OpenConnect
> Variable	String Array[1]
> [0]	
ActVariable	false
ResultVariable	false

Per scrivere le variabili bisogna:

- valorizzare il campo **Variable** nel formato **Nome=Valore**;
- settare il campo **ActVariable**;

il risultato si avrà nel campo **ResultVariable**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=OpenConnect.OPC100_Variable, nel formato **Nome=Valore;**
- ns=2;s=OpenConnect.OPC100_ResultVariable.

HTTP

Nel TTSMAN, per le macchine che hanno un WebServer, c'è un browser in grado di fare il parsing su una pagina HTML, strutture XML o Json.

Nel Server OPC UA abbiamo la variabile:

- ns=2;s=HTTP.Response (Get - lettura dei dati macchina)

The screenshot shows a tree view of the variable `HTTPResponseVariableType`. It is expanded to show its sub-variables: `DescriptionResponse`, `ResponseVariable`, and `TypeMachineResponse`. Below this, an array of `HTTPResponseType` is shown, with the first element `[0]` expanded to show its fields: `TypeMachine` (value: `HTT100`), `DescriptionMachine` (value: `JSW`), and `Response` (value: `String Array[0]`).

C'è poi la variabile semplice per ogni Machine, ad esempio:

- ns=2;s=HTTP.HTT100_Response.

- ns=2;s=HTTP.Request (Post - scrittura dei dati macchina)

The screenshot shows a tree view of the variable `HTTPRequestVariableType`. It is expanded to show its sub-variables: `ActRequest`, `DescriptionRequest`, `RequestVariable`, `ResultRequest`, and `TypeMachineRequest`. Below this, an array of `HTTPRequestType` is shown, with the first element `[0]` expanded to show its fields: `TypeMachine` (value: `HTT100`), `DescriptionMachine` (value: `JSW`), `Request` (value: `String Array[0]`), `ActRequest` (value: `false`), and `ResultRequest` (value: `false`).

Per scrivere le variabili bisogna:

- valorizzare il campo **Request** nel formato **Url,Nome=Valore**;
Request è un array di stringhe, il primo elemento contiene l'Url;
- settare il campo **ActRequest**;

il risultato si avrà nel campo **ResultRequest**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=HTTP.HTT100_Request, nel formato **Url,Nome=Valore**;
Request è un array di stringhe, il primo elemento contiene l'Url;
- ns=2;s=HTTP.HTT100_ResultRequest.

Il metodo:

- `ns=2;s=HTTP.HTTP100_GetStatus`

Call HTTP100_GetStatus on Machine_HTTP100

Output Arguments

Name	Value	Data Type	Description
Status	<input type="text"/>	Int32	Stato connessione (0 = connesso)

Result

Call Close

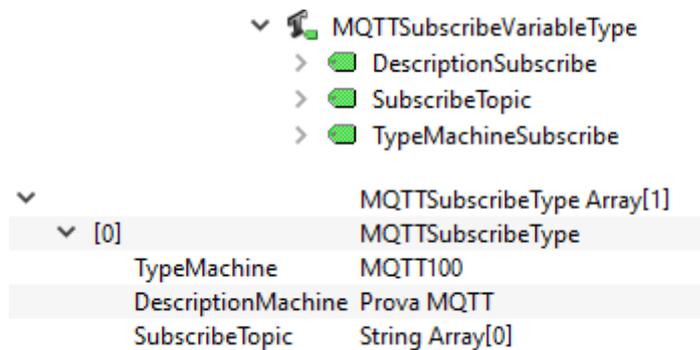
MQTT

Nel TTSMAN c'è un client MQTT che prevede.

- la lettura dei dati macchina (subscribe);
- l'invio di dati alle macchine (publish).

Nel Server OPC UA abbiamo le variabili:

- ns=2;s=MQTT.Subscribe (lettura dei dati macchina)



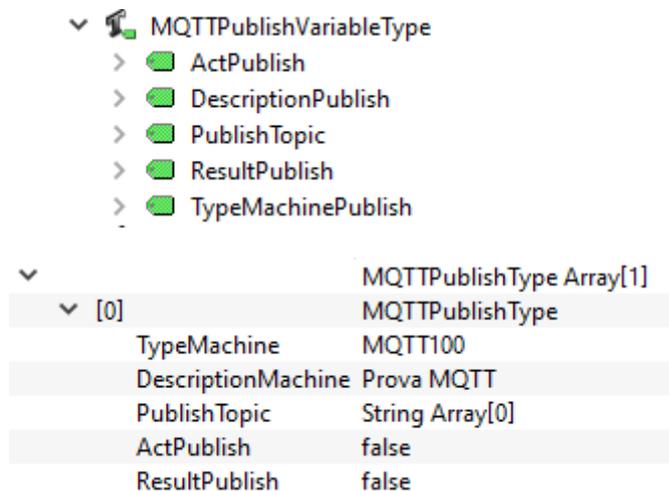
The screenshot shows a tree view of OPC UA variables. The root node is 'MQTTSubscribeVariableType', which is expanded to show its children: 'DescriptionSubscribe', 'SubscribeTopic', and 'TypeMachineSubscribe'. Below this, there is an array 'MQTTSubscribeType Array[1]' with one element '[0]'. This element is expanded to show its fields: 'TypeMachine' (value: MQTT100), 'DescriptionMachine' (value: Prova MQTT), and 'SubscribeTopic' (value: String Array[0]).

MQTTSubscribeVariableType	
DescriptionSubscribe	
SubscribeTopic	
TypeMachineSubscribe	
MQTTSubscribeType Array[1]	
[0]	MQTTSubscribeType
TypeMachine	MQTT100
DescriptionMachine	Prova MQTT
SubscribeTopic	String Array[0]

C'è poi la variabile semplice per ogni Machine, ad esempio:

- ns=2;s=MQTT.MQTT100_SubscribeTopic.

- ns=2;s=MQTT.Publish (invio di dati alle macchine)



The screenshot shows a tree view of OPC UA variables. The root node is 'MQTTPublishVariableType', which is expanded to show its children: 'ActPublish', 'DescriptionPublish', 'PublishTopic', 'ResultPublish', and 'TypeMachinePublish'. Below this, there is an array 'MQTTPublishType Array[1]' with one element '[0]'. This element is expanded to show its fields: 'TypeMachine' (value: MQTT100), 'DescriptionMachine' (value: Prova MQTT), 'PublishTopic' (value: String Array[0]), 'ActPublish' (value: false), and 'ResultPublish' (value: false).

MQTTPublishVariableType	
ActPublish	
DescriptionPublish	
PublishTopic	
ResultPublish	
TypeMachinePublish	
MQTTPublishType Array[1]	
[0]	MQTTPublishType
TypeMachine	MQTT100
DescriptionMachine	Prova MQTT
PublishTopic	String Array[0]
ActPublish	false
ResultPublish	false

Per inviare i topic bisogna:

- valorizzare il campo **PublishTopic** nel formato **Topic=Message**;
- settare il campo **ActPublish**;

il risultato si avrà nel campo **ResultPublish**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=MQTT.MQTT100_PublishTopic;
- ns=2;s=MQTT.MQTT100_ResultPublish.

Il metodo:

- ns=2;s=MQTT.MQTT100_GetStatus

Call MQTT100_GetStatus on Machine_MQTT100

Output Arguments			
Name	Value	Data Type	Description
Status	<input type="text"/>	Int32	Stato connessione (0 = connesso)

Result

Call Close

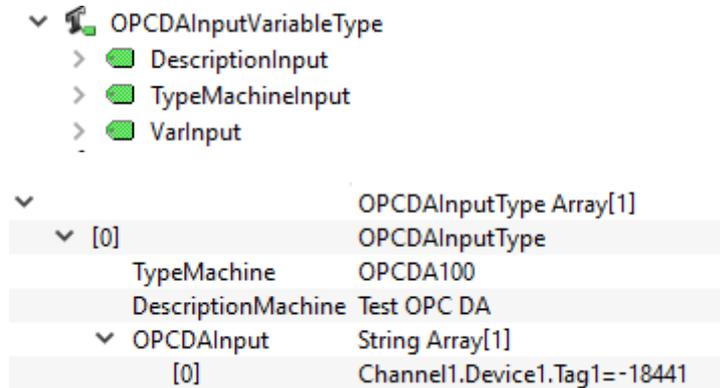
OPC DA

Nel TTSMAN c'è un client OPC DA che prevede.

- la lettura dei dati macchina;
- l'invio di dati alle macchine.

Nel Server OPC UA abbiamo le variabili:

- **ns=2;s=OPCDA.OPCDAInput (lettura dei dati macchina)**



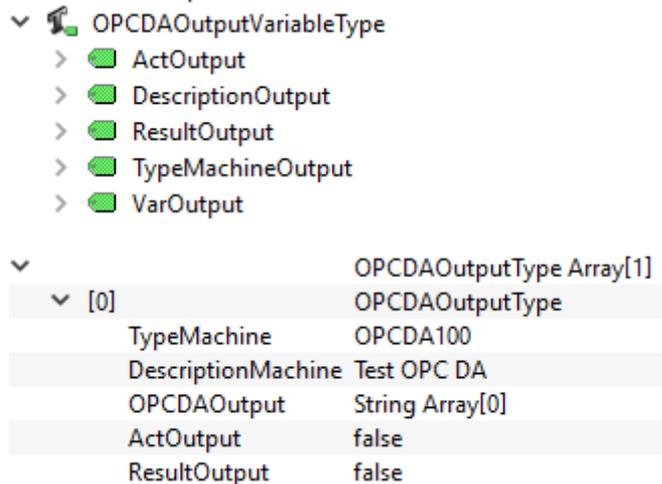
The screenshot shows a tree view of OPC UA variables. The root node is OPCDAInputVariableType, which is expanded to show its children: DescriptionInput, TypeMachineInput, and VarInput. Below this, the OPCDAInputType Array[1] is expanded to show its elements: [0], which is further expanded to show its fields: TypeMachine (OPCDA100), DescriptionMachine (Test OPC DA), OPCDAInput (String Array[1]), and [0] (Channel1.Device1.Tag1=-18441).

OPCDAInputVariableType	
> DescriptionInput	
> TypeMachineInput	
> VarInput	
OPCDAInputType Array[1]	
> [0]	OPCDAInputType
TypeMachine	OPCDA100
DescriptionMachine	Test OPC DA
OPCDAInput	String Array[1]
[0]	Channel1.Device1.Tag1=-18441

C'è poi la variabile semplice per ogni Machine, ad esempio:

- ns=2;s=OPCDA.OPCDA100_OPcDAInput.

- **ns=2;s=OPCDA.OPCDAOutput (invio di dati alle macchine)**



The screenshot shows a tree view of OPC UA variables. The root node is OPCDAOutputVariableType, which is expanded to show its children: ActOutput, DescriptionOutput, ResultOutput, TypeMachineOutput, and VarOutput. Below this, the OPCDAOutputType Array[1] is expanded to show its elements: [0], which is further expanded to show its fields: TypeMachine (OPCDA100), DescriptionMachine (Test OPC DA), OPCDAOutput (String Array[0]), ActOutput (false), and ResultOutput (false).

OPCDAOutputVariableType	
> ActOutput	
> DescriptionOutput	
> ResultOutput	
> TypeMachineOutput	
> VarOutput	
OPCDAOutputType Array[1]	
> [0]	OPCDAOutputType
TypeMachine	OPCDA100
DescriptionMachine	Test OPC DA
OPCDAOutput	String Array[0]
ActOutput	false
ResultOutput	false

Per inviare i dati bisogna:

- valorizzare il campo **OPCDAOutput**, nel formato **Variabile=Valore**;
- settare il campo **ActOutput**;

il risultato si avrà nel campo **ResultOutput**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=OPCDA.OPCDA100_OPcDAOutput, nel formato **Variabile=Valore**;
- ns=2;s=OPCDA.OPCDA100_ResultOutput.

Il metodo:

- ns=2;s=OPCDA.OPCDA100_GetStatus

Output Arguments		
Name	Value	Data Type Description
Status	<input type="text"/>	Int32 Stato connessione (0 = connesso)

Result

Call Close

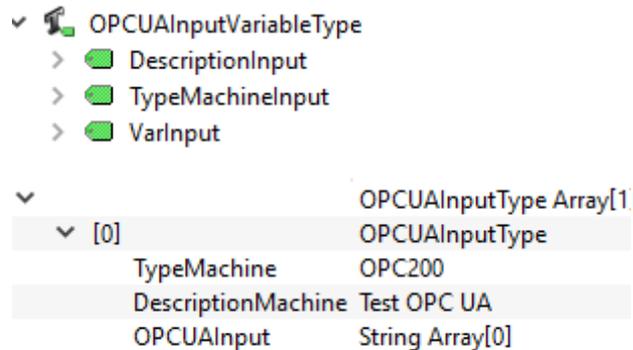
OPC UA

Nel TTSMAN c'è un client OPC UA che prevede.

- la lettura dei dati macchina;
- l'invio di dati alle macchine;
- la call di un metodo.

Nel Server OPC UA abbiamo le variabili:

- ns=2;s=OPCUA.OPCUAInput (lettura dei dati macchina)



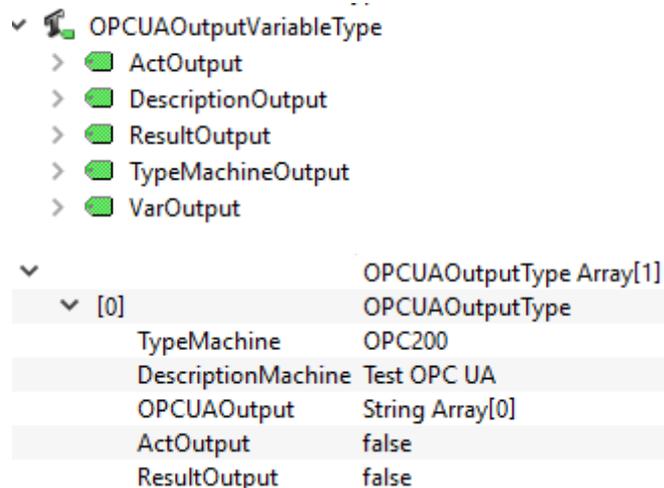
The screenshot shows a tree view of OPC UA variables. The root node is 'OPCUAInputVariableType'. It has four sub-nodes: 'DescriptionInput', 'TypeMachineInput', and 'VarInput'. Below this is an array 'OPCUAInputType Array[1]' with one element '[0]'. The '[0]' element is an 'OPCUAInputType' with three properties: 'TypeMachine' (value: OPC200), 'DescriptionMachine' (value: Test OPC UA), and 'OPCUAInput' (value: String Array[0]).

OPCUAInputVariableType	
> DescriptionInput	
> TypeMachineInput	
> VarInput	
OPCUAInputType Array[1]	
> [0]	OPCUAInputType
TypeMachine	OPC200
DescriptionMachine	Test OPC UA
OPCUAInput	String Array[0]

C'è poi la variabile semplice per ogni Machine, ad esempio:

- ns=2;s=OPCUA.OPC200_OPCUAInput.

- ns=2;s=OPCUA.OPCUAOutput (invio di dati alle macchine)



The screenshot shows a tree view of OPC UA variables. The root node is 'OPCUAOutputVariableType'. It has five sub-nodes: 'ActOutput', 'DescriptionOutput', 'ResultOutput', 'TypeMachineOutput', and 'VarOutput'. Below this is an array 'OPCUAOutputType Array[1]' with one element '[0]'. The '[0]' element is an 'OPCUAOutputType' with five properties: 'TypeMachine' (value: OPC200), 'DescriptionMachine' (value: Test OPC UA), 'OPCUAOutput' (value: String Array[0]), 'ActOutput' (value: false), and 'ResultOutput' (value: false).

OPCUAOutputVariableType	
> ActOutput	
> DescriptionOutput	
> ResultOutput	
> TypeMachineOutput	
> VarOutput	
OPCUAOutputType Array[1]	
> [0]	OPCUAOutputType
TypeMachine	OPC200
DescriptionMachine	Test OPC UA
OPCUAOutput	String Array[0]
ActOutput	false
ResultOutput	false

Per inviare i dati bisogna:

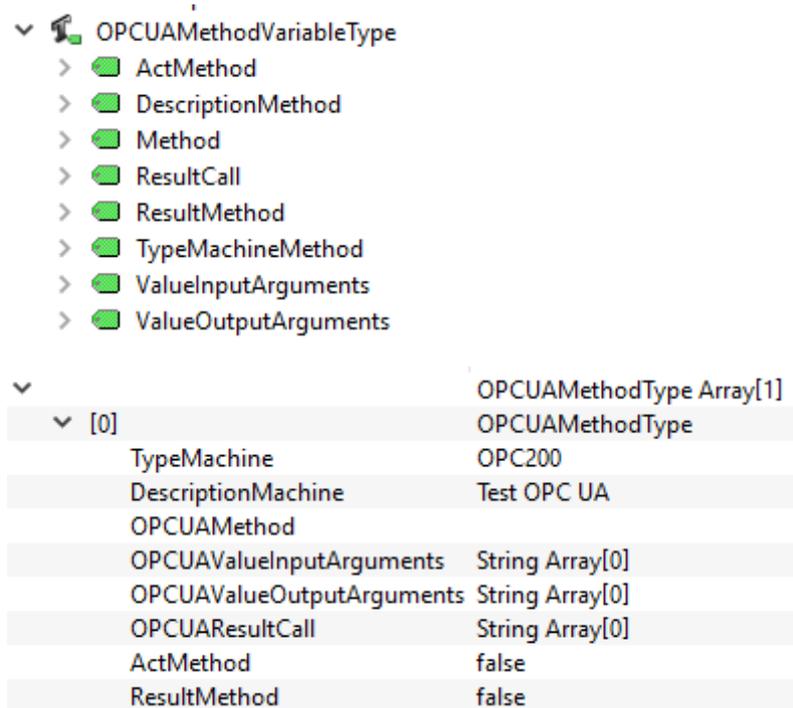
- valorizzare il campo **OPCUAOutput**, nel formato **Variabile||Valore[||IndexRange]**;
- settare il campo **ActOutput**;

il risultato si avrà nel campo **ResultOutput**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=OPCUA.OPC200_OPCUAOutput, nel formato **Variabile||Valore[||IndexRange]**;
- ns=2;s=OPCUA.OPC200_ResultOutput.

- ns=2;s=OPCUA.OPCUAMethod (call di un metodo)



The screenshot shows a tree view on the left and a table on the right. The tree view is expanded to show 'OPCUAMethodVariableType' with sub-items: ActMethod, DescriptionMethod, Method, ResultCall, ResultMethod, TypeMachineMethod, ValueInputArguments, and ValueOutputArguments. The table on the right is titled 'OPCUAMethodType Array[1]' and contains the following data:

OPCUAMethodType	
TypeMachine	OPC200
DescriptionMachine	Test OPC UA
OPCUAMethod	
OPCUAValueInputArguments	String Array[0]
OPCUAValueOutputArguments	String Array[0]
OPCUAResultCall	String Array[0]
ActMethod	false
ResultMethod	false

Per la call bisogna:

- valorizzare il campo **OPCUAMethod**;
- valorizzare il campo **OPCUAValueInputArguments**, nel formato **array valore**;
- settare il campo **ActMethod**;

il risultato si avrà nel campo **ResultMethod**. Il campo **OPCUAValueOutputArguments** conterrà i parametri restituiti e il campo **OPCUAResultCall** la descrizione del risultato.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=OPCUA.OPC200_OPCLUAMethod, nel formato **Method||ValueInputArguments; ValueInputArguments** nel formato **valore|valore|...**
- ns=2;s=OPCUA.OPC200_ResultMethod;
- ns=2;s=OPCUA.OPC200_OPCLUAValueOutputArguments;
- ns=2;s=OPCUA.OPC200_ResultCall.

I metodi:

■ ns=2;s=OPCUA.OPC200_GetStatus

Output Arguments			
Name	Value	Data Type	Description
Status	<input type="text"/>	Int32	Stato connessione (0 = connesso)

Result

Call Close

■ ns=2;s=OPCUA.OPC200_SetOPCUA

Input Arguments			
Name	Value	Data Type	Description
Variation	<input type="checkbox"/>	Boolean	Dati variati
Json	<input type="checkbox"/>	Boolean	Conversione Json

Output Arguments			
Name	Value	Data Type	Description
Esito	<input type="checkbox"/>	Boolean	Risultato Set

Result

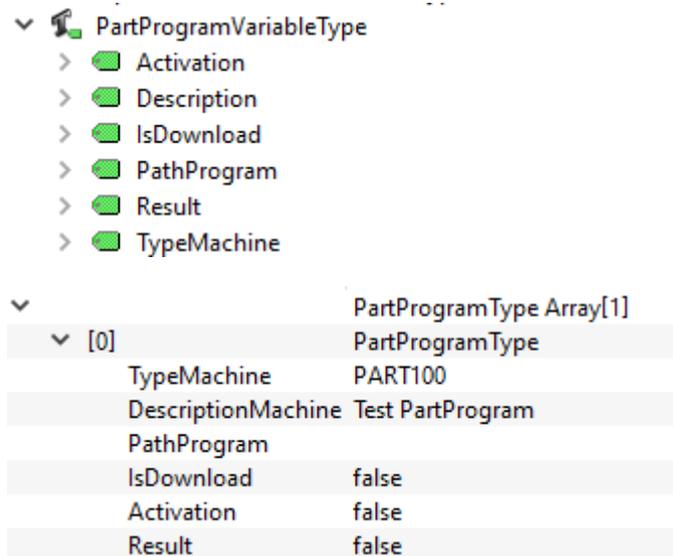
Call Close

PartProgram

Il TTSMAN permette, genericamente, di gestire i PartProgram tramite cartelle condivise o convertitori Ethernet/RS232C (IF304, Moxa, ecc.).

Nel Server OPC UA abbiamo la variabile:

- ns=2;s=PartProgram.PartProgram



The screenshot shows a tree view of OPC UA variables. The root is 'PartProgramVariableType', which contains several sub-variables: 'Activation', 'Description', 'IsDownload', 'PathProgram', 'Result', and 'TypeMachine'. Below this, there is an array 'PartProgramType Array[1]' with one element '[0]'. This element is a 'PartProgramType' object with the following properties:

TypeMachine	PART100
DescriptionMachine	Test PartProgram
PathProgram	
IsDownload	false
Activation	false
Result	false

Per inviare il partprogram bisogna:

- valorizzare il campo **PathProgram**, nel formato richiesto;
- settare il campo **IsDownload**;
- settare il campo **Activation**;

il risultato si avrà nel campo **Result**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=PartProgram.PART100_PathProgram, nel formato **PathProgram||0_1 (1=Download)**;
- ns=2;s=PartProgram.PART100_Result.

I metodi:

■ ns=2;s=PartProgram.PART100_GetStatus

Call PART100_GetStatus on Machine_PART100

Output Arguments			
Name	Value	Data Type	Description
Status	<input type="text"/>	Int32	Stato connessione (0 = connesso)

Result

Call Close

■ ns=2;s=PartProgram.PART100_AbortUpLoad

Call PART100_AbortUpLoad on Machine_PART100

Output Arguments			
Name	Value	Data Type	Description
AbortUpLoad	<input type="checkbox"/>	Boolean	AbortUpLoad

Result

Call Close

File

Il TTSMAN permette di gestire i file presenti su una macchina di produzione prevedendo:

- la lettura dei file;
- l'invio dei file;
- la cancellazione dei file.

Nel Server OPC UA abbiamo le variabili:

- **ns=2;s=File.FileRead**

The screenshot displays the OPC UA variable browser for the namespace ns=2;s=File.FileRead. It shows a hierarchy starting with FileReadVariableType, which contains three sub-variables: DescriptionReadFile, RecordFile, and TypeMachineFile. Below this, a FileReadType Array[2] is expanded to show two entries, [0] and [1]. Each entry contains a TypeMachine, a DescriptionMachine, and a RecordFile array. Entry [0] is for TypeMachine FTP100 and DescriptionMachine Prova FTP, with a RecordFile array of 12 elements. Entry [1] is for TypeMachine FIL100 and DescriptionMachine Prova Shared Folder, with a RecordFile array of 841 elements.

Index	TypeMachine	DescriptionMachine	RecordFile
[0]	FTP100	Prova FTP	String Array[12] [0] REQTTTS.JOB JOB reqlog RESPONSE "reqlog.log"; [1] REQTTTS.JOB REPORT reqdata [2] REQTTTS.JOB REWRITE "reqdata.dat" [3] REQTTTS.JOB START IMMEDIATE [4] REQTTTS.JOB STOP NEVER [5] REQTTTS.JOB CYCLIC SHOT 1 [6] REQTTTS.JOB SESSIONS 1 [7] REQTTTS.JOB PARAMETERS [8] REQTTTS.JOB DATE, [9] REQTTTS.JOB TIME, [10] REQTTTS.JOB COUNT; [11] REQTTTS.JOB EOF
[1]	FIL100	Prova Shared Folder	String Array[841] [0] install.log Call: 1247 [1] install.log Delete: "C:\Users\Aniello\AppData\Local... [2] install.log CreateDirectory: "C:\Users\Aniello\AppData... [3] install.log CreateDirectory: "C:\Users\Aniello\AppData... [4] install.log File: overwriteflag=1, allowskipfilesflag=... [5] install.log File: wrote 6656 to "C:\Users\Aniello\AppData...

La variabile semplice per ogni Machine è ad esempio:

ns=2;s=File.FIL100_RecordFileRead.

- ns=2;s=File.FileWrite

The screenshot shows a tree view on the left with the following items:

- FileWriteVariableType
 - ActFileWrite
 - DescriptionFileWrite
 - PathWriteFile
 - RecordFileWrite
 - ResultFileWrite
 - TypeMachineFileWrite

Below the tree view is a table representing the data for FileWriteType Array[2].

FileWriteType Array[2]	
[0]	FileWriteType
TypeMachine	FIL100
DescriptionMachine	Prova Shared Folder
PathFile	
RecordFile	String Array[0]
ActWriteFile	false
ResultWriteFile	false
[1]	FileWriteType
TypeMachine	FTP100
DescriptionMachine	Prova FTP
PathFile	
RecordFile	String Array[0]
ActWriteFile	false
ResultWriteFile	false

Per inviare i file bisogna:

- valorizzare il campo **PathFile**, nel formato richiesto;
- valorizzare il campo **RecordFile**;
- settare il campo **ActWriteFile**;

il risultato si avrà nel campo **ResultWriteFile**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=File.FIL100_RecordFileWrite, il primo elemento dell'Array deve essere il path del file da scrivere;
- ns=2;s=File.FIL100_ResultWriteFile.

- ns=2;s=File.FileDelete

```
FileDeleteVariableType
├── ActFileDelete
├── DescriptionFileDelete
├── PathDeleteFile
├── ResultFileDelete
├── TypeMachineFileDelete
└── FileDeleteType Array[2]
    ├── [0] FileDeleteType
        ├── TypeMachine FIL100
        ├── DescriptionMachine Prova Shared Folder
        ├── PathFile
        ├── ActDeleteFile false
        └── ResultDeleteFile false
    └── [1] FileDeleteType
        ├── TypeMachine FTP100
        ├── DescriptionMachine Prova FTP
        ├── PathFile
        ├── ActDeleteFile false
        └── ResultDeleteFile false
```

Per cancellare i file bisogna:

- valorizzare il campo **PathFile**, nel formato richiesto;
- settare il campo **ActDeleteFile**;

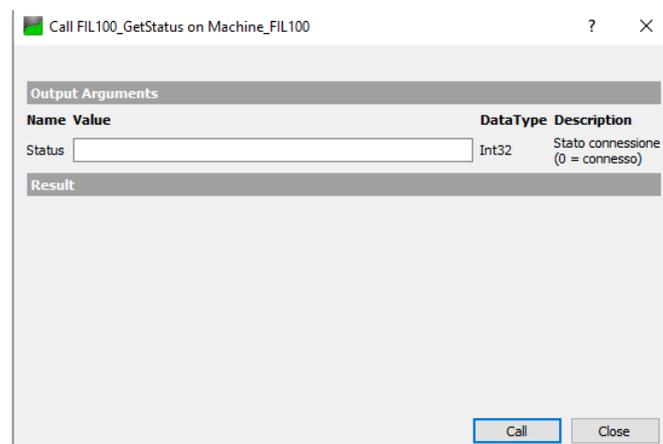
il risultato si avrà nel campo **ResultDeleteFile**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=File.FIL100_PathDeleteFile;
- ns=2;s=File.FIL100_ResultDeleteFile.

Il metodo:

- ns=2;s=File.FIL100_GetStatus



DBTable

Il TTSMAN permette di leggere le tabelle da un DB presente su una macchina di produzione.

Nel Server OPC UA abbiamo le variabili:

- ns=2;s=DBTABLE.DBTABLERead

DBTABLEReadTableVariableType	
> DBTABLERows	
> Description	
> TypeMachineDB	
DBTABLEReadType Array[2]	
[0] DBTABLEReadType	
TypeMachine	SQLS100
DescriptionMachine	Prova SqlServer
DBTABLERows	DBTABLEColumnsType Array[8]
[0] DBTABLEColumnsType	
DBTABLEColumns	String Array[7]
[0]	varOPC_id=1
[1]	varOPC_row=1
[2]	varOPC_item=COUNTER
[3]	varOPC_varType=UnsignedShort
[4]	varOPC_access=2
[5]	varOPC_comment=Contapezzi
[6]	varOPC_type=PLC10
[1] DBTABLEColumnsType	
DBTABLEColumns	String Array[7]
[0]	varOPC_id=2
[1]	varOPC_row=1
[2]	varOPC_item=COUNTER
[3]	varOPC_varType=UnsignedInteger
[4]	varOPC_access=4

La variabile semplice per ogni Machine è ad esempio:

- ns=2;s=DBTABLE.NOSQL100_DBTABLERows.

- ns=2;s=DBTABLE.DBTABLEInsertRow

```
DBTABLEInsertRowVariableType
  > ActInsertRow
  > DBTABLEInsertRow
  > InsertDescription
  > ResultInsertRow
  > TypeInsertMachineDB

DBTABLEInsertRowType Array[2]
  [0] DBTABLEInsertRowType
    TypeMachine NOSQL100
    DescriptionMachine Prova MongoDB
    DBTABLEInsertRow String Array[0]
    ActInsertRow false
    ResultInsertRow false
  [1] DBTABLEInsertRowType
    TypeMachine SQLS100
    DescriptionMachine Prova SqlServer
    DBTABLEInsertRow String Array[0]
    ActInsertRow false
    ResultInsertRow false
```

Per aggiungere una riga bisogna:

- valorizzare il campo **DBTableInsertROW**, nel formato Nome=Valore;
- settare il campo **ActInsertRow**;

il risultato si avrà nel campo **ResultInsertRow**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=DBTABLE.NOSQL100_InsertRow;
- ns=2;s=DBTABLE.NOSQL100_ResultInsertRow.

- ns=2;s=DBTABLE.DBTABLEUpdateField

```
DBTABLEUpdateFieldVariableType
  > ActUpdateFieldVar
  > ResultUpdateFieldVar
  > TypeUpdateMachineDB
  > UpdateDescription
  > UpdateFieldVar
  > UpdateKeyVar

DBTABLEUpdateFieldType Array[2]
  [0] DBTABLEUpdateFieldType
    TypeMachine NOSQL100
    DescriptionMachine Prova MongoDB
    Key
    Field
    ActUpdateField false
    ResultUpdateField false
  [1] DBTABLEUpdateFieldType
    TypeMachine SQLS100
    DescriptionMachine Prova SqlServer
    Key
    Field
    ActUpdateField false
    ResultUpdateField false
```

Per aggiornare un campo di una riga bisogna:

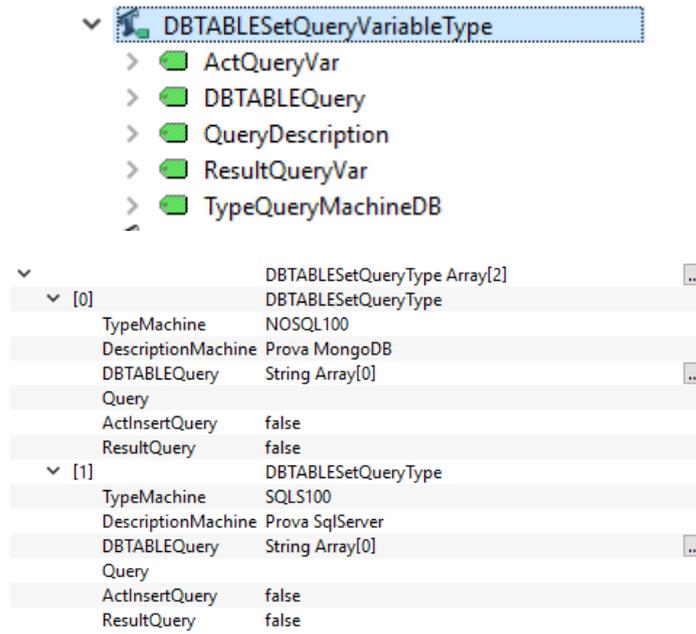
- valorizzare i campi **Key e Field**, nel formato Nome=Valore;
- settare il campo **ActUpdateField**;

il risultato si avrà nel campo **ResultUpdateField**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=DBTABLE.NOSQL100_UpdateField;
- ns=2;s=DBTABLE.NOSQL100_ResultUpdateField.

- ns=2;s=DBTABLE.DBTABLESetupQuery



Per definire una query bisogna:

- valorizzare il campo **DBTABLEQuery**, nel formato Nome,
- valorizzare il campo **Query** con la stringa definente,
- settare il campo **ActInsertQuery**;

il risultato si avrà nel campo **ResultQuery**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=DBTABLE.NOSQL100_SetQuery (l'ultimo elemento dell'array deve contenere la stringa definente);
- ns=2;s=DBTABLE.NOSQL100_ResultUpdateField.

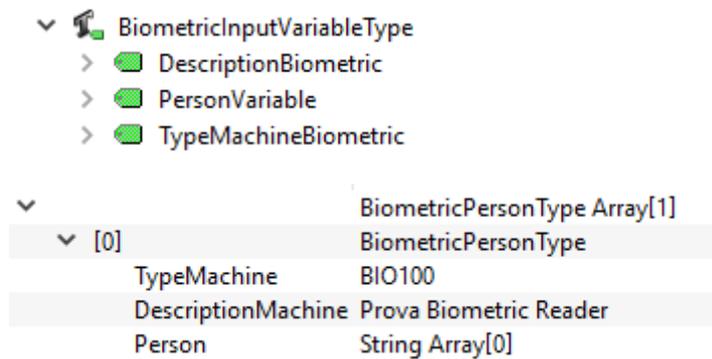
BiometricReader

Nel TTSMAN, per i lettori biometrici, c'è il driver che prevede:

- la lettura online delle timbrature;
- l'invio di comandi (cancellazione di un'immagine, cancellazione delle timbrature locali, ecc.).

Nel Server OPC UA abbiamo le variabili:

- ns=2;s=BiometricReader.BiometricInput (lettura delle timbrature)



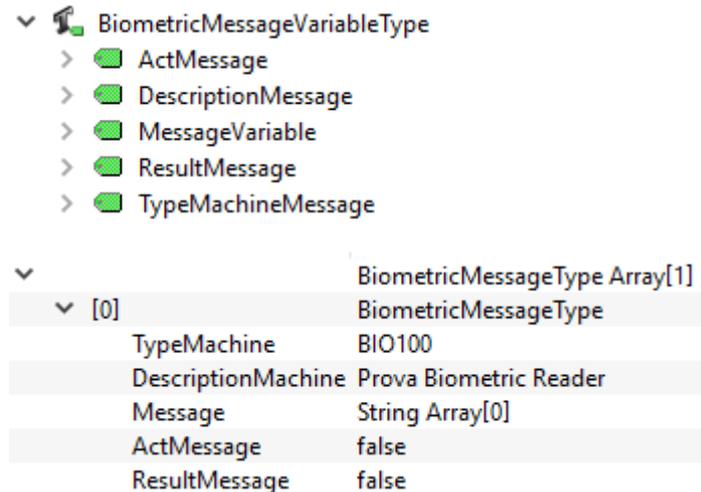
The screenshot shows a tree view of the variable `BiometricInputVariableType`. It is expanded to show its sub-variables: `DescriptionBiometric`, `PersonVariable`, and `TypeMachineBiometric`. Below this, an array `BiometricPersonType Array[1]` is expanded to show its element `[0]`, which is a `BiometricPersonType` object. The fields of this object are: `TypeMachine` (value: BIO100), `DescriptionMachine` (value: Prova Biometric Reader), and `Person` (value: String Array[0]).

BiometricInputVariableType	
> DescriptionBiometric	
> PersonVariable	
> TypeMachineBiometric	
BiometricPersonType Array[1]	
> [0]	BiometricPersonType
TypeMachine	BIO100
DescriptionMachine	Prova Biometric Reader
Person	String Array[0]

La variabile semplice per ogni Machine è ad esempio:

ns=2;s=BiometricReader.BIO100_Person.

- ns=2;s=BiometricReader.Message



The screenshot shows a tree view of the variable `BiometricMessageVariableType`. It is expanded to show its sub-variables: `ActMessage`, `DescriptionMessage`, `MessageVariable`, `ResultMessage`, and `TypeMachineMessage`. Below this, an array `BiometricMessageType Array[1]` is expanded to show its element `[0]`, which is a `BiometricMessageType` object. The fields of this object are: `TypeMachine` (value: BIO100), `DescriptionMachine` (value: Prova Biometric Reader), `Message` (value: String Array[0]), `ActMessage` (value: false), and `ResultMessage` (value: false).

BiometricMessageVariableType	
> ActMessage	
> DescriptionMessage	
> MessageVariable	
> ResultMessage	
> TypeMachineMessage	
BiometricMessageType Array[1]	
> [0]	BiometricMessageType
TypeMachine	BIO100
DescriptionMachine	Prova Biometric Reader
Message	String Array[0]
ActMessage	false
ResultMessage	false

Per inviare un comando bisogna:

- valorizzare il campo **Message**, nel formato richiesto;
- settare il campo **ActMessage**;

il risultato si avrà nel campo **ResultMessage**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=BiometricReader.BIO100_Message;
- ns=2;s=BiometricReader.BIO100_ResultMessage.

Il metodo:

- **ns=2;s=BiometricReader.BIO100_GetStatus**

Call BIO100_GetStatus on Machine_BIO100

Output Arguments

Name	Value	DataType	Description
Status	<input type="text"/>	Int32	Stato connessione (0 = connesso)

Result

Call Close

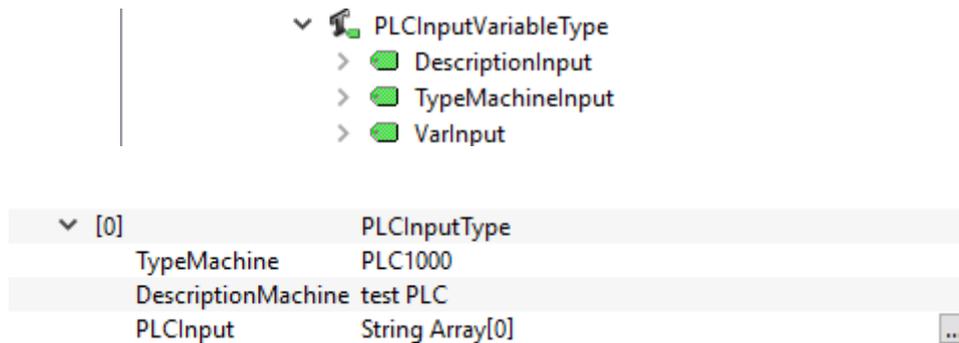
PLC

Nel TTSMAN, per i PLC(MODBUS, S7, ecc., ci sono i protocolli per lo scambio dati che prevedono:

- la lettura delle variabili;
- la scrittura delle variabili.

Nel Server OPC UA abbiamo le variabili:

- ns=2;s=PLC.PLCInput



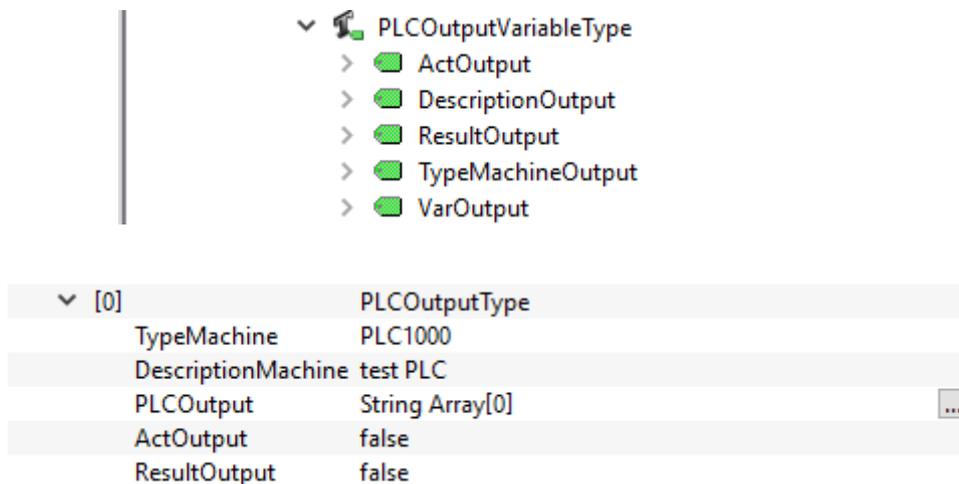
The screenshot shows a tree view of the variable PLCInputVariableType. It is expanded to show its sub-variables: DescriptionInput, TypeMachineInput, and VarInput. Below the tree, a table displays the values for the instance [0].

▼ [0]	PLCInputType
TypeMachine	PLC1000
DescriptionMachine	test PLC
PLCInput	String Array[0]

La variabile semplice per ogni Machine è ad esempio:

ns=2;s=PLC.PLC100_PLCInput.

- ns=2;s=PLC.PLCOutput



The screenshot shows a tree view of the variable PLCOutputVariableType. It is expanded to show its sub-variables: ActOutput, DescriptionOutput, ResultOutput, TypeMachineOutput, and VarOutput. Below the tree, a table displays the values for the instance [0].

▼ [0]	PLCOutputType
TypeMachine	PLC1000
DescriptionMachine	test PLC
PLCOutput	String Array[0]
ActOutput	false
ResultOutput	false

Per inviare dei dati bisogna:

- valorizzare il campo **PLCOutput**, nel formato **variabile=valore**;
- settare il campo **ActOutput**;

il risultato si avrà nel campo **ResultOutput**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=PLC.PLC100_PLCOutput, nel formato **variabile=valore**;
- ns=2;s=PLC.PLC100_ResultOutput.

I metodi:

■ ns=2;s=PLC.PLC100_GetStatus

Call PLC100_GetStatus on Machine_PLC100

Output Arguments			
Name	Value	Data Type	Description
Status	<input type="text"/>	Int32	Stato connessione (0 = connesso)

Result

Call Close

■ ns=2;s=PLC.PLC100_SetPLC

Call PLC100_SetPLC on Machine_PLC100

Input Arguments			
Name	Value	Data Type	Description
Variation	<input type="checkbox"/>	Boolean	Dati variati
Json	<input type="checkbox"/>	Boolean	Conversione Json

Output Arguments			
Name	Value	Data Type	Description
Esito	<input type="checkbox"/>	Boolean	Risultato Set

Result

Call Close

■ ns=2;s=PLC.PLC100_AbilLog

Call PLC100_AbilLog on Machine_PLC100

Input Arguments			
Name	Value	Data Type	Description
Log	<input type="checkbox"/>	Boolean	Abilitazione Log

Result

Call Close

■ ns=2;s=PLC.PLC100_SetS7200 (per S7 200)

Call PLC100_SetS7200 on Machine_PLC100 ? X

Input Arguments			
Name	Value	Data Type	Description
S7200	<input type="checkbox"/>	Boolean	S7200 Siemens

Result

Call Close

■ ns=2;s=PLC.PLC100_SetSwap (per MODBUS)

Call PLC100_SetSwap on Machine_PLC100 ? X

Input Arguments			
Name	Value	Data Type	Description
Swap	<input type="checkbox"/>	Boolean	Modbus swap

Result

Call Close

■ ns=2;s=PLC.PLC100_GetStation (per MODBUS multipoint)

Call PLC100_GetStation on Machine_PLC100 ? X

Output Arguments			
Name	Value	Data Type	Description
StationAct	<input type="text"/>	Int32	GetStation Corrente

Result

Call Close

■ ns=2;s=PLC.PLC100_SetStation (per MODBUS multipoint)

Call PLC100_SetStation on Machine_PLC100 ? X

Input Arguments

Name	Value	DataType	Description
Station	<input type="text"/>	Int32	SetStation

Result

Call Close

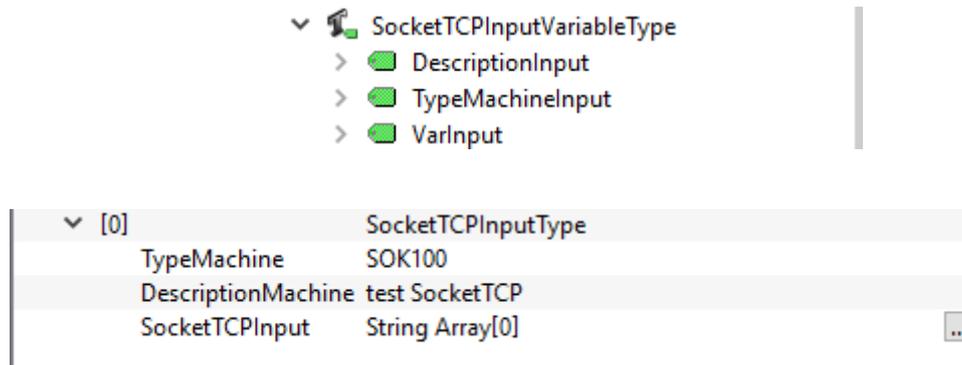
SocketTCP

Nel TTSMAN permette lo scambio dati tramite un socket TCP/IP sono previsti:

- la lettura dei dati nel formato previsto;
- la scrittura dei dati nel formato previsto.

Nel Server OPC UA abbiamo le variabili:

- ns=2;s=SocketTCP.SocketTCPInput



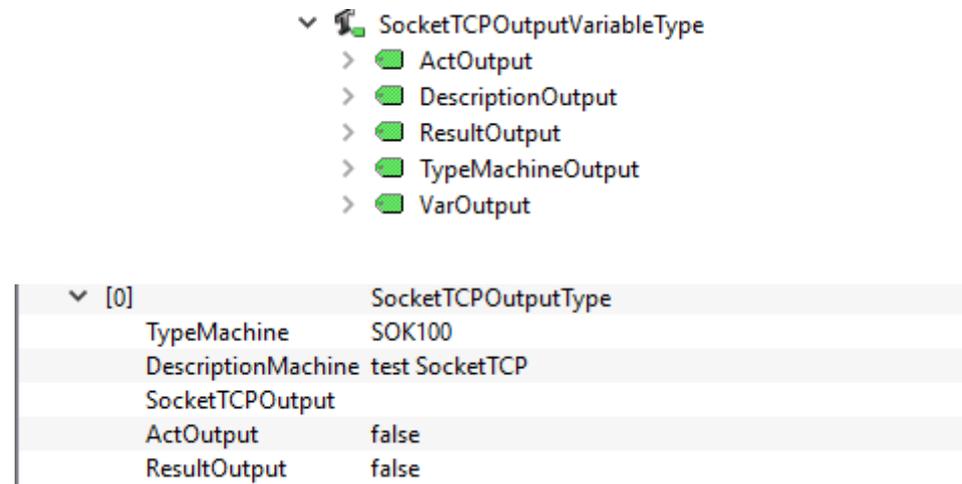
The screenshot shows a tree view of the variable `SocketTCPInputVariableType`. It contains three sub-variables: `DescriptionInput`, `TypeMachineInput`, and `VarInput`. Below this, a table displays the values for the `[0]` instance of `SocketTCPInputType`.

▼ [0]	SocketTCPInputType
TypeMachine	SOK100
DescriptionMachine	test SocketTCP
SocketTCPInput	String Array[0]

La variabile semplice per ogni Machine è ad esempio:

ns=2;s=SocketTCP.SOK100_SocketTCPInput.

- ns=2;s=SocketTCP.SocketTCPOutput



The screenshot shows a tree view of the variable `SocketTCPOutputVariableType`. It contains five sub-variables: `ActOutput`, `DescriptionOutput`, `ResultOutput`, `TypeMachineOutput`, and `VarOutput`. Below this, a table displays the values for the `[0]` instance of `SocketTCPOutputType`.

▼ [0]	SocketTCPOutputType
TypeMachine	SOK100
DescriptionMachine	test SocketTCP
SocketTCPOutput	
ActOutput	false
ResultOutput	false

Per inviare dei dati bisogna:

- valorizzare il campo **ActOutput**;
- settare il campo **ActOutput**;

il risultato si avrà nel campo **ResultOutput**.

Le variabili semplici per ogni Machine sono ad esempio:

- ns=2;s=SocketTCP.SOK100_SocketTCPOutput;
- ns=2;s=SocketTCP.SOK100_ResultOutput.

I metodi:

■ ns=2;s=SocketTCP.SOK100_GetStatus

Output Arguments			
Name	Value	Data Type	Description
Status	<input type="text"/>	Int32	Stato connessione (0 = connesso)

Result

Call Close

■ ns=2;s=SocketTCP.SOK100_SetSocket

Input Arguments			
Name	Value	Data Type	Description
Variation	<input type="checkbox"/>	Boolean	Dati variati
Json	<input type="checkbox"/>	Boolean	Conversione Json

Output Arguments			
Name	Value	Data Type	Description
Esito	<input type="checkbox"/>	Boolean	Risultato Set

Result

Call Close

TTSObject

L'IF304 ha un potente firmware che permette di scambiare i dati con i dispositivi di fabbrica tramite segnali ON/OFF e analogici, porte ethernet e seriali (PLC, Reader RFID, Bilance, ecc.), IOT.

Nel Server OPC UA le variabili dell'IF304 sono relative a:

- Stato della connessione del dispositivo;
- Digital I/O;
- RS232C_1 (porta seriale RS232C);
- RS232C_2 (porta seriale RS485);
- Analog Input;
- Barcode;
- Watch;
- VAC (emulazione TTS30M);
- VSC (emulazione TTS30M).

Connection

Nel Server OPC UA abbiamo la variabile:

- ns=3;s=Connection.Connection

▼	 ConnectionVariableType	
>	 DescriptionDevice	
>	 IsConnected	
>	 PHAddress	
>	 TypeDevice	
▼		ConnectionType Array[4]
▼	[0]	ConnectionType
	TypeDevice	IF304
	DescriptionDevice	Test Server OPCUA
	PHAddress	1
	IsConnected	false
▼	[1]	ConnectionType
	TypeDevice	IF304
	DescriptionDevice	Test OPCUA
	PHAddress	132
	IsConnected	false
▼	[2]	ConnectionType
	TypeDevice	TTS30M
	DescriptionDevice	Test VSC
	PHAddress	129
	IsConnected	false
▼	[3]	ConnectionType
	TypeDevice	TTS30M
	DescriptionDevice	Test VAC
	PHAddress	128
	IsConnected	false

La variabile semplice per ogni IF304 è ad esempio:

ns=3;s=Connection.IF304_132_IsConnected.

Digital I/O

L'IF304 è dotato di una porta Digital Input/Output e nel Server OPC UA abbiamo le variabili:

- ns=3;s=Digital.DigitalInput (lettura dello stato di una porta)

▼	 DigitalInputVariableType
>	 DescriptionDeviceIn
>	 PHAddressIn
>	 PortIn
>	 StateIn
>	 TypeDeviceIn
▼	DigitalInputType Array[4]
▼	[0] DigitalInputType
	TypeDeviceIn IF304
	DescriptionDeviceIn IF300
	PHAddressIn 132
	PortIn 2
	StateIn false
▼	[1] DigitalInputType
	TypeDeviceIn IF304
	DescriptionDeviceIn IF300
	PHAddressIn 132
	PortIn 3
	StateIn false
▼	[2] DigitalInputType
	TypeDeviceIn IF304
	DescriptionDeviceIn IF300
	PHAddressIn 132
	PortIn 4
	StateIn false
▼	[3] DigitalInputType
	TypeDeviceIn IF304
	DescriptionDeviceIn MS100
	PHAddressIn 1
	PortIn 2
	StateIn false

La variabile semplice per ogni IF304 è ad esempio:

ns=3;s=Digital.IF304_132_Input (lo stato delle quattro porte).

▼	Value	Boolean Array[4]
	[0]	false
	[1]	false
	[2]	false
	[3]	false

- ns=3;s=Digital.DigitalCounter (lettura dei contapezzi configurati)

▼		DigitalCounterVariableType
>		ActiveCount
>		DescriptionDeviceCount
>		PHAddressCount
>		PortCount
>		TypeDeviceCount
>		ValueCount
▼		DigitalCounterType Array[4]
▼	[0]	DigitalCounterType
	TypeDeviceCount	IF304
	DescriptionDeviceCount	IF300
	PHAddressCount	132
	PortCount	1
	ValueCount	0
	ActiveCount	false
▼	[1]	DigitalCounterType
	TypeDeviceCount	IF304
	DescriptionDeviceCount	MS100
	PHAddressCount	1
	PortCount	1
	ValueCount	0
	ActiveCount	false
▼	[2]	DigitalCounterType
	TypeDeviceCount	IF304
	DescriptionDeviceCount	MS100
	PHAddressCount	1
	PortCount	4
	ValueCount	0
	ActiveCount	false
▼	[3]	DigitalCounterType
	TypeDeviceCount	IF304
	DescriptionDeviceCount	MS100
	PHAddressCount	1
	PortCount	3
	ValueCount	0
	ActiveCount	false

La variabile ActiveCount è “true” se c’è attività.

La variabile semplice per ogni IF304 è ad esempio:

ns=3;s=Digital.IF304_132_Counter (il valore dei contatori, nel formato: "Port_Value_Activity").

▼	Value	String Array[4]
	[0]	1_0_false
	[1]	
	[2]	
	[3]	

- ns=3;s=Digital.DigitalThresholdCounter (soglia dei contapezzi configurati)

✓  DigitalTHCounterVariableType	
>  DescriptionDeviceTH	
>  PHAddressTH	
>  PortTH	
>  TypeDeviceTH	
>  ValueTH	
▼	DigitalTHCounterType Array[4]
▼ [0]	DigitalTHCounterType
TypeDeviceTH	IF304
DescriptionDeviceTH	IF300
PHAddressTH	132
PortTH	1
ValueTH	50
▼ [1]	DigitalTHCounterType
TypeDeviceTH	IF304
DescriptionDeviceTH	MS100
PHAddressTH	1
PortTH	1
ValueTH	50
▼ [2]	DigitalTHCounterType
TypeDeviceTH	IF304
DescriptionDeviceTH	MS100
PHAddressTH	1
PortTH	3
ValueTH	50
▼ [3]	DigitalTHCounterType
TypeDeviceTH	IF304
DescriptionDeviceTH	MS100
PHAddressTH	1
PortTH	4
ValueTH	50

La variabile semplice per ogni IF304 è ad esempio:

ns=3;s=Digital.IF304_132_ThresholdCounter (la soglia dei contatori, nel formato: "Port_Value").

▼ Value	String Array[4]
[0]	1_50
[1]	
[2]	
[3]	

- ns=3;s=Digital.DigitalOutput (attivazione statica delle porte di output)

▼		DigitalOutputVariableType
>		DescriptionDeviceOut
>		PHAddressOut
>		PortOut
>		StateOut
>		TypeDeviceOut
▼	Value	DigitalOutputType Array[8]
>	[0]	DigitalOutputType
>	[1]	DigitalOutputType
>	[2]	DigitalOutputType
>	[3]	DigitalOutputType
▼	[4]	DigitalOutputType
	TypeDeviceOut	IF304
	DescriptionD...	IF300
	PHAddressOut	132
	PortOut	4
	StateOut	true
▼	[5]	DigitalOutputType
	TypeDeviceOut	IF304
	DescriptionD...	IF300
	PHAddressOut	132
	PortOut	1
	StateOut	true
▼	[6]	DigitalOutputType
	TypeDeviceOut	IF304
	DescriptionD...	IF300
	PHAddressOut	132
	PortOut	3
	StateOut	true
▼	[7]	DigitalOutputType
	TypeDeviceOut	IF304
	DescriptionD...	IF300
	PHAddressOut	132
	PortOut	2
	StateOut	true

La variabile semplice per ogni IF304 è ad esempio:

ns=3;s=Digital.IF304_132_Output.

▼	Value	Boolean Array[4]
	[0]	true
	[1]	true
	[2]	true
	[3]	true

- ns=3;s=Digital.DigitalCmdOutput (attivazione dinamica delle porte di output)

✓  DigitalCmdOutputVariableType	
>  Act_DisactCmd	
>  DescriptionDeviceCmd	
>  EnableCommand	
>  PHAddressCmd	
>  PortCmd	
>  TimeCmd	
>  TypeDeviceCmd	

▼ Value	DigitalCmdOutputType Array[8]
> [0]	DigitalCmdOutputType
> [1]	DigitalCmdOutputType
> [2]	DigitalCmdOutputType
> [3]	DigitalCmdOutputType
▼ [4]	DigitalCmdOutputType
TypeDeviceCmd	IF304
DescriptionDeviceCmd	IF300
PHAddressCmd	132
PortCmd	1
Act_DisactCmd	true
TimeCmd	0
EnableCommand	false
> [5]	DigitalCmdOutputType
> [6]	DigitalCmdOutputType
▼ [7]	DigitalCmdOutputType
TypeDeviceCmd	IF304
DescriptionDeviceCmd	IF300
PHAddressCmd	132
PortCmd	4
Act_DisactCmd	true
TimeCmd	0
EnableCommand	false

Per inviare un comando bisogna:

- settare il campo **EnableCommand**;
- settare/resettare il campo **Act_DisactCmd**, per attivare o disattivare la porta di uscita;
- valorizzare il campo TimeCmd, per definire la durata dell'impulso.

La variabile semplice per ogni IF304 è ad esempio:

- ns=3;s=Digital.IF304_132_CmdOutput, nel formato **Port_Act/Disact_Time**.

▼ Value	String Array[4]
[0]	1_true_0
[1]	2_true_0
[2]	3_true_0
[3]	4_true_0

- ns=3;s=Digital.DigitalCommand (comandi)

- ▼  DigitalCommandVariableType
 - >  ActCommand
 - >  DescriptionDeviceCommand
 - >  DeviceLock
 - >  PHAddressCommand
 - >  Poll
 - >  PortCommand
 - >  ReadCounter
 - >  TypeDeviceCommand

▼ Value	DigitalCommandType Array[2]
▼ [0]	DigitalCommandType
TypeDevice	IF304
DescriptionD...	IF300
PHAddress	132
DeviceLock	false
Poll	false
ReadCounter	false
ActCommand	false
▼ [1]	DigitalCommandType
TypeDevice	IF304
DescriptionD...	MS100
PHAddress	1
DeviceLock	false
Poll	false
ReadCounter	false
ActCommand	false

Per inviare un comando bisogna:

- settare il campo **ActCommand**;
- settare_resettare i campi **DeviceLock**, **Poll**, **ReadCounter**.

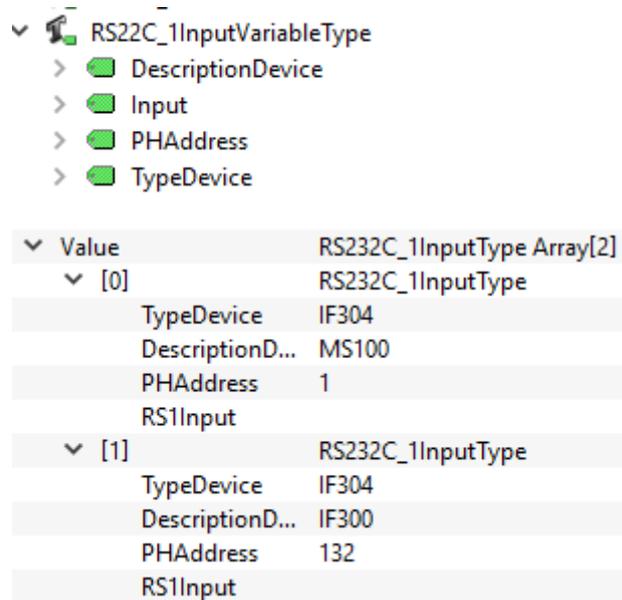
Le variabili semplici per ogni IF304 sono ad esempio:

- ns=3;s=Digital.IF304_132_DeviceLock;
- ns=3;s=Digital.IF304_132_Poll;
- ns=3;s=Digital.IF304_132_ReadCounter.

RS232C_1

L'IF304 è dotato di una porta Seriale RS232C e nel Server OPC UA abbiamo le variabili:

- ns=3;s=RS232C_1.RS232C_1Input (input da canale seriale)



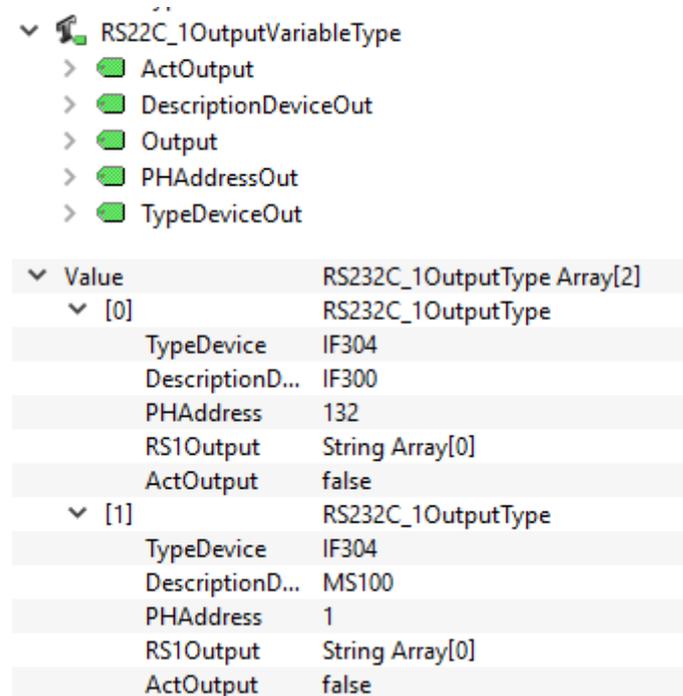
The screenshot shows a tree view of the variable RS232C_1InputVariableType. It has sub-nodes: DescriptionDevice, Input, PAddress, and TypeDevice. Below is a table of its value, which is an array of two RS232C_1InputType objects.

Value	RS232C_1InputType Array[2]
[0]	RS232C_1InputType
TypeDevice	IF304
DescriptionD...	MS100
PAddress	1
RS1Input	
[1]	RS232C_1InputType
TypeDevice	IF304
DescriptionD...	IF300
PAddress	132
RS1Input	

La variabile semplice per ogni IF304 è ad esempio:

- ns=3;s=RS232C_1.IF304_132_RS1Input.

- ns=3;s=RS232C_1.RS232C_1Output (output verso canale seriale)



The screenshot shows a tree view of the variable RS232C_1OutputVariableType. It has sub-nodes: ActOutput, DescriptionDeviceOut, Output, PAddressOut, and TypeDeviceOut. Below is a table of its value, which is an array of two RS232C_1OutputType objects.

Value	RS232C_1OutputType Array[2]
[0]	RS232C_1OutputType
TypeDevice	IF304
DescriptionD...	IF300
PAddress	132
RS1Output	String Array[0]
ActOutput	false
[1]	RS232C_1OutputType
TypeDevice	IF304
DescriptionD...	MS100
PAddress	1
RS1Output	String Array[0]
ActOutput	false

Per inviare bisogna:

- settare il campo **ActOutput**;
- valorizzare il campo **RS1Output**.

La variabile semplice per ogni IF304 è ad esempio:

- ns=3;s=RS232C_1.IF304_132_RS1Output.

- ns=3;s=RS232C_1.RS232C_1CmdOutput (lettura temporizzata da canale seriale, PLC, ecc.)

The screenshot shows a tree view of a variable named 'RS22C_2CmdOutputVariableType'. It has several sub-variables: ActCmdOutput, Command, DescriptionDeviceCmdOut, PHAddressCmdOut, TimeCmdOut, and TypeDeviceCmdOut. Below the tree view is a table showing the 'Value' of the variable, which is an array of two 'RS232C_1CmdOutputType' objects.

Value	RS232C_1CmdOutputType Array[2]
[0]	RS232C_1CmdOutputType
TypeDeviceC...	IF304
DescriptionD...	IF300
PHAddressC...	132
RS1Command	
TimeCmd	0
ActCommand	false
[1]	RS232C_1CmdOutputType
TypeDeviceC...	IF304
DescriptionD...	MS100
PHAddressC...	1
RS1Command	
TimeCmd	0
ActCommand	false

Per attivare bisogna:

- settare il campo **ActCommand**;
- valorizzare il campo **RS1Command**;
- valorizzare il campo **TimeCmd**.

La variabile semplice per ogni IF304 è ad esempio:

- ns=3;s=RS232C_1.IF304_132_RS1Command, nel formato: **Command||Timer**.

- ns=3;s=RS232C_1.RS232C_1Command (comandi)

The screenshot shows a tree view of a variable named 'RS22C_2CommandVariableType'. It has several sub-variables: ActAux1, ActBreakIn, ActCommand, DescriptionDeviceCmd, DeviceLock, PHAddressCmd, and TypeDeviceCmd.

▼ Value	RS232C_1CommandType Array[2]
▼ [0]	RS232C_1CommandType
TypeDevice	IF304
DescriptionDevice	MS100
PHAddress	1
RS1DeviceLock	false
RS1ActAux1	false
Rs1ActBreakIn	false
ActCommand	false
▼ [1]	RS232C_1CommandType
TypeDevice	IF304
DescriptionDevice	IF300
PHAddress	132
RS1DeviceLock	false
RS1ActAux1	false
Rs1ActBreakIn	false
ActCommand	false

Per attivare bisogna:

- settare il campo **ActCommand**;
- settare_resettare i campi **RS1DeviceLock**, **RS1ActAux1**, **RS1ActBreakIn**.

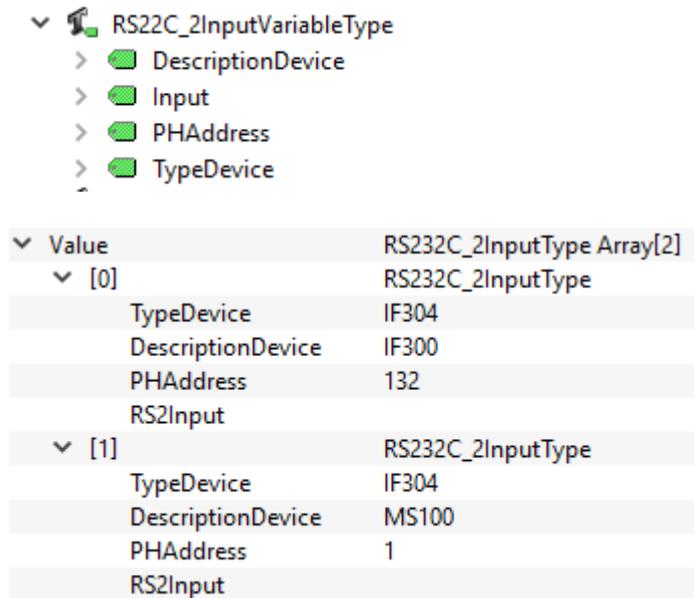
Le variabili semplici per ogni IF304 sono ad esempio:

- ns=3;s=RS232C_1.IF304_132_RS1DeviceLock;
- ns=3;s=RS232C_1.IF304_132_RS1ActAux1;
- ns=3;s=RS232C_1.IF304_132_Rs1ActBreakIn.

RS232C_2

L'IF304 è dotato di una porta Seriale RS485 e nel Server OPC UA abbiamo le variabili:

- ns=3;s=RS232C_2.RS232C_2Input (input da canale seriale)



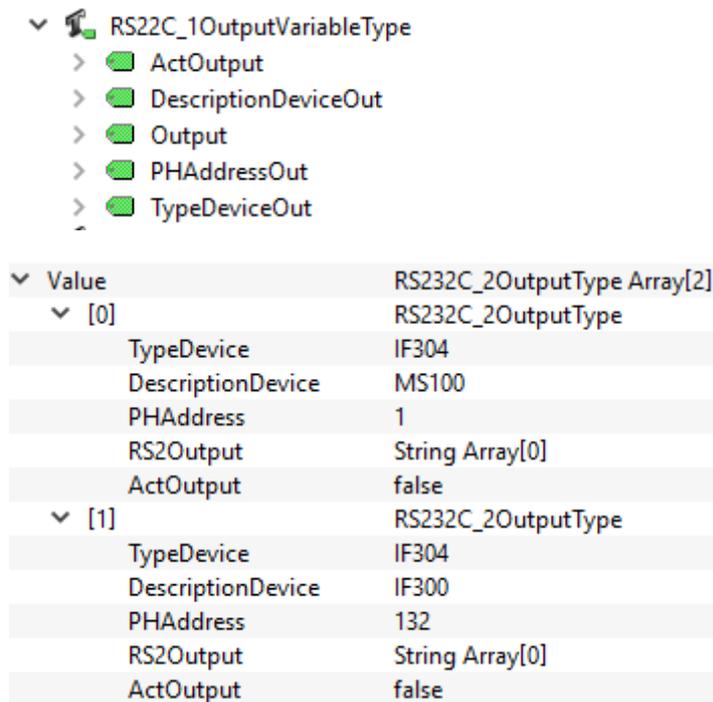
The screenshot shows a tree view of the variable `RS232C_2InputVariableType`. It has four sub-variables: `DescriptionDevice`, `Input`, `PHAddress`, and `TypeDevice`. Below this is a table showing the value of the variable as an array of two `RS232C_2InputType` objects.

Value	RS232C_2InputType Array[2]
[0]	RS232C_2InputType
TypeDevice	IF304
DescriptionDevice	IF300
PHAddress	132
RS2Input	
[1]	RS232C_2InputType
TypeDevice	IF304
DescriptionDevice	MS100
PHAddress	1
RS2Input	

La variabile semplice per ogni IF304 è ad esempio:

■ ns=3;s=RS232C_2.IF304_132_RS2Input.

- ns=3;s=RS232C_2.RS232C_2Output (output verso canale seriale)



The screenshot shows a tree view of the variable `RS232C_2OutputVariableType`. It has five sub-variables: `ActOutput`, `DescriptionDeviceOut`, `Output`, `PHAddressOut`, and `TypeDeviceOut`. Below this is a table showing the value of the variable as an array of two `RS232C_2OutputType` objects.

Value	RS232C_2OutputType Array[2]
[0]	RS232C_2OutputType
TypeDevice	IF304
DescriptionDevice	MS100
PHAddress	1
RS2Output	String Array[0]
ActOutput	false
[1]	RS232C_2OutputType
TypeDevice	IF304
DescriptionDevice	IF300
PHAddress	132
RS2Output	String Array[0]
ActOutput	false

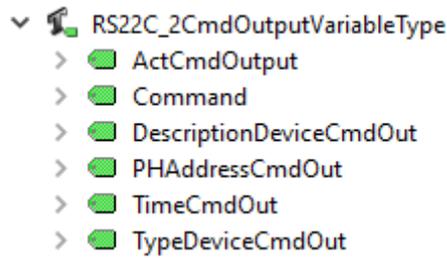
Per inviare bisogna:

- settare il campo **ActOutput**;
- valorizzare il campo **RS2Output**.

La variabile semplice per ogni IF304 è ad esempio:

- ns=3;s=RS232C_2.IF304_132_RS2Output.

- ns=3;s=RS232C_2.RS232C_2CmdOutput (lettura temporizzata da canale seriale, PLC, ecc.)



▼ Value	RS232C_2CmdOutputType Array[2]
▼ [0]	RS232C_2CmdOutputType
TypeDeviceCmd	IF304
DescriptionDeviceCmd	MS100
PHAddressCmd	1
RS2Command	
TimeCmd	0
ActCommand	false
▼ [1]	RS232C_2CmdOutputType
TypeDeviceCmd	IF304
DescriptionDeviceCmd	IF300
PHAddressCmd	132
RS2Command	
TimeCmd	0
ActCommand	false

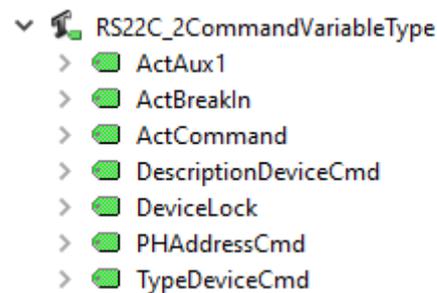
Per attivare bisogna:

- settare il campo **ActCommand**;
- valorizzare il campo **RS2Command**;
- valorizzare il campo **TimeCmd**.

La variabile semplice per ogni IF304 è ad esempio:

- ns=3;s=RS232C_2.RS232C_2CmdOutput, nel formato: **Command||Timer**.

- ns=3;s=RS232C_2.RS232C_2Command (comandi)



Value	RS232C_2CommandType Array[2]
[0]	RS232C_2CommandType
TypeDevice	IF304
DescriptionDevice	IF300
PHAddress	132
RS2DeviceLock	false
RS2ActAux1	false
RS2ActBreakIn	false
ActCommand	false
[1]	RS232C_2CommandType
TypeDevice	IF304
DescriptionDevice	MS100
PHAddress	1
RS2DeviceLock	false
RS2ActAux1	false
RS2ActBreakIn	false
ActCommand	false

Per attivare bisogna:

- settare il campo **ActCommand**;
- settare/resettare i campi **RS2DeviceLock**, **RS2ActAux1**, **RS2ActBreakIn**.

Le variabili semplici per ogni IF304 sono ad esempio:

- ns=3;s=RS232C_2.IF304_132_RS2DeviceLock;
- ns=3;s=RS232C_2.IF304_132_RS2ActAux1;
- ns=3;s=RS232C_2.IF304_132_Rs2ActBreakIn.

Analog Input

L'IF304 è dotato di input analogico (A/D converter 12 bit) e nel Server OPC UA abbiamo le variabili:

- ns=3;s=AnalogInput.AnalogInput (input)

▼	 AnalogInputVariableType	
>	 DescriptionDevice	
>	 Input	
>	 PHAddress	
>	 TypeDevice	
▼	Value	AnalogInputType Array[2]
▼	[0]	AnalogInputType
	TypeDevice	IF304
	DescriptionDevice	IF300
	PHAddress	132
>	AnalogInput	AnalogChannelType Array[1]
▼	[1]	AnalogInputType
	TypeDevice	IF304
	DescriptionDevice	MS100
	PHAddress	1
>	AnalogInput	AnalogChannelType Array[1]

La variabile semplice per ogni IF304 è ad esempio:

- ns=3;s=AnalogInput.IF304_132_AnalogInput.

- ns=3;s=AnalogInput.AnalogCommand (comandi)

▼	 AnalogCommandVariableType	
>	 AnalogActCommand	
>	 AnalogDeviceLock	
>	 AnalogPoll	
>	 DescriptionDeviceCmd	
>	 PHAddressCmdmd	
>	 TypeDeviceCmd	
▼	Value	AnalogCommandType Array[2]
▼	[0]	AnalogCommandType
	TypeDevice	IF304
	DescriptionDevice	MS100
	PHAddress	1
	AnalogDeviceLock	false
	AnalogPoll	false
	ActCommand	false
▼	[1]	AnalogCommandType
	TypeDevice	IF304
	DescriptionDevice	IF300
	PHAddress	132
	AnalogDeviceLock	false
	AnalogPoll	false
	ActCommand	false

Per attivare bisogna:

- settare il campo **ActCommand**;
- settare_resettare i campi **AnalogDeviceLock, AnalogPoll**.

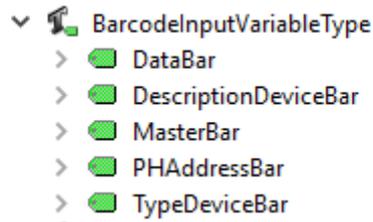
Le variabili semplici per ogni IF304 sono ad esempio:

- ns=3;s=AnalogInput.IF304_132_AnalogDeviceLock;
- ns=3;s=AnalogInput.IF304_132_AnalogPoll.

Barcode

L'IF304 è dotato di una porta USB su cui collegare un lettore barcode HID e nel Server OPC UA abbiamo le variabili:

- ns=3;s=Barcode.BarcodeInput (input)

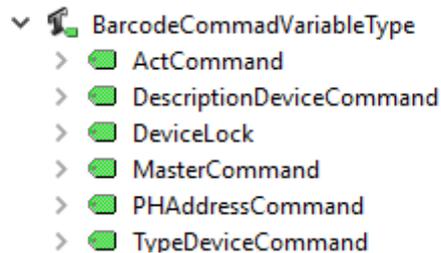


[0]	BarcodeInputType
TypeDeviceBar	IF304
DescriptionDeviceBar	
MasterBar	1
PHAddressBar	1
DataBar	

La variabile semplice per ogni IF304 è ad esempio:

- ns=3;s=Barcode.IF304_1_1_DataBar.

- ns=3;s=Barcode.BarcodeCommand (comandi)



[0]	BarcodeCommandType
TypeDevice	IF304
DescriptionDevice	
MasterAddress	1
PHAddress	1
DeviceLock	false
ActCommand	false

Per attivare bisogna:

- settare il campo **ActCommand**;
- settare_resettare il campo **DeviceLock**.

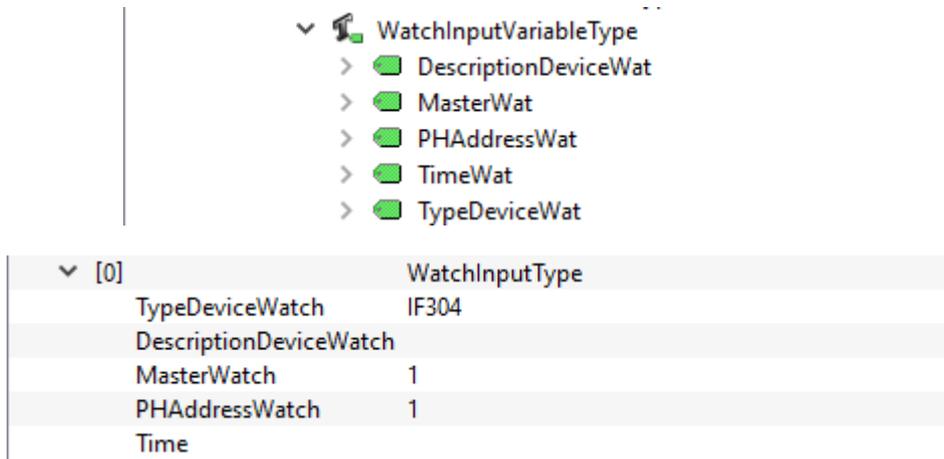
Le variabili semplici per ogni IF304 sono ad esempio:

- ns=3;s=Barcode.IF304_1_1_DeviceLock.

Watch

L'IF304 è dotato orologio_datario e nel Server OPC UA abbiamo le variabili:

- ns=3;s=Watch.WatchInput (input)



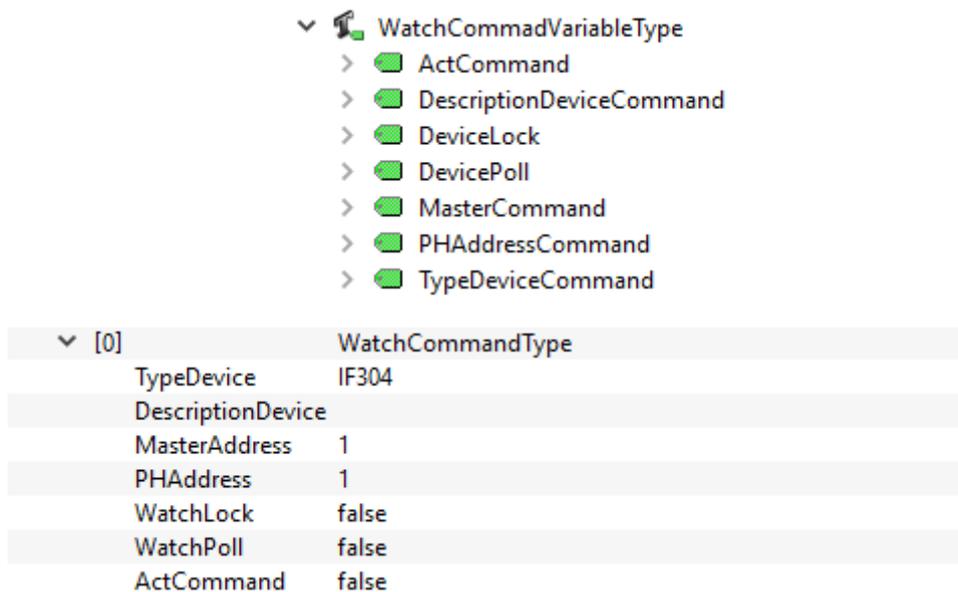
The screenshot shows a tree view of the 'WatchInputVariableType' structure. It contains several sub-variables: DescriptionDeviceWat, MasterWat, PHAddressWat, TimeWat, and TypeDeviceWat. Below this, a table shows the instance values for the '[0]' array element.

[0] WatchInputType	
TypeDeviceWatch	IF304
DescriptionDeviceWatch	
MasterWatch	1
PHAddressWatch	1
Time	

Le variabili semplici per ogni IF304 sono ad esempio:

■ ns=3;s=Watch.IF304_1_1_TimeWat.

- ns=3;s=Watch.WatchCommand (comandi)



The screenshot shows a tree view of the 'WatchCommandVariableType' structure. It contains several sub-variables: ActCommand, DescriptionDeviceCommand, DeviceLock, DevicePoll, MasterCommand, PHAddressCommand, and TypeDeviceCommand. Below this, a table shows the instance values for the '[0]' array element.

[0] WatchCommandType	
TypeDevice	IF304
DescriptionDevice	
MasterAddress	1
PHAddress	1
WatchLock	false
WatchPoll	false
ActCommand	false

Per attivare bisogna:

- settare il campo **ActCommand**;
- settare_resettare i campi **WatchLock, WatchPoll**.

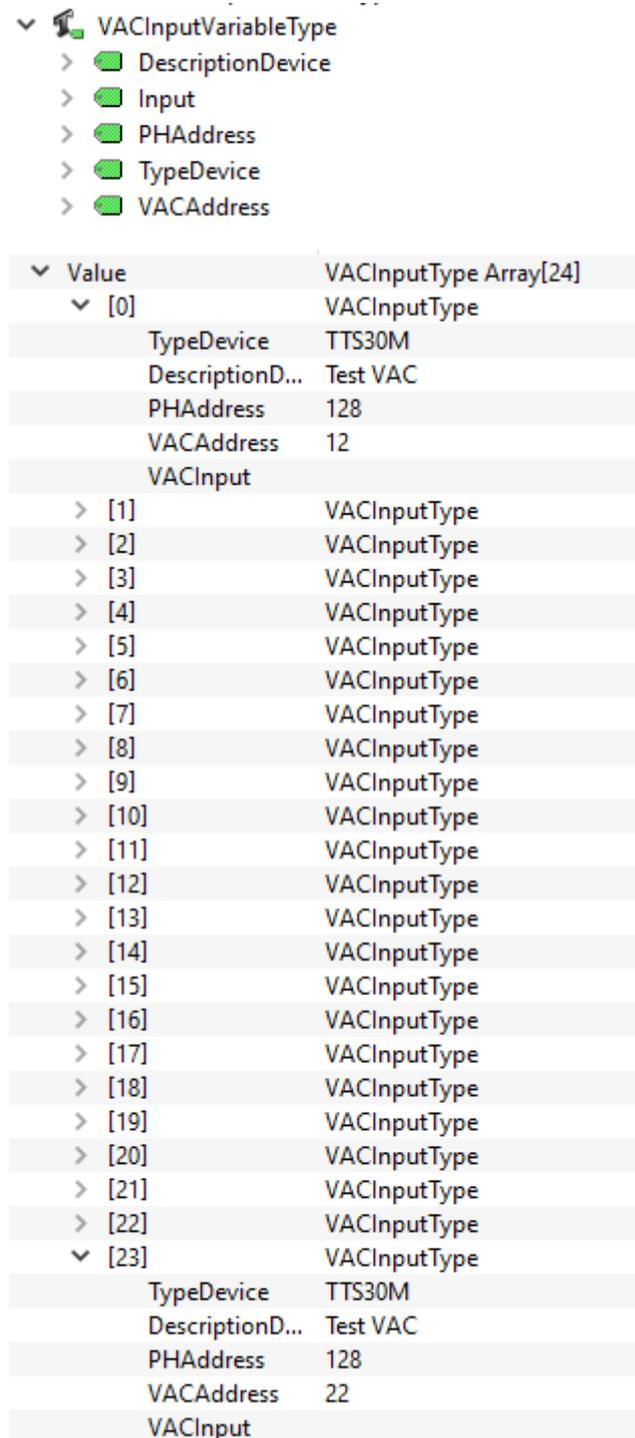
Le variabili semplici per ogni IF304 sono ad esempio:

- ns=3;s=Watch.IF304_1_1_DeviceLock;
- ns=3;s=Watch.IF304_1_1_DevicePoll

VAC

L'IF304, in emulazione di un TTS30M, è dotato dei canali di comunicazione VAC (UDP/IP) e nel Server OPC UA abbiamo le variabili:

- ns=3;s=VAC.VACInput (input)



The screenshot shows the Variable Browser for the VACInputVariableType. It displays a tree structure with the following nodes:

- VACInputVariableType
 - DescriptionDevice
 - Input
 - PHAddress
 - TypeDevice
 - VACAddress
- Value: VACInputType Array[24]
 - [0]: VACInputType
 - TypeDevice: TTS30M
 - DescriptionD...: Test VAC
 - PHAddress: 128
 - VACAddress: 12
 - VACInput
 - [1] to [22]: VACInputType
 - [23]: VACInputType
 - TypeDevice: TTS30M
 - DescriptionD...: Test VAC
 - PHAddress: 128
 - VACAddress: 22
 - VACInput

La variabile semplice per ogni IF304 è ad esempio:

- ns=3;s=VAC.TTS30M_128_1_VACInput.

- ns=3;s=VAC.VACOutput (output)

The screenshot shows a configuration tree for 'VACOutputVariableType'. It includes sub-items: ActOutput, DescriptionDeviceOut, Output, PHAddressOut, TypeDeviceOut, and VACAddressOut. Below this is a 'Value' section showing an array of 24 'VACOutputType' objects. The first object (index 0) and the last object (index 23) are expanded to show their properties:

Index	TypeDevice	DescriptionD...	PHAddress	VACAddress	VACOutput	ActOutput
[0]	TTS30M	Test VAC	128	1	String Array[0]	false
[1]						
[2]						
[3]						
[4]						
[5]						
[6]						
[7]						
[8]						
[9]						
[10]						
[11]						
[12]						
[13]						
[14]						
[15]						
[16]						
[17]						
[18]						
[19]						
[20]						
[21]						
[22]						
[23]	TTS30M	Test VAC	128	19	String Array[0]	false

Per attivare bisogna:

- settare il campo **ActOutput**;
- valorizzare il campo **VACOutput**.

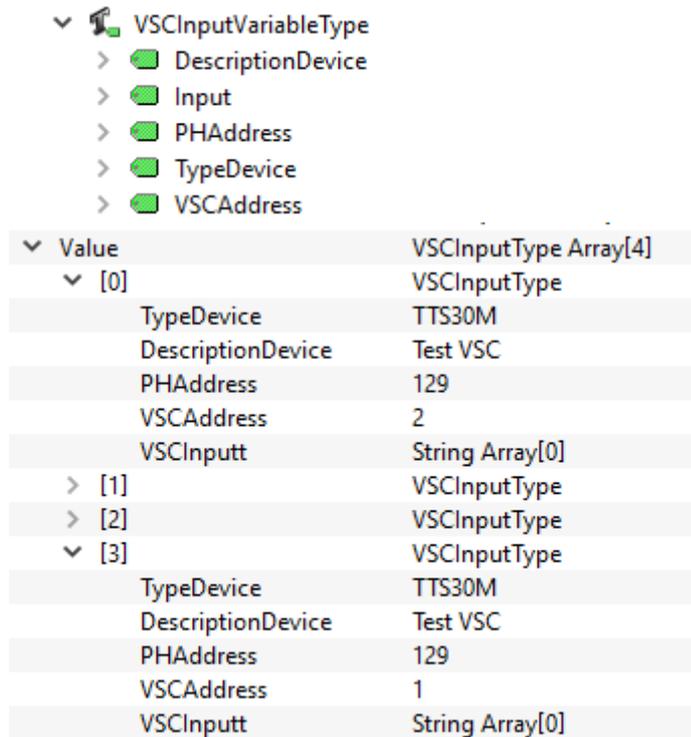
La variabile semplice per ogni IF304 è ad esempio:

- ns=3;s=VAC.TTS30M_128_1_VACOutput.

VSC

L'IF304, in emulazione di un TTS30M, è dotato dei canali di comunicazione VSC (TCP/IP) e nel Server OPC UA abbiamo le variabili:

- ns=3;s=VSC.VSCInput (input)



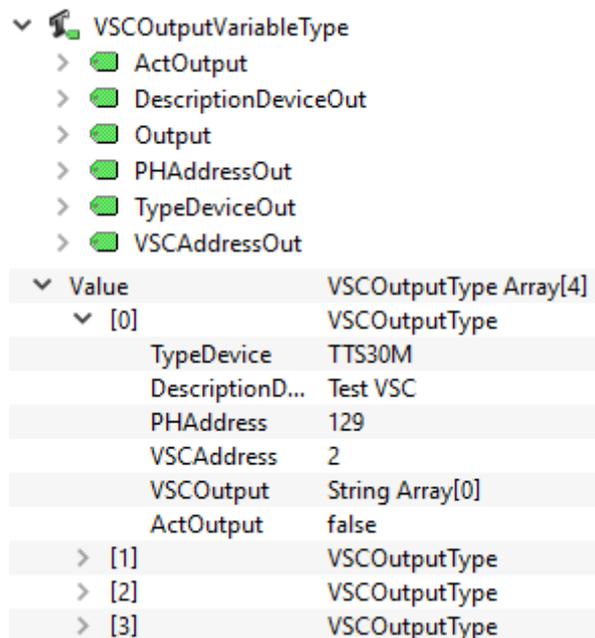
The screenshot shows a tree view of the VSCInputVariableType variable. It is expanded to show its sub-properties: DescriptionDevice, Input, PHAddress, TypeDevice, and VSCAddress. Below this, the 'Value' property is expanded to show an array of 4 VSCInputType objects. The first two objects are expanded to show their sub-properties: TypeDevice (TTS30M), DescriptionDevice (Test VSC), PHAddress (129), and VSCAddress (2). The first object also shows a VSCInputt property with a String Array[0]. The third and fourth objects are not expanded.

Value	VSCInputType Array[4]
[0]	VSCInputType
TypeDevice	TTS30M
DescriptionDevice	Test VSC
PHAddress	129
VSCAddress	2
VSCInputt	String Array[0]
[1]	VSCInputType
[2]	VSCInputType
[3]	VSCInputType
TypeDevice	TTS30M
DescriptionDevice	Test VSC
PHAddress	129
VSCAddress	1
VSCInputt	String Array[0]

La variabile semplice per ogni IF304 è ad esempio:

■ ns=3;s=VSC.TTS30M_129_1_VSCInput.

- ns=3;s=VSC.VSCOutput (output)



The screenshot shows a tree view of the VSCOutputVariableType variable. It is expanded to show its sub-properties: ActOutput, DescriptionDeviceOut, Output, PHAddressOut, TypeDeviceOut, and VSCAddressOut. Below this, the 'Value' property is expanded to show an array of 4 VSCOutputType objects. The first object is expanded to show its sub-properties: TypeDevice (TTS30M), DescriptionD... (Test VSC), PHAddress (129), VSCAddress (2), VSCOutput (String Array[0]), and ActOutput (false). The other three objects are not expanded.

Value	VSCOutputType Array[4]
[0]	VSCOutputType
TypeDevice	TTS30M
DescriptionD...	Test VSC
PHAddress	129
VSCAddress	2
VSCOutput	String Array[0]
ActOutput	false
[1]	VSCOutputType
[2]	VSCOutputType
[3]	VSCOutputType

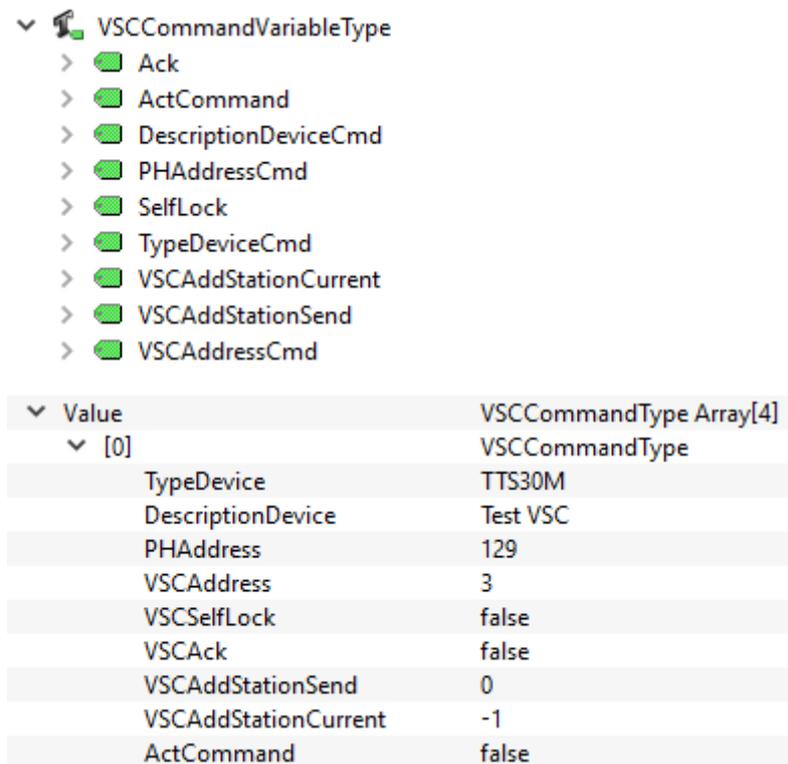
Per attivare bisogna:

- settare il campo **ActOutput**;
- valorizzare il campo **VSCOutput**.

La variabile semplice per ogni IF304 è ad esempio:

- ns=3;s=VSC.TTS30M_129_1_VSCOutput, nel formato **Variable=Value**.

- ns=3;s=VSC.VSCCommand (comandi)



The screenshot shows a tree view of a variable named **VSCCommandVariableType**. It is expanded to show a list of command types: Ack, ActCommand, DescriptionDeviceCmd, PHAddressCmd, SelfLock, TypeDeviceCmd, VSCAddStationCurrent, VSCAddStationSend, and VSCAddressCmd. Below the tree view is a table representing the **Value** array, which is of type **VSCCommandType Array[4]**. The array contains one element at index **[0]**, which is of type **VSCCommandType**. The table lists the following values:

Property	Value
TypeDevice	TTS30M
DescriptionDevice	Test VSC
PHAddress	129
VSCAddress	3
VSCSelfLock	false
VSCAck	false
VSCAddStationSend	0
VSCAddStationCurrent	-1
ActCommand	false

Per attivare bisogna:

- settare il campo **ActCommand**;
- settare_resettare i campi **VSCSelfLock, VSCAck**.
- valorizzare, per i PLC Modbus, **VSCAddStationSend**.
-

Si ottiene, per i PLC Modbus, **VSCAddStationCurrent**.

Le variabili semplici per ogni IF304 sono ad esempio:

- ns=3;s=VSC.TTS30M_129_1_VSCSelfLock;
- ns=3;s=VSC.TTS30M_129_1_VSCAck;
- ns=3;s=VSC.TTS30M_129_1_VSCAddStationSend;
- ns=3;s=VSC.TTS30M_129_1_VSCAddStationCurrent.