

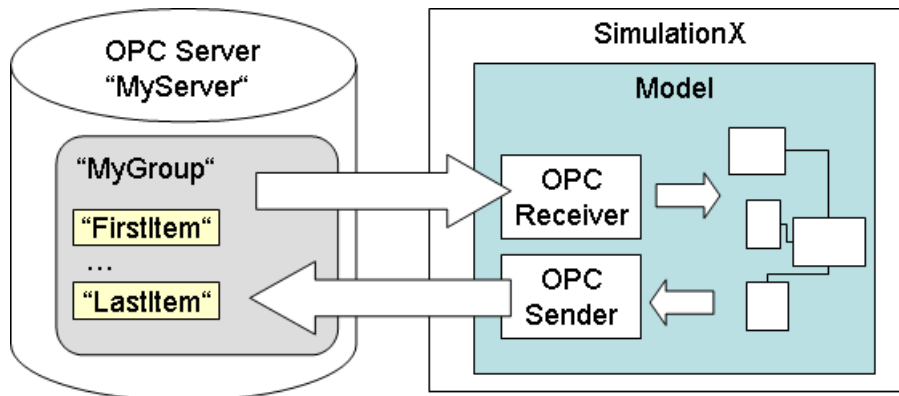
Tutorials

Tutorial 16 – OPC Interface in SimulationX

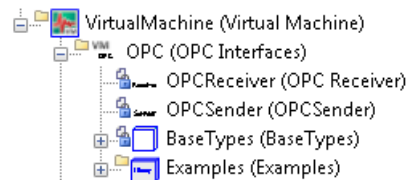
Objective

SimulationX provides an interface which enables you to exchange data between a SimulationX model and OPC data items of an OPC Server.

OPC data items have a data type and they are organized in groups. Server, groups and items are identified by names.



The OPC package in SimulationX contains types for receiving OPC data (OPCReceiver), sending OPC data (OPCSender) and example models.



Defining connections to OPC items

Receiver and sender can handle multiple items of one OPC group. They need the server name and the group name to establish the connection to the OPC group at the server.

The following parameters of sender and receiver define the connection to the items of the group:

```
nData          ... Number of items to be handled
opc_names[nData] ... Vector with item names
opc_types[nData] ... Vector of item types
```

The length of `opc_names` and `opc_types` must be equal to `nData`. Names and types of the items have to match the settings of the items at the server. The following example accesses two items of "MyGroup" at "MyServer":

```
opc_server="MyServer"
opc_group="MyGroup"
nData = 2
opc_names = {"FirstItem", "SecondItem"}
opc_types = {OPCType.bool, OPCType.integer}
```

- *Virtual start-up*
- *Physical plant models*
- *SiL or HiL techniques for PLC controlled equipment*
- *PLC code from the same source during the complete design process*
- *Minimizing the risk in development*
- *Data exchange with industrial automation systems*
- *Standardized open connectivity interface (OPC)*
- *Dynamic number of exchanged data*
- *Code export available*

Data exchange

The vector *data* of OPCReceiver contains the values received from the OPC server items. The length of *data* is equal to the number of items (nData). Other model elements can use these values by referencing to elements of the *data* vector.

The vector *data* of OPCSender contains the references to values in the SimulationX model which should be sent to OPC server items. The length of *data* must be equal to the number of items (nData).

Simulation time and real time

OPC servers are working in real time, but SimulationX executes the simulation calculation as quick as possible. For simple models the real calculation time can be less than the simulated time interval. You can decrease the maximum step size (dtMax) in order to increase the real calculation time.

OPC Interface and Codeexport executable model

If you want to use the codeexport for models with OPCSenders or OPCReceivers, then you have to copy OPCClient.dll to the project path and add the OPC client library to the code export project.

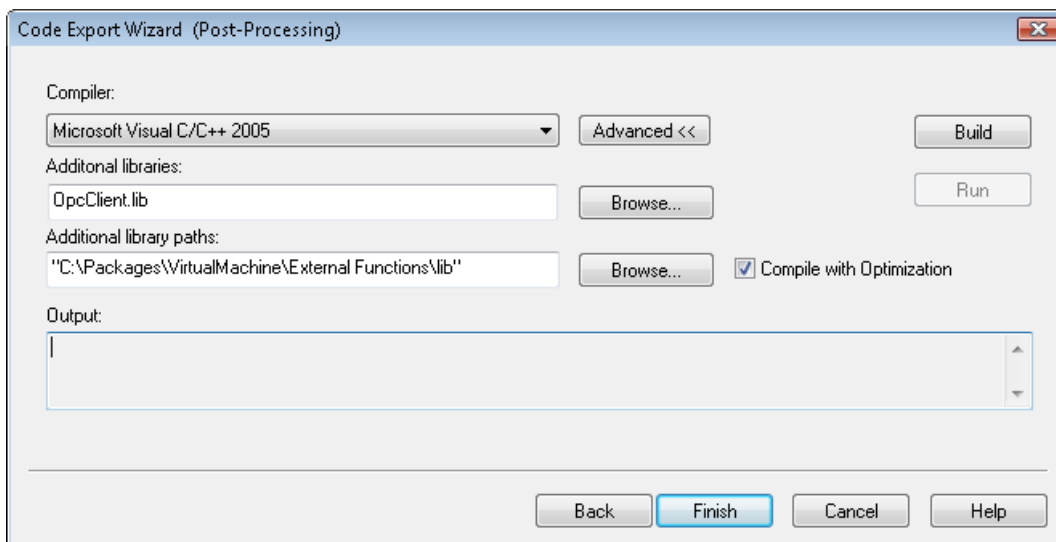


Figure 1: Post-Processing of Code Export

Tutorial

This tutorial describes how a SimulationX model can be connected to an OPC Server. The MatrikonOPC Simulation Server is used as OPC server. You can get it from <http://www.matrikonopc.com> for free.

1.1. Setting up SimulationX

To install the OPC package you have to extract the archive iti_opc.zip to one folder of the Modelica search path. You can check the Modelica search path in the options of SimulationX.

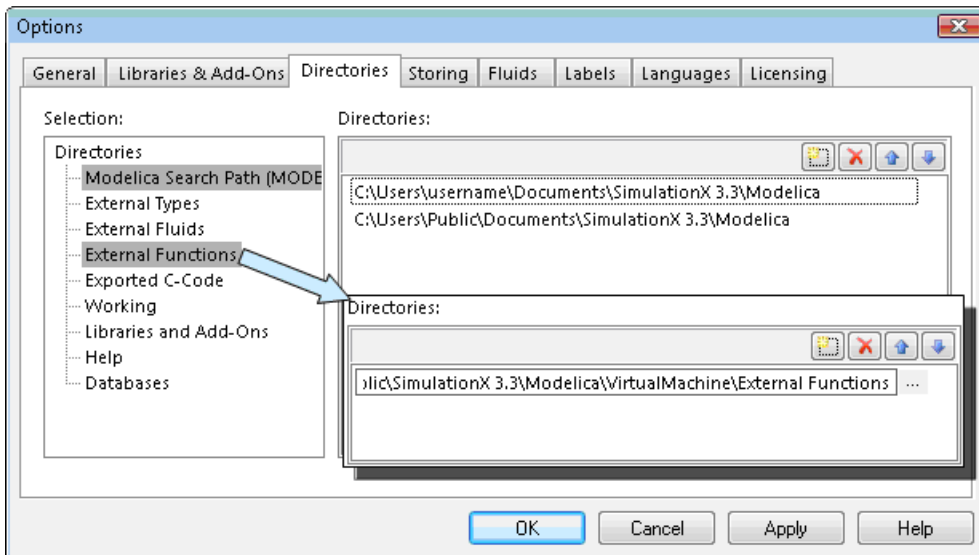


Figure 2: Setup Modelica search path and external function search path

The OPC interface uses external functions in OpcClient.dll. You will find OpcClient.dll in the subfolder "External Functions" of the extracted archive. Add this subfolder to the list of external functions directories in SimulationX.

1.2. Setting up the Matrikon OPCServer

After installing the Matrikon OPCServer you have to start the MatrikonOPC Explorer.

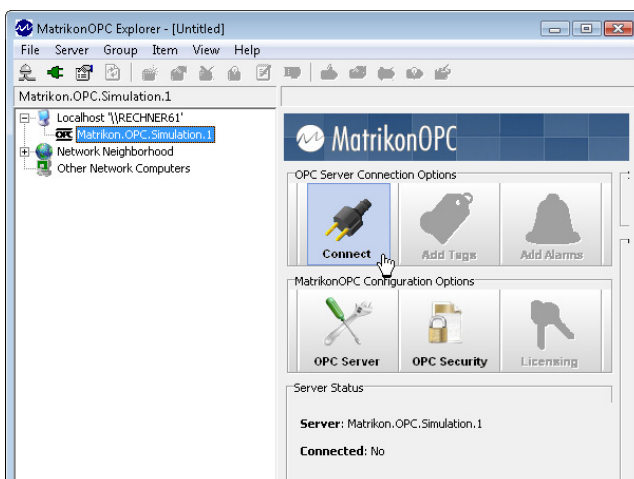


Figure 3: MatrikonOPC Explorer

The explorer lists all installed OPC server at your PC. Select the server with the name Matrikon.OPC.Simulation.1 and connect to it. Now you could add your own OPC groups and items to the server. In the OPC examples directory you will find a configuration file (OPCTutorial.xml) which adds the needed items for the tutorial.

After opening the XML file the OPC server should have the two groups "ServerInputs" and "ServerOutputs". Each of this group contains one item for each supported data type (Boolean, Int2, Int4, Real4, Real8).

Don't close the MatrikonOPC Explorer in order to keep the OPC server running.

1.3. Create SimulationX model

At first create a new model and add an OPCSender and add an OPCReceiver. Please rename the OPCSender to *opc_s* and the OPCReceiver to *opc_r*.

1.1.1 Define connection to OPC group

Set the following parameters to define the connection to the OPC groups of the selected OPC server.

```
opc_r:      opc_server: "Matrikon.OPC.Simulation.1"
opc_group: "ServerOutput"
```

```
opc_s:      opc_server: "Matrikon.OPC.Simulation.1"
opc_group: "ServerInput"
```

Now *opc_r* can receive data from all items in the *opc_group* "ServerOutput" and *opc_s* can send to all items in "ServerInput".

OPCSender and OPCReceiver can reference to the same group, but they should not use the same items.

1.1.2 Define connection to OPC items

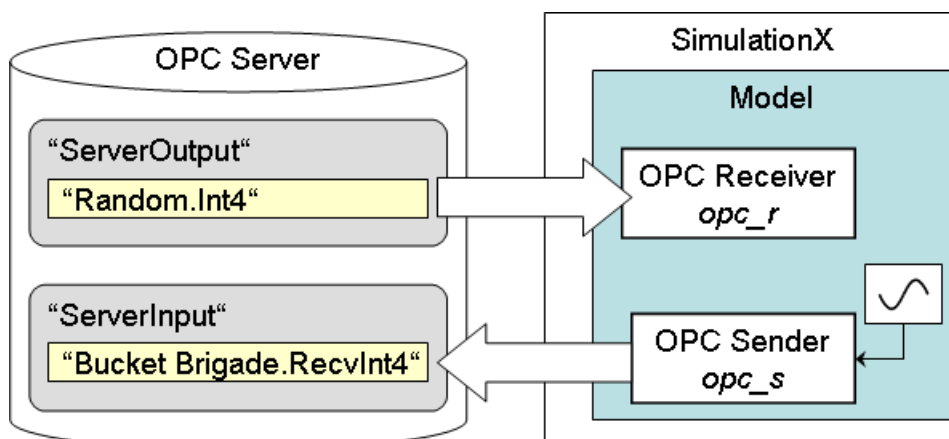


Figure 4: Data exchange between OPC Server and SimulationX

In this tutorial we want to receive a (random) value generated from the OPC item "Random.Int4" and send a sinus curve to the OPC item "Bucket Brigade.RecvInt4".

The following picture shows the parameters of *opc_r*:

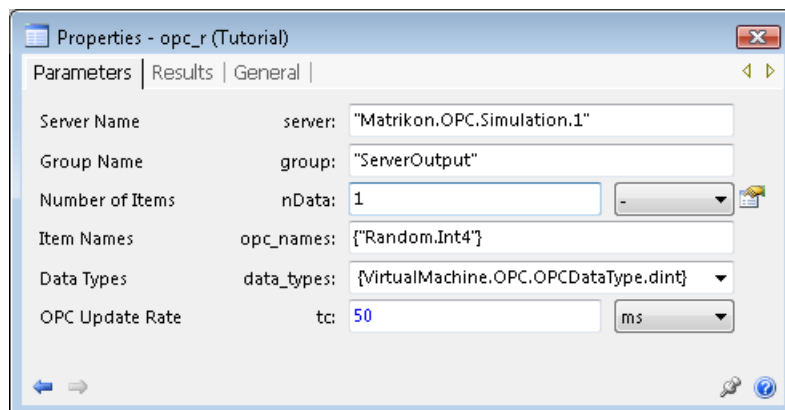


Figure 5: Parameters of OPC Receiver *opc_r*

Parameter *nData* defines the number of items to be received from the OPC server. The vectors *opc_names* and *data_types* must match to item names and item types at the OPC server. Please type the vector of *data_types* by hand. The combo box does not define the needed vector of values.

In order to see the received data you should enable the protocol attribute at the result *data* and add a result window.

The following picture shows the parameters of *opc_s*:

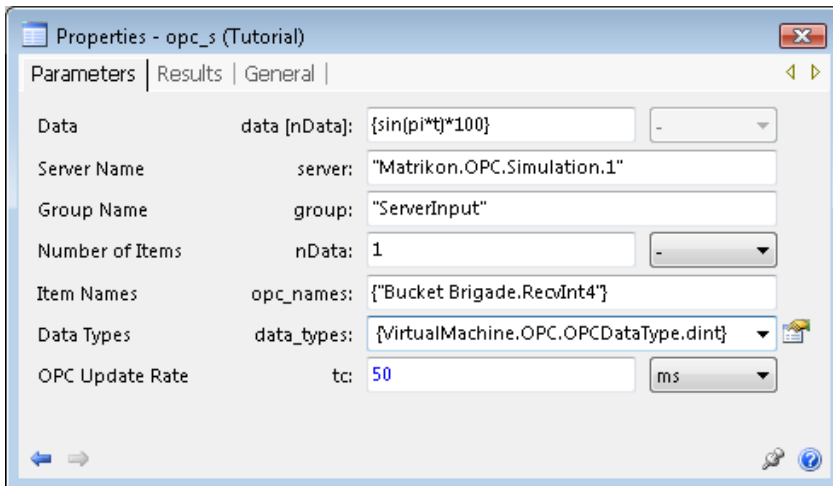


Figure 6: Parameters of OPC Sender *opc_s*

OPC senders have the same parameters like OPC receivers to identify the OPC items, but the parameter *data* contains the data to be sent.

1.1.3 Adjust simulation settings

After connecting to the OPC items we can start the simulation. If all settings are right, then the simulation should run without any error message. The result window should show (random) curve similar to the curve in the picture.

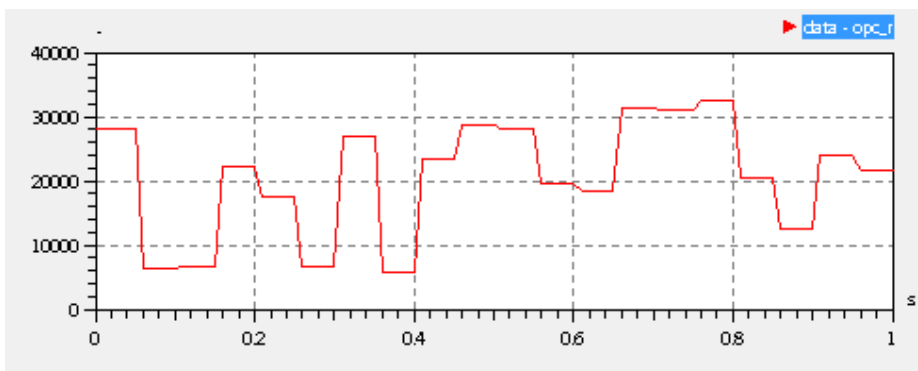


Figure 7: Received data from OPC item "Random.Int4"

But the calculation time is too short to see the data sent by SimulationX in the Matrikon OPC explorer. In order to see the changes in the OPC explorer we have to slow down the calculation in SimulationX by defining a smaller maximum step size (for example $1e-005$ s).

Arrange the windows of the MatrikonOPC Explorer and the SimulationX side by side, select the OPC group "ServerInput" in the MatrikonOPC Explorer and start the simulation in SimulationX. Now you can check the data changes of the item "Bucket Brigade.RecvInt4" in the MatrikonOPC Explorer. The item receives its data from the OPC sender *opc_s* during the simulation in SimulationX.