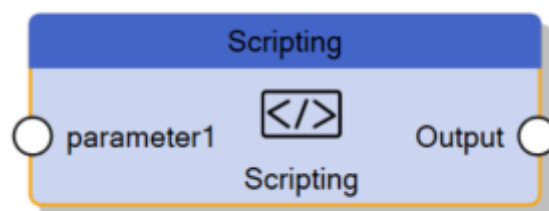


Scripting Node



The **Scripting Node** allows users with basic programming knowledge to freely define custom operational logic using code.

To configure the node, the script must be written using the integrated IDE and then executed by the Logic Module.

The method called by default is:

MainMethod

Additional input parameters can be created by adding parameters to the method signature.

For example:

```
public static object MainMethod(object parameter1, object parameter2,
object parameterN)
```

To complete the script execution successfully, the method must return a value.

MainMethod

The **MainMethod** is the entry point of the script.

Each parameter declared in the method signature becomes available as an input connector of the node.

For example:

```
public static object MainMethod(object a, object b)
```

creates two input parameters:

- **a**
- **b**

The method can then use these values inside the custom logic.

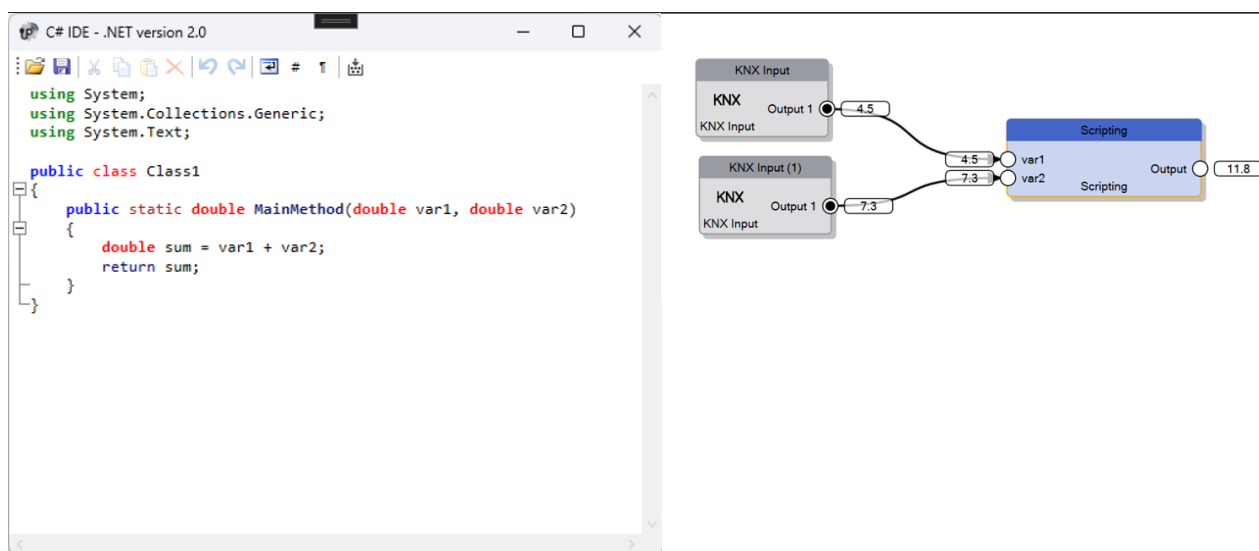
Return value

The script must return a value to complete its execution successfully.

For example:

```
public static object MainMethod(object a, object b)
{
    return Convert.ToDouble(a) + Convert.ToDouble(b);
}
```

The returned result is sent to the default output of the Scripting Node.



Example of an addition script

Input parameters

The parameters declared in the **MainMethod** signature define the input connectors available on the Scripting Node.

For example:

```
public static object MainMethod(object temperature, object setpoint,
object enable)
```

creates three input parameters:

- **temperature**
- **setpoint**
- **enable**

Each parameter receives the value coming from the corresponding connector and can be used inside

the script.

A practical example could be:

```
public static object MainMethod(object temperature, object setpoint,
object enable)
{
    bool active = Convert.ToBoolean(enable);

    if (!active)
        return 0;

    double t = Convert.ToDouble(temperature);
    double sp = Convert.ToDouble(setpoint);

    if (t < sp)
        return 1;

    return 0;
}
```

In this example:

- **temperature** contains the measured room temperature;
- **setpoint** contains the target temperature;
- **enable** determines whether the logic is active;
- the script returns **1** when the logic is enabled and the measured temperature is lower than the setpoint;
- otherwise, it returns **0**.



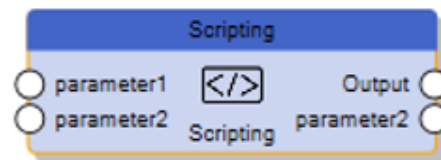
The parameter names used in the **MainMethod** should be clear and descriptive, because they identify the corresponding values used by the script.

Multiple outputs

If more than one output value is required, additional parameters can be defined and used both as inputs and as outputs.

This allows the script to return the main result through the normal function return value and update additional values through the declared parameters.

```
public static object MainMethod(object parameter1, ref object parameter2)
{
    return parameter1;
}
```



Example of return parameter 2 value

Operation

The Scripting Node follows this sequence:

- the script is written in the integrated IDE;
- the **MainMethod** is used as the entry point;
- parameters declared in the method signature become available to the node;
- the script processes the received values;
- the method returns the main output value;
- additional parameters can also be used to provide additional output values.

Example - Addition

A simple addition can be implemented as:

```
public static object MainMethod(object value1, object value2)
{
    double a = Convert.ToDouble(value1);
    double b = Convert.ToDouble(value2);

    return a + b;
}
```

If:

value1 = 10

and:

value2 = 5

the returned result is:

15

Example - Conditional logic

The Scripting Node can also be used to implement custom conditions.

For example:

```
public static object MainMethod(object temperature, object setpoint)
{
    double t = Convert.ToDouble(temperature);
    double sp = Convert.ToDouble(setpoint);

    if (t > sp)
        return 1;

    return 0;
}
```

The script returns:

- **1** when the temperature is greater than the setpoint;
- **0** otherwise.

Typical applications

The Scripting Node is useful when the required logic cannot be conveniently implemented using standard Logic Module nodes.

Typical applications include:

- custom mathematical calculations;
- complex conditional logic;
- manipulation of multiple input values;
- custom conversions;
- advanced data processing;
- creation of reusable algorithms;
- implementation of logic requiring multiple results.

Verification through simulation

The Scripting Node can be tested using the Logic Module simulation tools.

A typical test procedure is the following:

- open the integrated script editor;
- define the required **MainMethod** parameters;
- write the script;
- make sure that the method returns a value;

- verify that the expected input parameters are available on the node;
- assign test values to the inputs;
- run the simulation;
- verify the returned output;
- if additional output parameters are used, verify their values as well.

Configuration notes

- **MainMethod** is the default method executed by the Scripting Node.
- Parameters added to the method signature are used by the node as configurable values.
- The method must return a value.
- Convert input values to the expected data type before using them in calculations.
- Additional parameters can be used when multiple output values are required.
- Test custom scripts carefully before using them in the final project.
- Avoid unnecessary complexity when the same behaviour can be implemented with standard Logic Module nodes.

Related pages

- [Logic Module](#)
- [Logic Module - Simulation management](#)

From:

<https://www.thinknx.com/wiki/> - **Learning Thinknx**

Permanent link:

https://www.thinknx.com/wiki/doku.php?id=lm_scripting

Last update: **2026/09/22 16:09**

