PERFECTION IN AUTOMATION
A MEMBER OF THE ABB GROUP

B&R OPC UA System
Create and import OPC UA information models

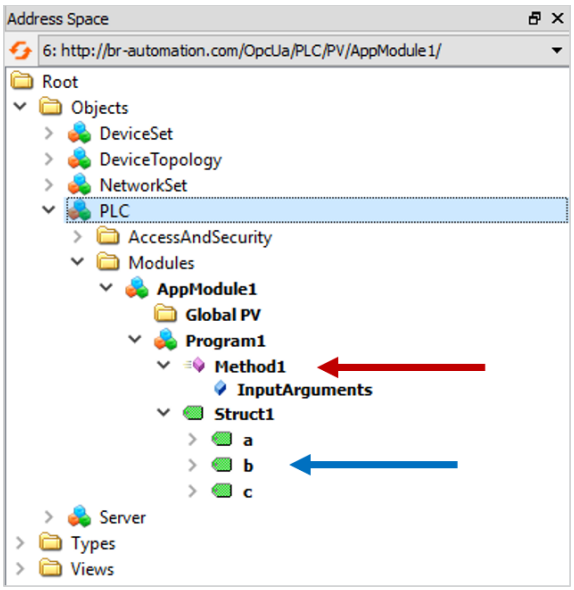
Date: February 15, 2019

I Versions

Version	Date	Comment	Edited by
1.0	Jan 4, 2019	First Edition	Karl Mayr
1.1	Feb 8, 2019	Multiple changes	Karl Mayr
1.2	Feb 15, 2019	Added chapter 5 Event	Karl Mayr

II Table of Contents

1 Situation



The OPC UA information model in the Automation Runtime OPC UA Server is structured according to the structure in the PLC by default.

Starting from the application module, the task, the variable and the element of a variable follows.

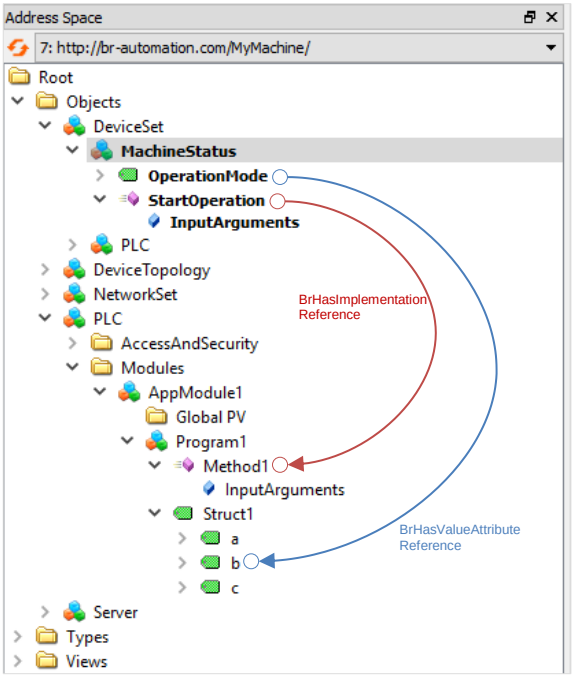
For instance:

PLC variable:

Root/Objects/PLC/Modules/AppModule1/Program1/Struct1.b

PLC method:

Root/Objects/PLC/Modules/AppModule1/Program1/Method1()



But the OPC UA users want a different information model that corresponds to the machine.

For instance:

Variable

Root/Objects/DeviceSet/MachineStatus/OperationMode

is supposed to define a reference to

Program1/Struct1.b

The value of OperationMode should be transferred to Struct1.b and vice versa by the B&R OPC UA server.

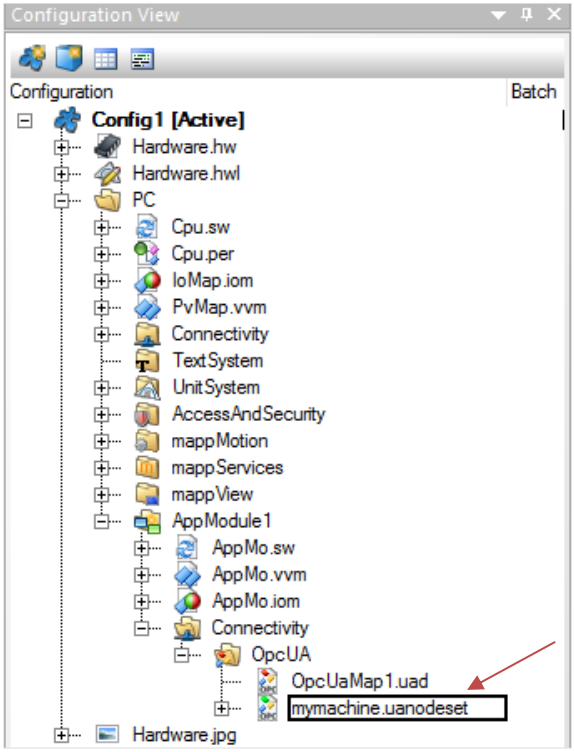
Method :

Root/Objects/DeviceSet/MachineStatus/StartOperationMode()

is supposed to define a reference to

Program1/Method1()

If a client calls StartOperation() the Method1() should be called



This can be achieved by importing NodeSet2 compliant XML files created with an OPC UA modeler or similar tools.

These nodesets are then simply added in the AR configuration and contains all necessary objects (also instances) and references between the models.

And there are information models on the market defined by OPC UA companion specifications like EUROMAP or PackML. These NodeSet2 files also can be imported.

E.g. Euromap defines OPC UA interfaces for plastics and rubber machinery for the data exchange between injection moulding machines and MES (Manufacturing Execution System).

2 Demo Application

Before you try to create an own information model and make references to the B&R information model the recommended way is to write and test the application before, because then you have a target for the references from your own model to the B&R model.

2.1 Write and test your application

Program1::Types.typ [Datatype Declaration]

```
TYPE
    Struct1_Type :  STRUCT
        a : REAL;
        b : INT;
        c : STRING[80];
    END_STRUCT;

    MethodType : STRUCT
        Step : INT;
        LastErrorID : DWORD;
        UaSrv_MethodOperate : UaSrv_MethodOperate;
    END_STRUCT;

    Method1Type : STRUCT
        Method : MethodType;
        Input1 : INT;
    END_STRUCT;

END_TYPE
```

Declare data types for the structured variable

Declare data types for the method

Program1::Variables.var [Variable Declaration]

```
VAR
    Struct1 : Struct1_Type;

    Method1Code : Method1Type;

END_VAR
```

Declare variables for the structure variable and the method code.

Program1::Methods.uam [OPC UA Method Declaration]

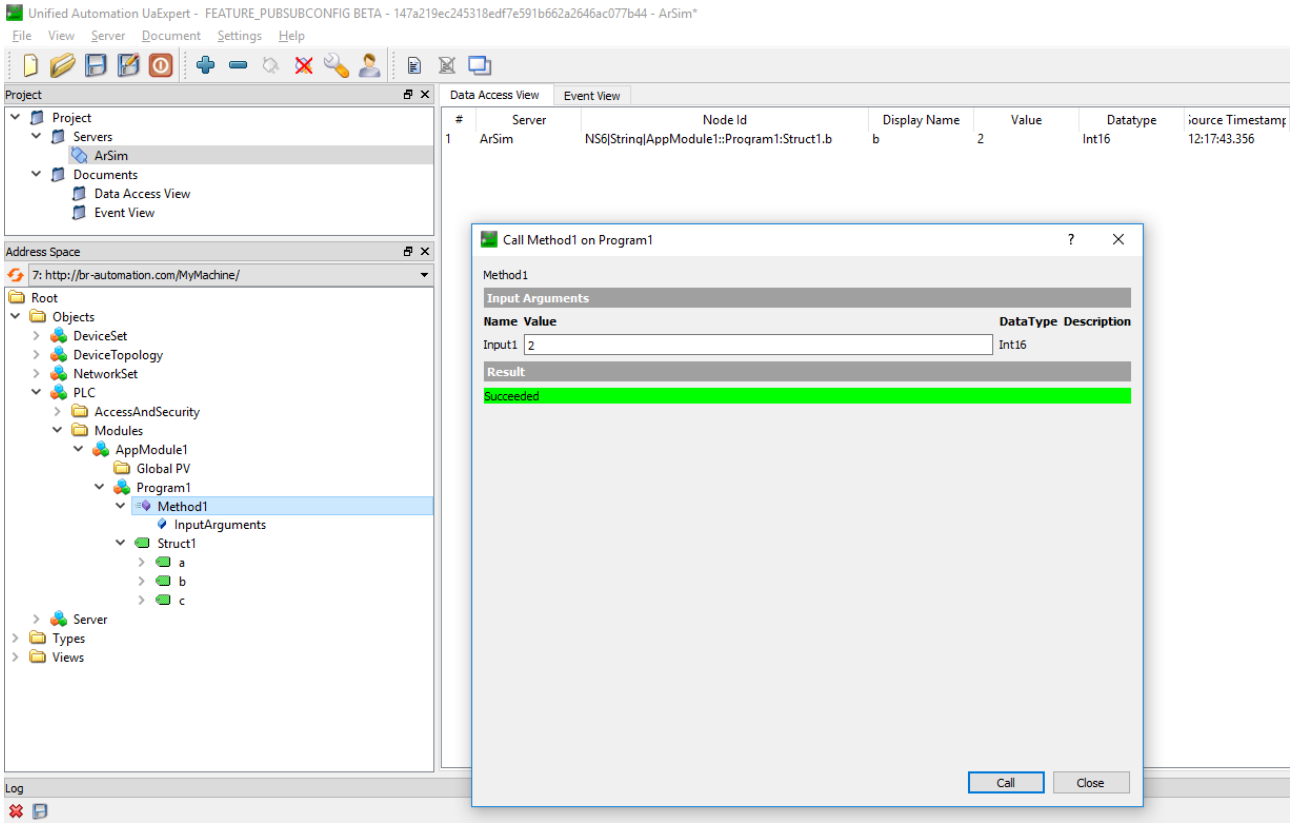
```
OPCUA_METHOD Method1
    ARG_INPUT
        Input1 : Int16 := Method1Code.Input1;
    END_ARG
END_OPCUA_METHOD
```

Declare OPC UA method interface

```
PROGRAM _INIT

  Struct1.b := 0;

END_PROGRAM

PROGRAM _CYCLIC

  // execute Method1()

  CASE Method1Code.Method.Step OF
    0: // start operate
      Method1Code.Method.UaSrv_MethodOperate(Execute := FALSE);
      Method1Code.Method.Step := 1;
    1: // check if method called
      Method1Code.Method.UaSrv_MethodOperate(Execute := TRUE,
        Action  := UaMoa_CheckIsCalled,
        MethodName := 'Method1');
      IF NOT Method1Code.Method.UaSrv_MethodOperate.Busy THEN
        IF Method1Code.Method.UaSrv_MethodOperate.Done THEN
          IF Method1Code.Method.UaSrv_MethodOperate.IsCalled THEN
            Method1Code.Method.Step := 2; // method is called
          ELSE
            Method1Code.Method.Step := 0; // again
          END_IF
        END_IF
        IF Method1Code.Method.UaSrv_MethodOperate.Error THEN
          Method1Code.Method.LastErrorID := Method1Code.Method.UaSrv_MethodOperate.ErrorID;
          Method1Code.Method.Step := 0;
        END_IF
      END_IF
    2: // execute method code
      Struct1.b := Method1Code.Input1;
      Method1Code.Method.Step := 3;
    3: // finish operate
      Method1Code.Method.UaSrv_MethodOperate(Execute := FALSE);
      Method1Code.Method.Step := 4;
    4: // finish method
      Method1Code.Method.UaSrv_MethodOperate(Execute := TRUE,
        Action  := UaMoa_Finished);
      IF NOT Method1Code.Method.UaSrv_MethodOperate.Busy THEN
        IF Method1Code.Method.UaSrv_MethodOperate.Done THEN
          Method1Code.Method.LastErrorID:= 0;
          Method1Code.Method.Step := 0; // method is finished, wait again for call
        END_IF
        IF Method1Code.Method.UaSrv_MethodOperate.Error THEN
          Method1Code.Method.LastErrorID:= Method1Code.Method.UaSrv_MethodOperate.ErrorID;
          Method1Code.Method.Step := 0;
        END_IF
      END_IF
  END_CASE;


END_PROGRAM
```

Write program code

Test your application. Call the <Method1> and check if variable <Struct1.b> is set to the <Input1> value.



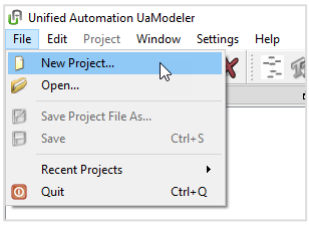
3 Create new information model

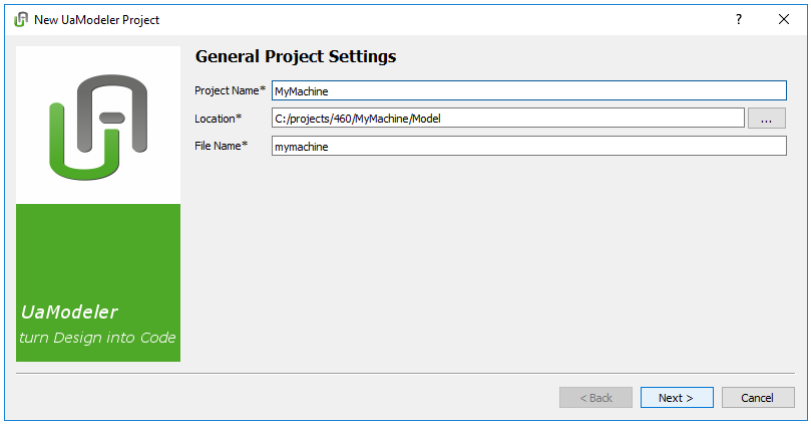
We use for instance the UA Modeler (https://www.unified-automation.com/) to create an own OPC UA information model.

At first we create a new UA Modeler project.

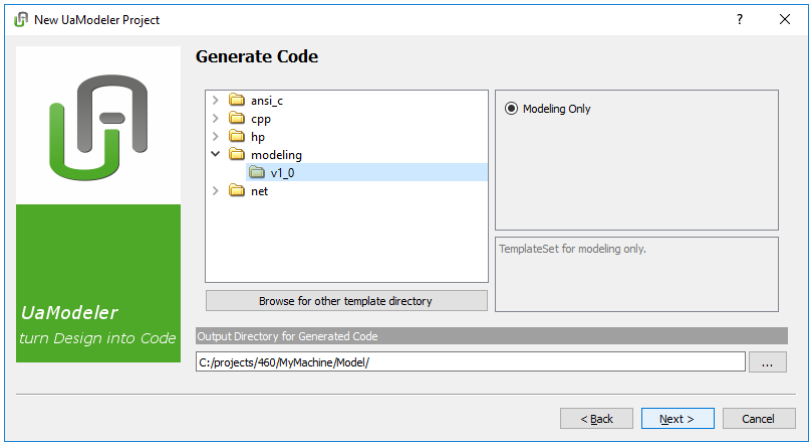
3.1 Create new project

Create new project and enter required data:





We use option ‘Modeling Only’ and enter the Output Directory :

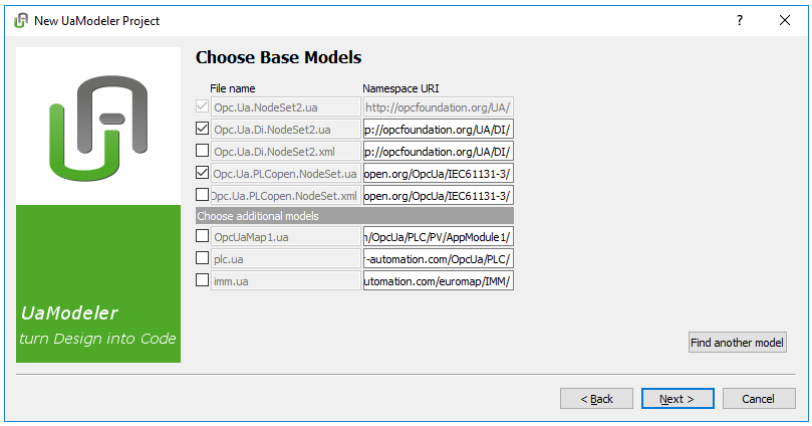


Select the base models

Opc.Ua.NodeSet2.ua – it contains namespace 0 objects,

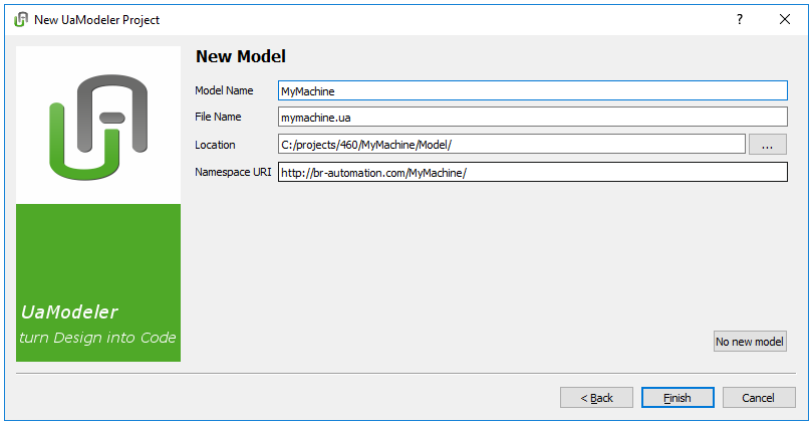
Opc.Ua.Di.NodeSet2.ua – it contains Device Integration objects,

Opc.Ua.PLCopen.NodeSet.ua – it contains IEC 61131 objects

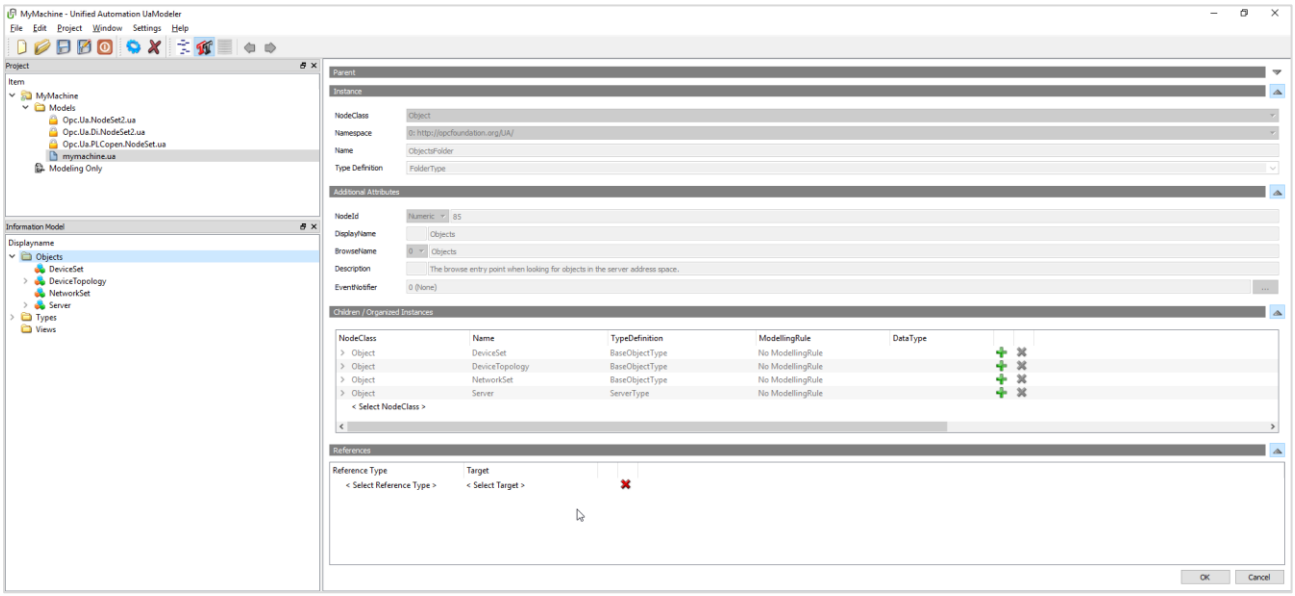


Press Next and enter required data:

Note : the Namespace URI should contain your company name



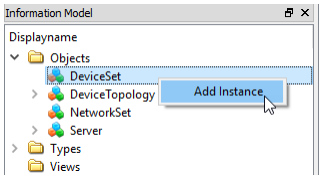
A new namespace <mymachine> is created



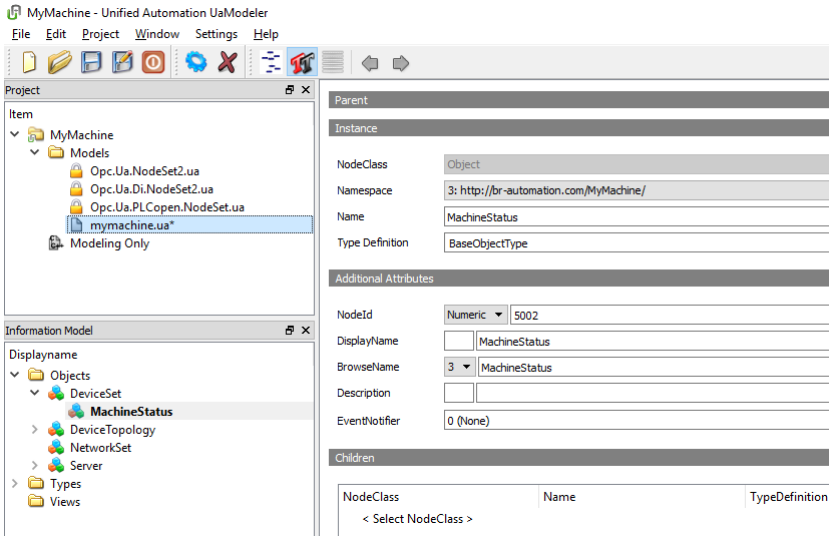
3.2 Add new instances

We can now add new instances to the namespace <mymachine>.

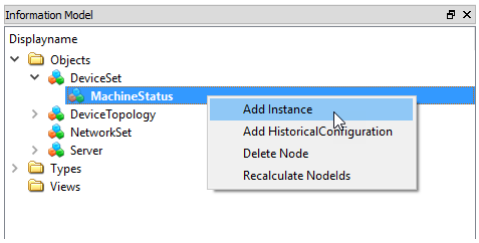
Select the <DeviceSet> Object and click right mouse button to <Add Instance>



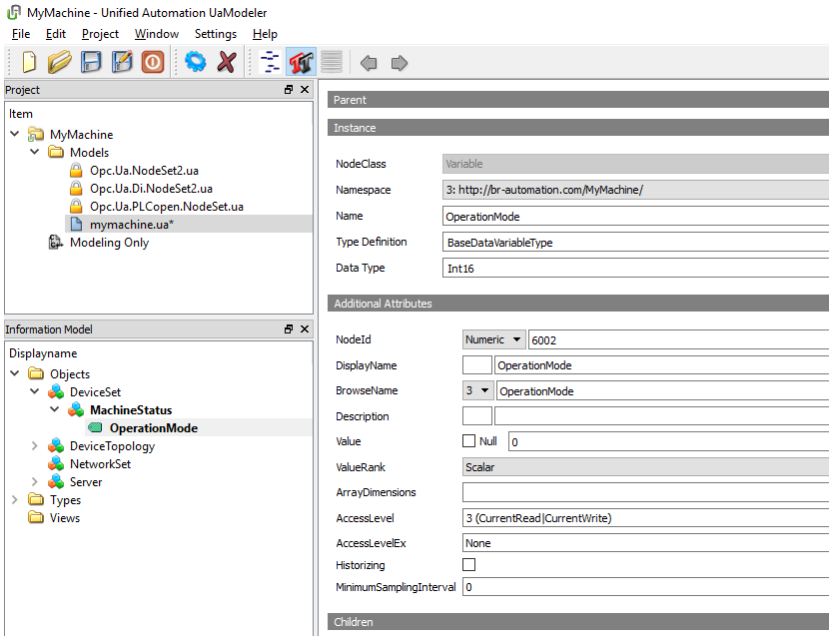
Enter the NodeClass and the name of the instance, for example <MachineStatus> and click OK button



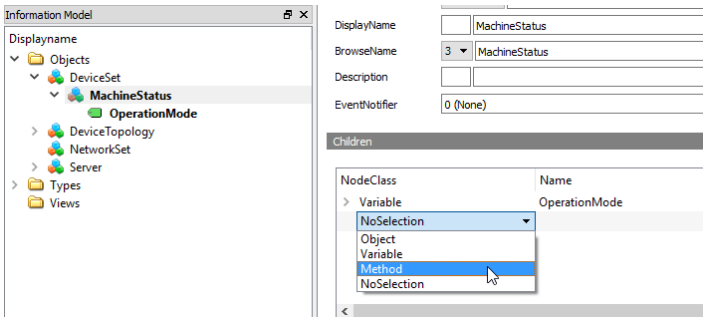
Add variable <OperationMode> for object <MachineStatus>



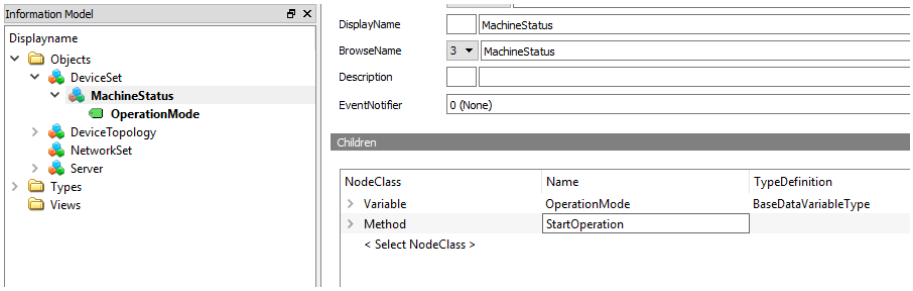
Enter the NodeClass <Variable>, the Name <OperationMode> and the DataType <Int16> and click OK button.



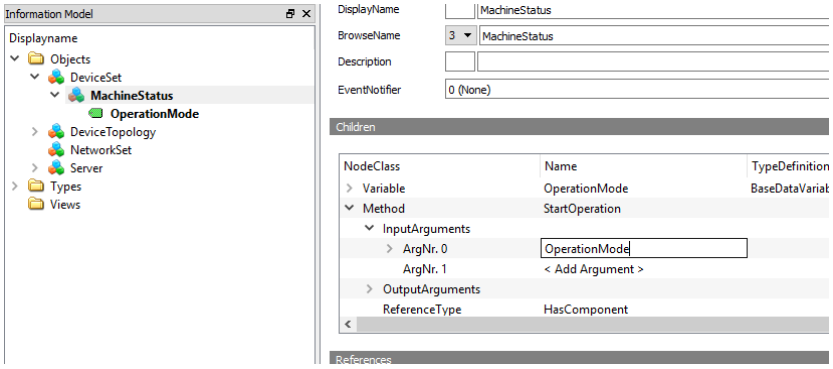
Select <MachineStatus> and Select <Method> in the children section of the right pane.



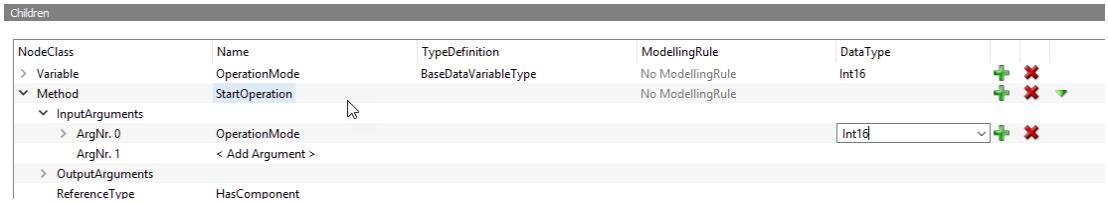
Enter the name of the method <StartOperation>



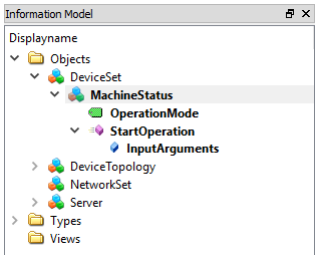
Enter the InputArgument <OperationMode> for the method



Enter the DataType <Int16> for the InputArgument and press OK button.



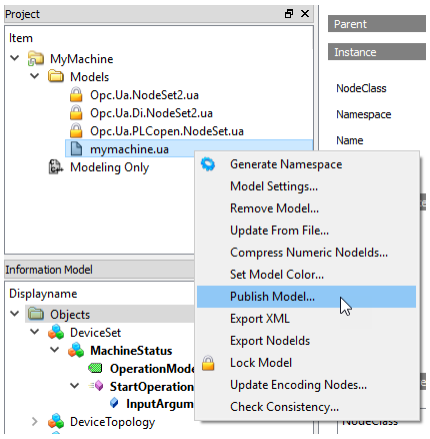
The new information model is completed now



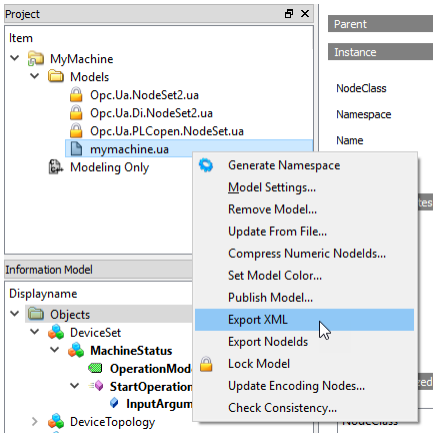
If you want you can test now the new model, if it can be read by the OPC UA server on Automation Runtime

3.3 Export the new information model

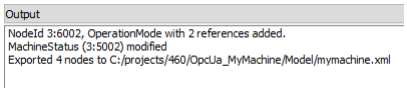
Save all the changes and publish the model with right mouse click on <mymachine.ua> and select <Publish Model>



Save all changes and export the NodeSet2 file with right mouse click on <mymachine.ua> and select <Export XML>.

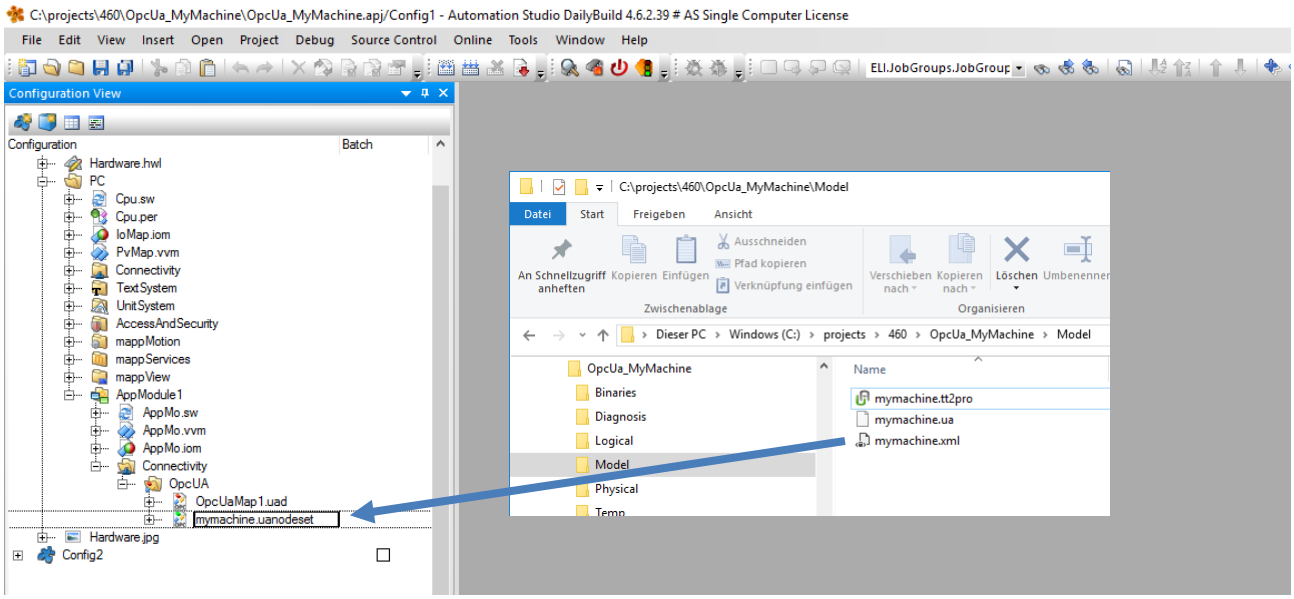


The output windows shows how many nodes are exported and where the nodeset file is stored.

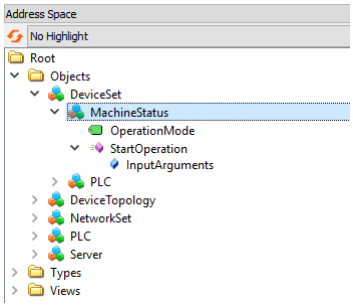


3.4 Import the new model in Automation Studio

Drag and drop this new .xml file to the Configuration view of the AS project <OpcUa_MyMachine> and rename the file type <.xml> to <.uanodeset>.



After Build and transfer the new model running on the B&R OPC UA server can be explored e.g. using the UA Expert.

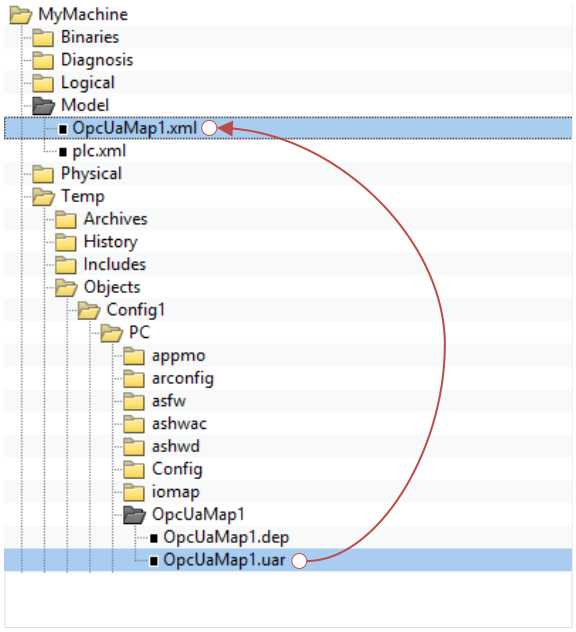


4 Set references

To supply the variables and methods of the new model with the PLC variables and PLC methods of the application references from the new model to the PLC model has to be set.

The references can be set if the target model (the PLC model) is known by the modeler tool. This can be done by importing the PLC model.

4.1 Copy the B&R model files

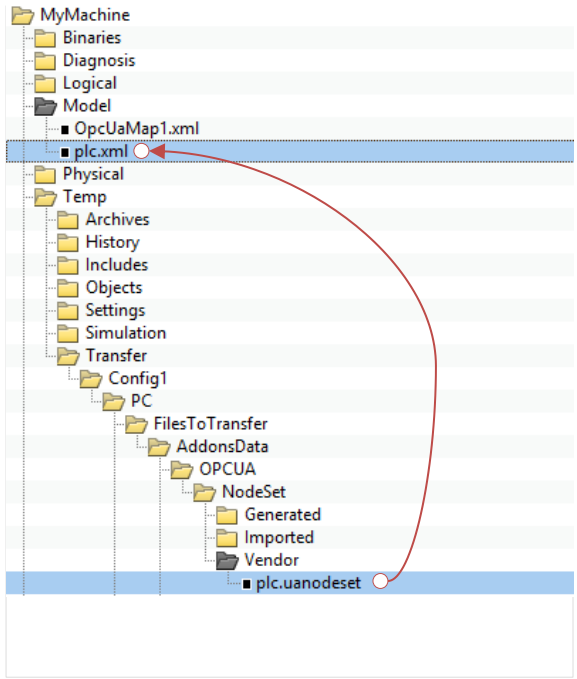


After each Build the Automation Studio generates NodeSet2 compliant XML files containing the B&R OPC UA models.

These files are stored in the projects \Temp\Objects directory of the current configuration. The filename is identical to the Default View filename and has extension .uar

For instance :

copy “\MyMachine\Temp\Objects\Config1\PC\OpcUaMap1\OpcUaMap1.uar” to “\MyMachine\Model” and rename it to .xml for later use in the modeler tool.



Also the plc.uanoteset file is generated. It contains the PLC Object model.

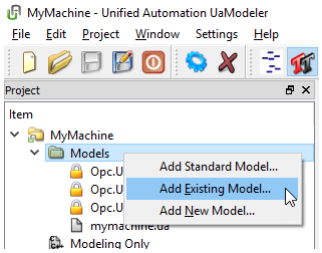
This file is stored in the projects \Temp\Transfer directory of the current configuration. The filename is plc.uanodeset.

For instance :

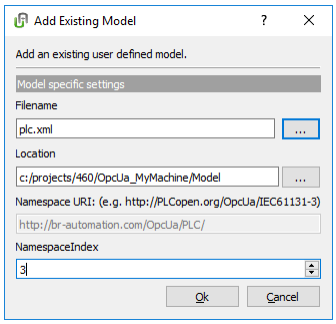
copy “\MyMachine\Temp\Transfer\Config1\PC\FilesToTransfer\AddonsData\OPCUA\NodeSet\Vendor\plc.uanodeset” to “\MyMachine\Model” and rename it to .xml for later use in the modeler tool.

4.2 Import B&R PLC models

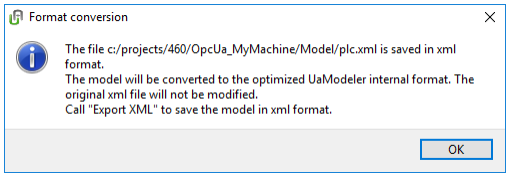
Open the UA Modeler project again and with a right mouse click select <Add Existing Model>



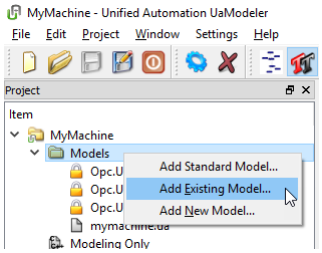
Select the nodeset <plc.xml> copied and renamed before and set <NamespaceIndex> to 3.



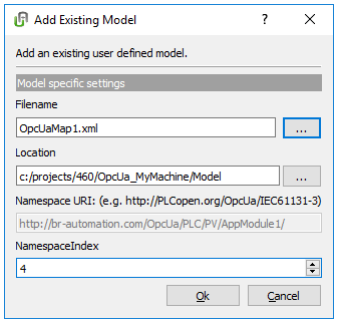
A format conversion is made



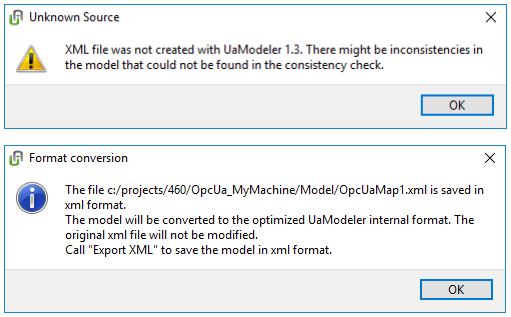
With a right mouse click select <Add Existing Model> again



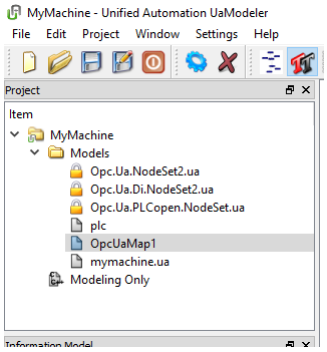
Select the nodeset <OpcUaMap1.xml> copied and renamed before and set <NamespaceIndex> to 4.



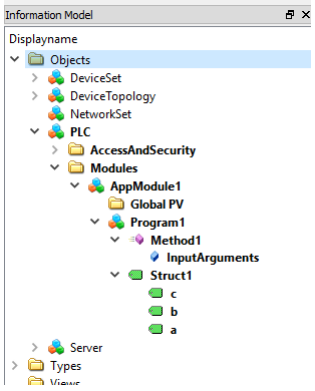
A format conversion is made



Both nodeset files are imported now



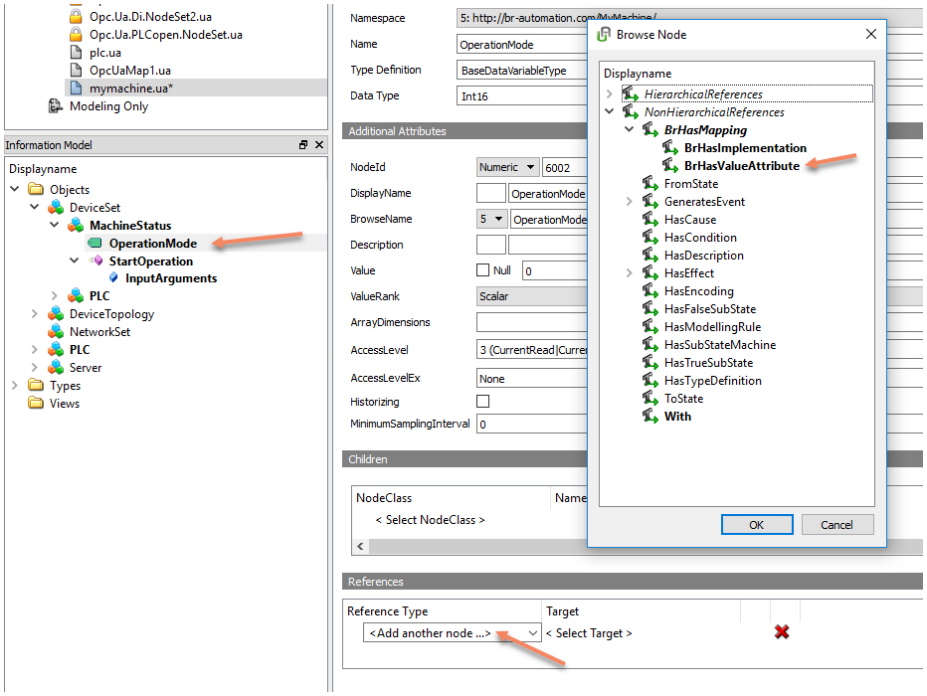
And the PLC objects are available now to be used



4.3 Set References from new model to B&R model

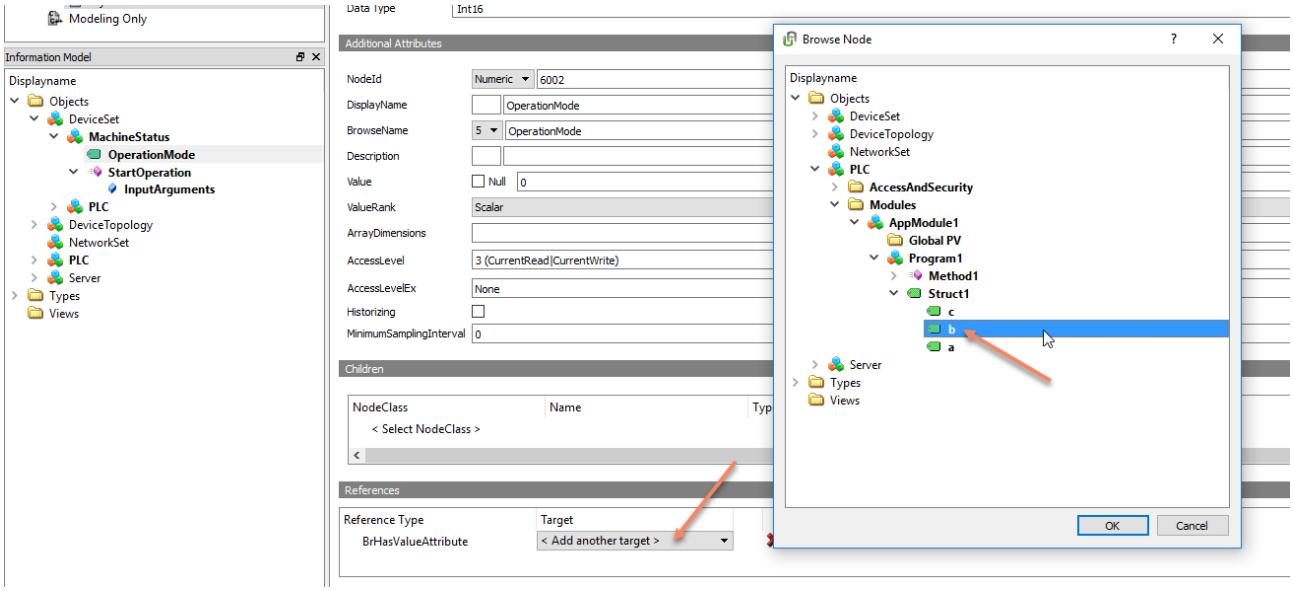
Select variable <OperationMode> and click a Reference Type <Add another node> in the References section in the right pane

Select the <BrHasValueAttribute> reference type

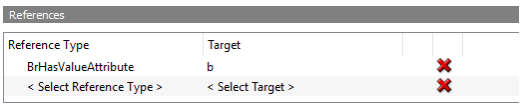


Click the <Target> in the References section <Add another target> in the right pane

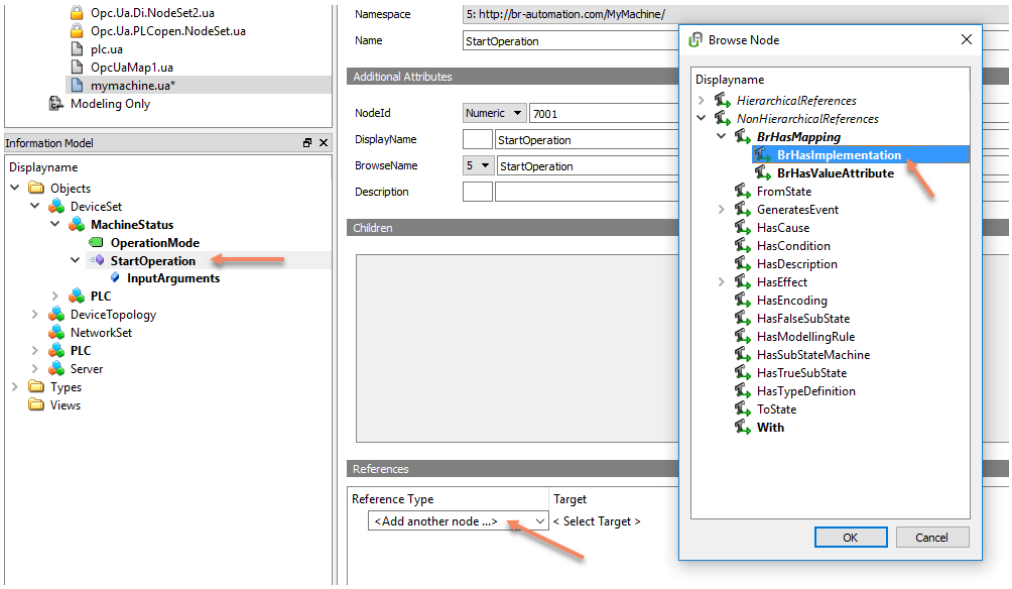
Select the PLC variable Struct1.b and click OK button.

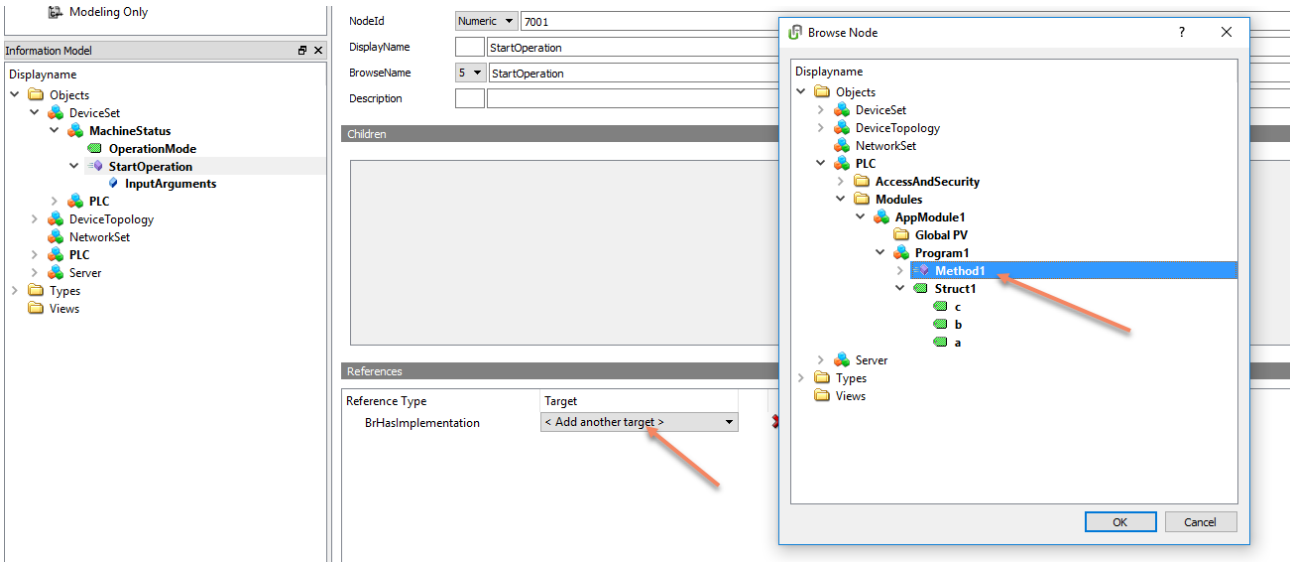


The reference for the variable is now set from the new information model to the B&R PLC information model.



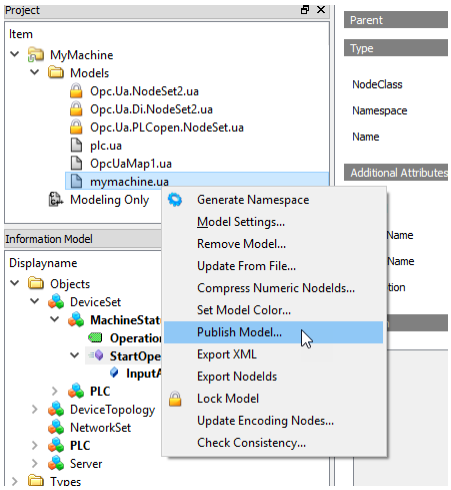
Repeat this with the method <StartOperation> but with reference type <BrHasImplementation>



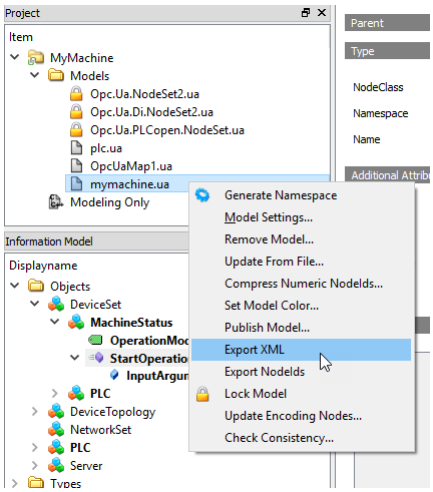


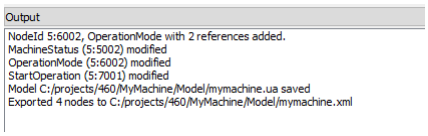
4.4 Export new information model

Save all the changes and publish the model with right mouse click on mymachine.ua and select Pulish Model …



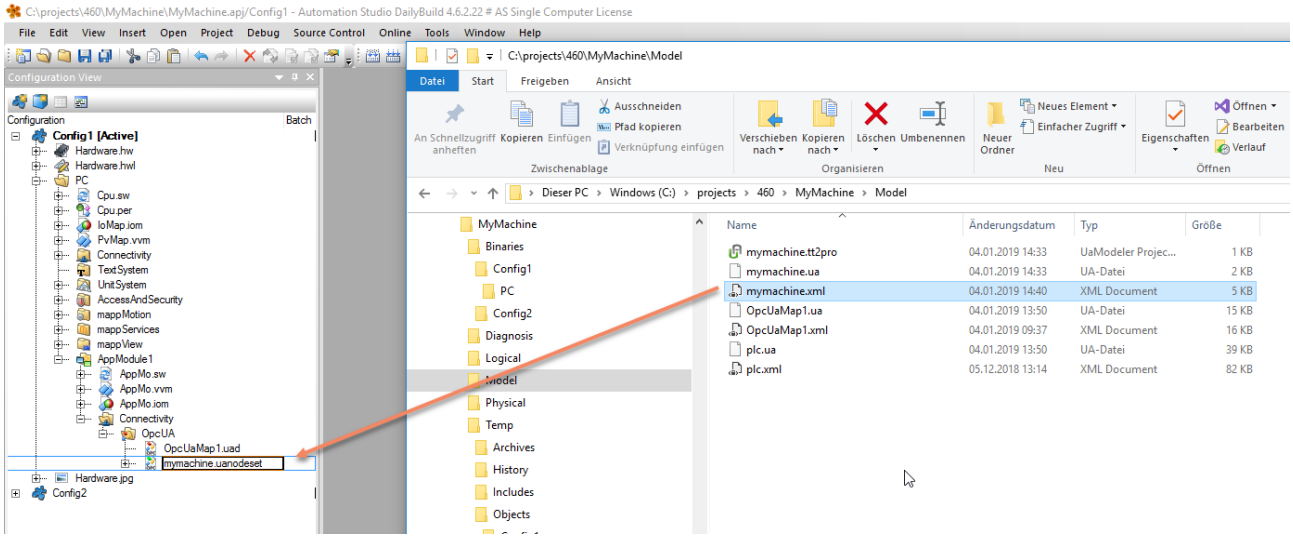
Save all changes and export the NodeSet2 file with right mouse click on mymachine.ua and select Export XML.





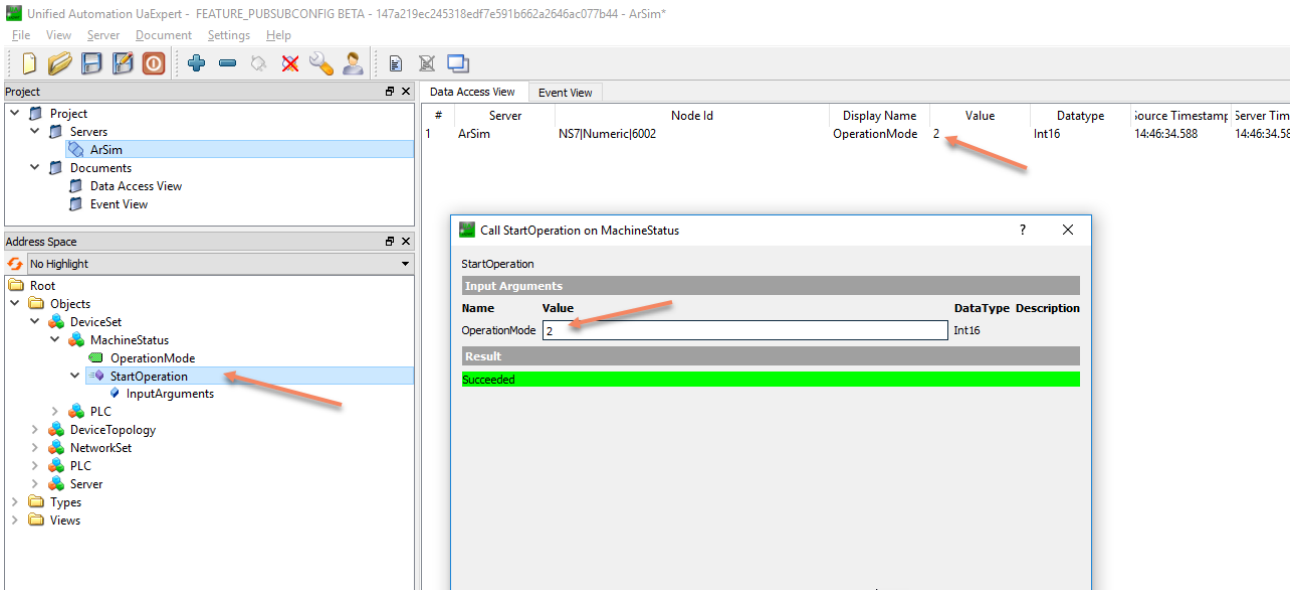
4.5 Import the new model into AS project

Copy this NodeSet2 file to the corresponding Connectivity folder in the Configuration View and rename it to “mymachine.uanodeset”.



After compile and download the own information model can be tested

Call the method <StartOperation> and enter 2 for <OperationMode>. After successful call the value of <OperationMode> should be 2.

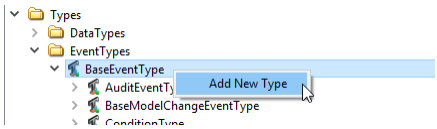


5 Events

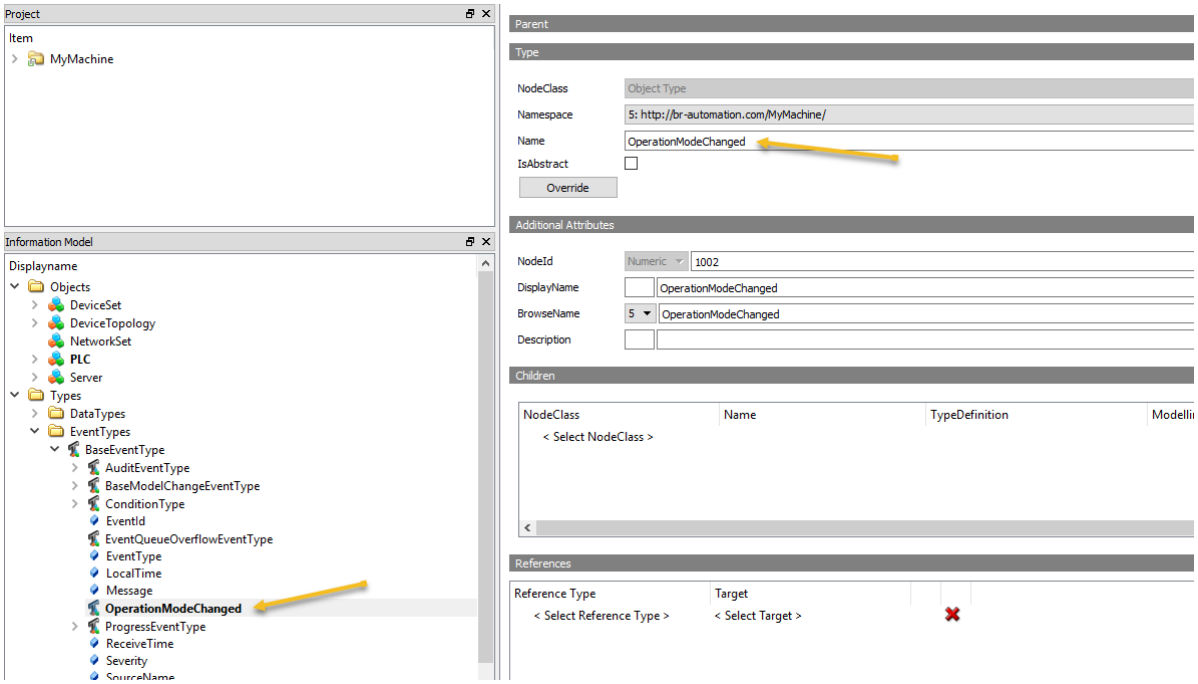
If you want to fire own defined events, you can model these events deriving the BaseEventType.

5.1 Add own event

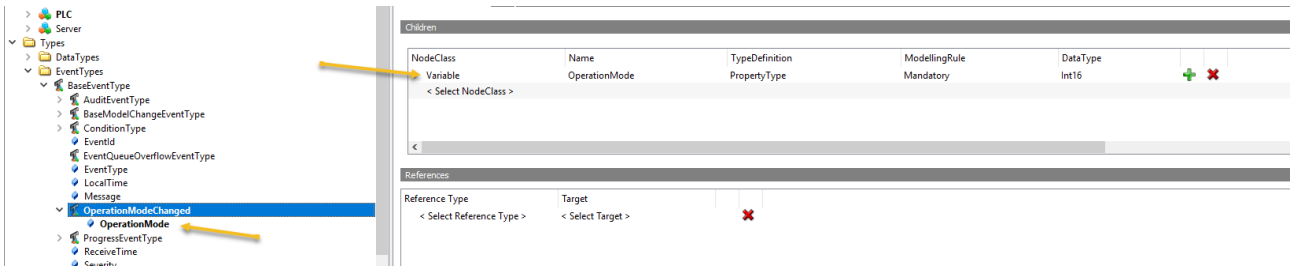
With a right mouse click to <BaseEventType> select <Add New Type> :



Enter the NodeClass and the name of the type, for example <OperationModeChanged> and click OK button

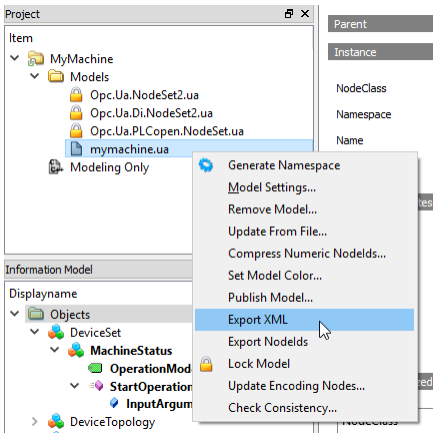


Select the new event type and add an event field in the Children section on the right pane and press OK

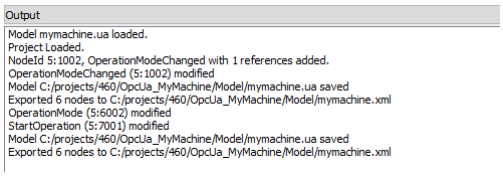


5.2 Reimport the new model in AS

Save all changes and export the NodeSet2 file with right mouse click on <mymachine.ua> and select <Export XML>.

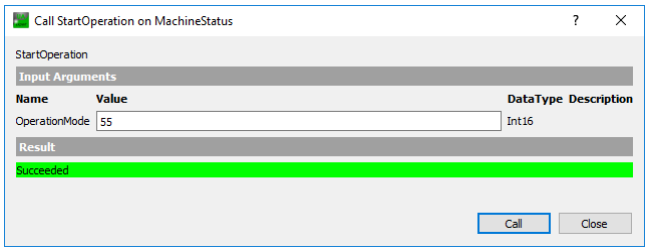


The output windows shows how many nodes are exported and where the nodeset file is stored.

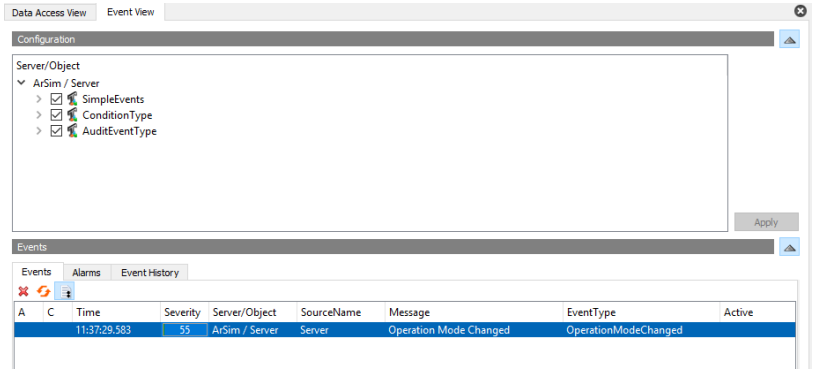


5.3 Test application

Call the method <StartOperation>. The Value of OperationMode is set and the event will be fired.



In the EventView of UAExpert the fired event is shown.



The details of the events shows the new value in the field “Operation mode”