OMRON controller NJ-Series provider

Version 1.2.8

User’s guide

June 12, 2024

[Remarks]

[Revision history]

Version	Date	Content
1.0.0	2015-1-21	First edition
1.1.0	2016-12-9	Added the function of obtaining the variables information later on Added the function of direct writing to Arrays Support extension Ethernet/IP unit
1.1.1	2016-12-26	Added Error code list
1.1.2	2017-1-7	Support the larger element number of arrays
1.1.2	2018-5-21	Added the "OpenMode" option of "AddController" method.
1.1.2	2018-10-29	Added the "Timeout" option of "AddController" method.
1.2.0	2018-11-13	Added the "OpenMode" option of "AddVariable" method. Added the "Model" option of "AddController" method.
1.2.1	2019-3-4	It is corrected that the array of two dimensions or more can be correctly acquired in the acquisition of the batch of the structure. The performance has improved.
1.2.2	2019-7-30	Bug fixed. Add the direct access method to the structure.
1.2.3	2020-7-22	Fixed GetVariableNames bug.
1.2.4	2020-11-9	Fixed the description of AddController method and added data-size. Fixed the description of AddVariable method. Added the another station access method using tag data link Updated error code list Fixed to output a structure size error when the size of the structure exceeds the upper limit.
	2021-4-27	Corrected the description of "DelayInit".option Corrected the wording of "OpenMode" option. Corrected the description of "LBound" option.
1.2.5	2021-6-18	Fixed some error codes
1.2.6	2021-12-16	Added the "BufferClearMode" option of "AddController" method. Collected an error.
1.2.7	2022-3-25	Added consistency check for request and response packets.
1.2.8	2024-6-12	Modify the sequence number to be used during LargeForwardOpen to be generated and used randomly when connecting controllers. Added a process to recreate the sequence number when the sequence number used during LargeForwardOpen is duplicated and an error occurs. The error message issued by the provider was changed.

【Operation confirmation model】

Model	Version	Notes
NJ301-1100		
NX701-1720		
NX1P2		

Contents

1. Introduction

OMRON controller NJ-Series provider is an ORiN2 CAO provider that allows the access to variables provided by NJ-Series controller through CIP communication. By using NJ-Series provider, any users who are unfamiliar with the CIP protocol can easily access NJ-Series variables.

This document describes the overview of the OMRON NJ-Series provider and the CAO interface (function specifications) implemented in the provider.

2. Overview of the provider

2.1. Installation

OMRON NJ-Series provider module comprises the following DLLs. You do not need to perform this step if the provider has been installed with the ORiN2 SDK installer. If you install the provider manually, refer to the Table 2-1.

Table 2-1 NJ series provider

File name	CaoProvOmronNJ.dll
ProgID	CaoProv.OMRON.NJ
Registry registration	regsvr32 CaoProvOmronNJ.dll
Deregistration	regsvr32 /u CaoProvOmronNJ.dll

2.2. Overview

OMRON NJ-Series provider accesses variables through CIP protocol.

2.3. Method properties

2.3.1. CaoWorkspace::AddController method

In CaoWorkspace, add a controller object. For OMRON NJ series providers, refer to the parameters passed when AddController method is executed and connect to the corresponding OMRON controller NJ series. The following are the specifics of AddController method:

FORMAT

AddController

```
(
        "<controller name>",            // Controller name (optional)
        "CaoProv.OMRON.NJ",             // Provider name (fixed)
        "<machine name>",               // Provider running machine name
        "<Option>"                      // Option character string
)
```

Option

The following is an optional specification for Option character string: Option character string is a comma (,) string consisting of the options listed below.

Option	Required	Description	Value Range	Default Value
Conn	○	Specify the connection destination. For more information, see section 2.3.1.1Conn Option. e.g.) "Conn = TCP: {IP address of connection destination}"	--	--
Timeout	--	Specify the amount of time, in milliseconds, before communication times out.	1 - 2147483647	60000
DelayInit	--	This is the setting of the delayed variable information acquisition function. Specify one of the following: 0 : Get all variable information at the time of AddController (connection). 1 : Get the variable information required for Add Variable.	0 - 1	0
Network_Type_Number	--	Specify the network identification number of the CPU module.	1 - 255	1

Option	Required	Description	Value Range	Default Value
Unit_Address	--	Specify the CPU Unit No. address.	0 - 255	0
OpenMode	--	Set the communication method for sending/receiving data. For more information, see section 2.3.1.2, OpenMode Options.2.3.1.2	0 - 2	0
Model	--	Specifies the model of the CPU to be connected. For more information, see section 2.3.1.3, Model Options.2.3.1.3	0 - 1	0
BufferClearMode	--	Specifies whether to clear the buffer before sending the request. Please specify one of the following. False : Do not clear the buffer. True : Clears the buffer.	False,True	False

2.3.1.1. Conn Option

The following is a Conn optional connection parameter string: Here, brackets ("[]") are optional, and the underlined part in the description of each parameter indicates the default value when no options are specified.

TCP

"Conn = TCP: <destination IP>[:<destination port>]"

<Destination IP>: Specify the destination IP address in ***.***.***.*** (Required)

<Destination port>: Specify the port number to connect to. 44818

E.g. 1) IP address 192.168.1.10, port number 44818, timeout 3000ms, variable information delay acquisition function 0, CPU unit network identification number 1, CPU unit number 0

"Conn=TCP:192.168.1.10,Timeout=3000"

E.g. 2) IP address 192.168.1.10, port number 44816, timeout 3000ms, delayed acquisition function of variable information 1, network identification number 2 of CPU unit, when specifying by CPU unit No. 1

"Conn=TCP:192.168.1.10:44816,Timeout=3000,DelayInit=1,Network_Type_Number=2,Unit_Address=1"

2.3.1.2. OpenMode Option

This option specifies the communication method for sending and receiving data. When the size of the data is smaller than 240 bytes, better to select UnConnected Send mode. When the size is larger than 240 bytes, better to Large Forward Open.[1]

The communication methods that can be connected differ depending on the connected unit. See Table 2-2 for details.

Set value	Mode	Advanced	Upper limit of struct size (byte)	Sequence partitioning size (byte)
0	Unconnected Send	Select this when the size of the data to be handled is 240 bytes or less. If an array of 240 bytes or more is requested, communication is divided in units of 240 bytes.	240	240

[1] UnConnected Send: When the size of the data part + header part exceeds 502 bytes of the CIP convention, communication fails.
Large Forward Open mode: When the size of the data part + header part exceeds 1994 bytes of the CIP convention, communication fails.

Set value	Mode	Advanced	Upper limit of struct size (byte)	Sequence partitioning size (byte)
1	Large Forward Open (Direct connection)	Select this option when Ethernet cable is connected directly to NJ/NX and the data size to be handled is larger than 240 bytes. If an array of 1200 bytes or more is requested, communication is divided in units of 1200 bytes.	1200	1200
2	Large Forward Open (Redirect)	Select this when connecting to NJ or NX via a hub and the size of the data to be handled is larger than 240 bytes. If an array of 1200 bytes or more is requested, communication is divided in units of 1200 bytes.	1200	1200

Table 2-2 Connection target units that can be connected

Set value	Mode	Internal EtherNet Unit	Expansion EtherNet Unit
0	Unconnected Send	○	○
1	Large Forward Open (Direct connection)	○	×
2	Large Forward Open	○	○

E.g. 1) IP address 192.168.1.10, port number 44818, when the size of data to be handled is 240 bytes or less

"Conn=TCP:192.168.1.10,Timeout=3000"

E.g. 2) IP address 192.168.1.10, port number 44818, the size of the data to be handled is larger than 240 bytes, when connecting to NJ or NX via a hub

"Conn=TCP:192.168.1.10,OpenMode=2"

2.3.1.3. Model Option

This option specifies the model of the CPU to be connected. This option is meaningful only when 2.3.1.2 OpenMode option specifies Large Forward Open.[2]

[2] NX701 CPU: If the size of the data part + header part exceeds 8192 bytes of the CIP convention, communication fails.

Set value	Model	Advanced	Upper limit of struct size (byte)	Sequence partitioning size (byte)
0	All models	Can be selected for all models.	1200	1200
1	NX701 CPU	Select this when using NX701. The size of data that can be transferred at one time increases, making communication more efficient.	7396	7396

e.g.) IP-address 192.168.1.10, port number 44818, CPU-model connected to NX701 via hubs

"Conn=TCP:192.168.1.10,OpenMode=2,Model=1"

2.3.2. CaoController::GetVariableNames method

Gets the list of variables defined in the connected NJ series. The variable name obtained by this method can be used as the first argument of AddVariable method described later. GetVariableNames is specifi ed as follows.

FORMAT

```
GetVariableNames
(
        "<Option>"              // Option character string (not used)
)
```

2.3.3. CaoController::AddVariable method

Adds a CaoVariable for accessing variables. AddVariable is specifi ed as follows.

FORMAT

```
AddVariable
(
        "<variable name>",      // Variable name
        "<Option>"              // Option character string
)
```

Variable name

The variable name can be specified by variable names got by 2.3.2 CaoController::GetVariableNames method. When specify the access path in the Path option, you can specify any variable name.

Option

The following is an optional specification for Option character string: Option character string is a comma (,) string consisting of the options listed below.

Option	Required	Description	Value Range	Default Value
Path	--	Specify the access path (see 6 Method of specifying access passing). When the Path option is specified, any character string can be specified for the variable name. When a non-existent access path is specified, or a union is specified directly, an error results. If omitted, the name specified by <variable name> is assumed to be specified.	--	--
LBound	--	Specifies the first subscript of the array. If-1 is specified for Elem, this option is ignored.	0 - 65535	0
Elem	--	Specifies the array length. For more information, specify 2.3.3.1 Elem Option	-1 - 2147483647	-1
OpenMode	--	Set the communication method for sending/receiving data for each Variable. The details are the same as for the 2.3.1.2 OpenMode Option. If OpenMode is specified in Variable, operation will be performed in the form of overwriting OpenMode option specified when adding controller object.	0 - 2	The OpenMode value at AddController.

2.3.3.1. Elem Option

This option specifies the length of the array to be accessed. If 1 is specified, it is treated as a single value instead of an array. If -1 is specified, NJ is accessed and LBound and Elem options are automatically determined. You can also combine LBound and Elem options to access parts of the array.

E.g.) To access Arr[100] to Arr[107] of the array Arr[0..255], specify as follows.

LBound=100,Elem=8

3. Command reference

3.1. Controller class

Table 3-1 CaoController::Execute command list

3.1.1. **CaoController::Execute("GetVariableType") command**

Obtain a variable type that is converted to the type used in the provider.

FORMAT GetVariableType (<VariableName>)

< VariableName >	:	[in] variable name (VT_BSTR)
Return value	:	[out] variable type (VT_UI2). See Variable type. An error occurs when a non-existent variable is specified.

E.g.

```
Object obj = ctrl.Execute("GetVariableType", "Var_A");
```

3.1.2. **CaoController::Execute("GetStructName") command**

Obtain a structure name.

FORMAT GetStructName (<VariableName>)

< VariableName >	:	[in] variable name (VT_BSTR)
Return value	:	[out] structure name (VT_BSTR) An error occurs in the following cases: - when a non-existent variable is specified, or - when the variable type is structure type, or - when the variable type is other than the structured array

E.g.

```
string structName = (string)ctrl.Execute("GetStructName", "Var_A");
```

3.1.3. **CaoController::Execute("GetStructMemberNames") command**

Obtain a list of specified structure/union member names.

FORMAT GetStructMemberNames (<StructName>)

< StructName >	:	[in] structure/union name (VT_BSTR)
Return value	:	[out] Structure/union member names list (VT_BSTR | VT_ARRAY). An error occurs when a non-existent structure or union is specified.

E.g.

```
string[] varNames = (string[])ctrl.Execute("GetStructMemberNames", "STRUCT_A");
```

3.1.4. **CaoController::Execute("GetStructMemberType") command**

Obtain a variable type of structure/union member that is converted to the type used in the provider.

FORMAT GetStructMemberType (<MemberName>)

< MemberName >	:	[in] structure/union name and structure/union member name (VT_BSTR | VT_ARRAY or VT_VARIANT | VT_ARRAY) Specify a structure/union name for the first element and a structure/union member name for the second element, respectively.
Return value	:	[out] variable type of the structure/union member (VT_UI2). See Variable type. An error occurs in the following cases: - when a non-existent structure/union is specified, or, - when a structure/union member that is not exist in the structure/union is specified.

E.g.

```
Object obj = ctrl.Execute("GetStructMemberType",
new string[]{"STRUCT_A", "MEMBER_A"})
```

3.1.5. **CaoController::Execute("GetStructMemberStructName") command**

Obtain a structure name of a specified structure/union member.

FORMAT GetStructMemberStructName (<MemberName>)

< MemberName >	:	[in] structure/union name and structure/union member name (VT_BSTR | VT_ARRAY or VT_VARIANT | VT_ARRAY) Specify a structure/union name for the first element and a structure/union member name for the second element, respectively.
Return value	:	[out] structure name (VT_BSTR). An error occurs in the following cases: - when a non-existent structure/union is specified, or, - when a structure/union member that is not exist in the structure/union is specified.

E.g.

```
string structName = (string)ctrl.Execute("GetStructMemberStructName",
new string[]{"STRUCT_A", "MEMBER_A"})
```

3.1.6. **CaoController::Execute("GetArrayLength") command**

Obtain the number of elements in a specified array variable.

FORMAT GetArrayLength (<VariableName>)

< VariableName >	:	[in] variable name (VT_BSTR)
Return value	:	[out] number of array elements (VT_UI4 | VT_ARRAY). The number of elements in the return value stands for the number of array dimensions. An error occurs in the following cases: - when a non-existent variable is specified, or - when a specified variable is not an array.

E.g.

```
uint[] arrLen = (uint[])ctrl.Execute("GetArrayLength", "Var_A");
```

3.1.7. **CaoController::Execute("GetStructMemberArrayLength") command**

Obtain the number of array elements of a specified structure/union member.

FORMAT GetStructMemberArrayLength (<MemberName>)

< MemberName >	:	[in] structure/union name and structure/union member name (VT_BSTR | VT_ARRAY or VT_VARIANT | VT_ARRAY). Specify a structure/union name for the first element and a structure/union member name for the second element, respectively.
Return value	:	[out] number of array elements (VT_UI4 | VT_ARRAY). The number of elements in the return value stands for the number of array dimensions. An error occurs in the following cases: - when a non-existent structure/union is specified, or - when a structure/union member that is not exist in the structure/union is specified, or - when a specified structure/union member is not an array.

E.g.

```
uint[] arrLen = (uint[])ctrl.Execute("GetStructMemberArrayLength",
new string[]{"STRUCT_A", "MEMBER_A"})
```

3.1.8. **CaoController::Execute("GetArrayUBound") command**

Obtain the minimum array subscript of a specified variable.

FORMAT GetArrayUBound (<VariableName>)

< VariableName >	:	[in] variable name (VT_BSTR)
Return value	:	[out] minimum array subscript (VT_UI4 | VT_ARRAY). The number of elements in the return value stands for the number of array dimensions. An error occurs in the following cases: - when a non-existent variable is specified, or - when a specified variable is not an array.

E.g.

```
uint[] arrLen = (uint[])ctrl.Execute("GetArrayUBound ", "Var_A");
```

3.1.9. **CaoController::Execute("GetStructMemberArrayUBound") command**

Obtain the minimum array subscript of a specified structure/union member.

FORMAT GetStructMemberArrayUBound (<MemberName>)

< MemberName >	:	[in] structure/union name and structure/union member name (VT_BSTR | VT_ARRAY or VT_VARIANT | VT_ARRAY) Specify a structure/union name for the first element and a structure/union member name for the second element, respectively.
Return value	:	[out] minimum array subscript (VT_UI4 | VT_ARRAY). The number of elements in the return value stands for the number of array dimensions. An error occurs in the following cases: - when a non-existent structure/union is specified, or - when a structure/union member that is not exist in the structure/union is specified, or - when a specified structure/union member is not an array.

E.g.

```
uint[] arrLen = (uint[])ctrl.Execute("GetStructMemberArrayUBound",
new string[]{"STRUCT_A", "MEMBER_A"})
```

3.1.10. **CaoController::Execute("GetValue") command**

Obtain a variable value directly without using CaoVariable

FORMAT GetValue (<AccessPath>)

< AccessPath >	:	[in] access path (VT_BSTR). See Access path setting .
Return value	:	[out] variable value (VT_VARIANT). An error occurs in the following cases: - when a non-existent variable is specified, or - when a union/ structure member is specified directly, or - when an array with two or more dimensions is specified directly.

E.g.

```
Object obj = _ctrl.Execute("GetValue", "Var_A.MEMBER_A[0]")
```

3.1.11. **CaoController::Execute("PutValue") command**

Set a variable value directly without using CaoVariable

FORMAT PutValue(<Data>)

< Data>	:	[in] Access path and the value to be set (T_VARIANT | VT_ARRAY). Specify the access path for the first element (see Access path setting) and the value to be set for the second element, respectively.
Return value	:	[out] None. An error occurs in the following cases: - when a non-existent variable is specified, or - when a structure/union member is specified directly, or - when a specified value cannot be converted to the variable type.

E.g.

```
_ctrl.Execute("PutValue",
new object[]{ "Var_A.MEMBER_A[0]", "Value" });
```

3.1.12. **CaoController::Execute("GetVariableCIPType") command**

Obtain a variable type that is converted to the NJ-Series internal variable type.

FORMAT GetVariableCIPType (<VariableName>)

< VariableName >	:	[in] variable name (VT_BSTR)
Return value	:	[out] NJ-Series internal variable type (VT_UI1). See Variable type. If the variable is an array, variable type of the array element will be returned. An error occurs when a non-existent variable is specified.

E.g.

```
Object obj = ctrl.Execute("GetVariableCIPType", "Var_A");
```

3.1.13. **CaoController::Execute("GetStructMemberCIPType") command**

Obtain a variable type of structure/union member that is converted to the NJ-Series internal variable type.

FORMAT GetStructMemberType (<MemberName>)

< MemberName >	:	[in] structure/union name and structure/union member name (VT_BSTR | VT_ARRAY or VT_VARIANT | VT_ARRAY) Specify a structure/union name for the first element and a structure/union member name for the second element, respectively.
Return value	:	[out] structure/union member's variable type that is converted to the NJ-Series internal variable type. (VT_UI1). See Variable type. If the variable is an array, the variable type of the array element will be returned. An error occurs when a non-existent variable is specified.

E.g.

```
Object obj = ctrl.Execute("GetStructMemberCIPType",
new string[]{"STRUCT_A", "MEMBER_A"})
```

4. Error code list

OMRON NJ-Series provider specifies the unique error code as follows:

Table 4-1 Unique error code

Error name	Error No.	Description
Failed to change styles	0x80101001	Failed to change from CAO data style to NJ data style.
Styles other than being supported	0x80101002	NJ variable style is the other styles of being supported by the provider.
Incorrect Socket length	0x80101003	Socket length is incorrect.
Failed to read variable	0x80101004	Failed to read NJ variable.
Failed session register	0x80101006	Failed to register CIP session.
Failed to write variable	0x80101009	Failed to write NJ variable.
Error of specifying variable	0x8010100b	Variable specified to NJ does not exist.
Error of specifying structure type	0x8010100c	Variable that was specified when executing the command for structure operation is not the structure type.
Command execution error	0x8010100d	Failed to execute CIP command.
Error of specifying array	0x8010100f	Variable that was specified when executing the command for array operation is not the array.
Error of communication disconnection	0x80101010	Communication disconnection was found when executing CIP command.
Error of getting structure name	0x80101011	Specified variable is not the structure type when getting a structure name.
Union direct access error	0x80101012	Tried to directly access the union.
Array dimension specification error	0x80101013	The number of dimensions of the specified array is invalid.
Array index out of range error (lower)	0x80101014	The specified index is less than the minimum index.
Array index out of range error (upper)	0x80101015	The specified index exceeds the maximum index.
Structure size error	0x80101016	The size of the structure exceeds the upper limit.
Internal error	0x80101fff	Unexpected error occurred. Please contact DENSO WAVE service.

5. About the data that can be acquired when it directly accesses the structure

5.1. Method of direct access to structure

Specify the tag name of the array of the structure or the structure to access the structure directly.

5.2. Data type when it directly accesses structure

If it is an access to the unit when it directly accesses the structure, it is VT_UI1. | If it is VT_ARRAY, and an array, it is VT_VARIANT. | It is VT_UI1 of as many as one structure in each element of VT_ARRAY. | Data where VT_ARRAY is stored can be acquired.

5.3. Computational method of offset of member of structure

The structure of the CJ type and the structure of the NJ type can be specified for the structure of the NJ series. The structure of the CJ type can confirm byte offset on Sysmac Studio, and should calculate the offset value for myself for the structure of the NJ type.

The rule of the calculation of the offset of the structure of the NJ type becomes as follows.

- The member's offset should be a multiple of the size of the type.
    - However, when the member is a structure, it becomes the multiple of the biggest type included in the structure.
- The size of the structure should be a multiple of the size of the biggest data in the structure.

When the structure of the NJ type is used, we will recommend the member to be arranged in large the order because the calculation becomes complex if the big size member is arranged after the small size member.

E.g.) When the following structures exist, the offset is as follows.

Membername	Size	Offset in calculation	Actual offset that considers alignment
boolData	2	0	0
byteData	1	2	2
wordData	2	3	Four (multiple of the smallest two in three or more)
dwordData	4	6	Eight (multiple of the smallest four in six or more)
lwordData	8	12	16 (multiple of the smallest eight in 12 or more)

6. Method of specifying access passing

6.1. Specification of variable identifier

The variable with types other than the structure can specify the access passing by the value acquired with CaoController::GetVariableNames.

CaoVariable::Value made by the access passing specified by the variable identifier can be read and be written.

However, it is necessary to pass the array that the number of elements is the same as the number of elements of variables when writing it in the array.

6.2. Specification of structure membername

The variable with the type of the structure is variable identifier + ".". The access passing can be specified with + structure membername.

In addition to the above-mentioned access passing "." when the member of the structure is a structure Specify the membername of the structure that is + nest.

6.3. Specification of array element

The array element specifies the affixing character of the array by using large parentheses).

The lower bound of the affixing character that can be specified can be acquired with GetArrayLBound or GetStructMemberArrayLBound.

The upper bound of the affixing character that can be specified is lower bound + of the affixing character (number of elements acquired with GetArrayLength or GetStructMemberArrayLength) It is ? 1.

(period) .."".. specify the affixing character by the delimitation when you specify the array of two dimensions or more. whether only the amount of the number of dimension specifies the affixing character

E.g.)When it accesses the element of (1,2) of MEMBER_D of the second element of variable Var with the following structures, it is specified, "Var 3. MEMBER_B.MEMBER_D 12" or " Var 3. MEMBER_B.MEMBER_D 1.2".

```
Struct STRUCT_A {
        INT                         MEMBER_A;
        STRUCT_B                    MEMBER_B
};
Struct STRUCT_B {       INT                         MEMBER_C;
        ARRAY[0..7,0..7] OF INT     MEMBER_D;
};
ARRAY[2..9] OF STRUCT_A         Var;
```

7. Variable type

7.1. Variable types obtained by GetVariableType / GetStructMemberVariableType

Variable types obtained by GetVariableType/GetStructMemberVariableType are expressed by variant type value of C++.

If the variable is an array, 8192(VT_ARRAY) is added to the value of variable type.

Table 5-1 Variable type list

VARTYPE	Value (decimal)	Description
VT_I1	16	Signed: 1 byte integer
VT_I2	2	Signed: 2 bytes integer
VT_I4	3	Signed: 4 bytes integer
VT_I8	20	Signed: 8 bytes integer
VT_R4	4	4 bytes real number
VT_R8	5	8 bytes real number
VT_BSTR	8	Character string
VT_UI1	17	Unsigned: 1 byte integer
VT_UI2	18	Unsigned: 2 bytes integer
VT_UI4	19	Unsigned: 4 bytes integer
VT_UI8	21	Unsigned: 8 bytes integer
VT_USERDEFINED	29	Structure/Union

7.2. Variable type obtained by GetVariableCIPType / GetStructMemberVariableCIPType

Variable types obtained by GetVariableCIPType/GetStructMemberVariableCIPType are expressed as follows. If the variable is an array, the variable type of values obtained will be the same as that of the array elements.

Table 5-2 Variable type list

NJ-Series internal variable type	Value (hexadecimal)	Description
BOOL	0xc1	Logical value
SINT	0xc2	Signed: 1 byte integer
INT	0xc3	Signed: 2 bytes integer
DINT	0xc4	Signed: 4 bytes integer
LINT	0xc5	Signed: 8 bytes integer
USINT	0xc6	Unsigned: 1 byte integer
UINT	0xc7	Unsigned: 2 bytes integer
UDINT	0xc8	Unsigned: 4 bytes integer
ULINT	0xc9	Unsigned: 8 bytes integer
REAL	0xca	4 bytes real number
LREAL	0xcb	8 bytes real number
STRING	0xd0	Character string
BYTE	0xd1	Unsigned: 1 byte integer
WORD	0xd2	Unsigned: 2 bytes integer
DWORD	0xd3	Unsigned: 4 bytes integer
LWORD	0xd4	Unsigned: 8 bytes integer
STRUCT	0xa2	Structure
UNION	0x0c	Union
ENUM	0x07	Enumeration
DATE_NSEC	0x08	Return the elapsed time from 1970-01-01 by nanosecond unit with unsigned 8 bytes integer.
TIME_NSEC	0x09	Return the elapsed time by nanosecond unit with unsigned 8 bytes integer.
DATE_AND_TIME_NSEC	0x0a	Return the elapsed time from 1970-01-01 00:00:00 by nanosecond unit with unsigned 8 bytes integer.
TIME_OF_DAY_NSEC	0x0b	Return the elapsed time from 00:00:00 by nanosecond unit with unsigned 8 bytes integer.

8. Sample program

This section introduces a simple program written by Visual C# designed for OMRON NJ-Series provider. The complete program is stored in the following folder for your reference.

" <ORiN2 SDK install folder>¥CAO¥ProviderLib¥OMRON¥NJ¥SAMPLE¥NJSample "

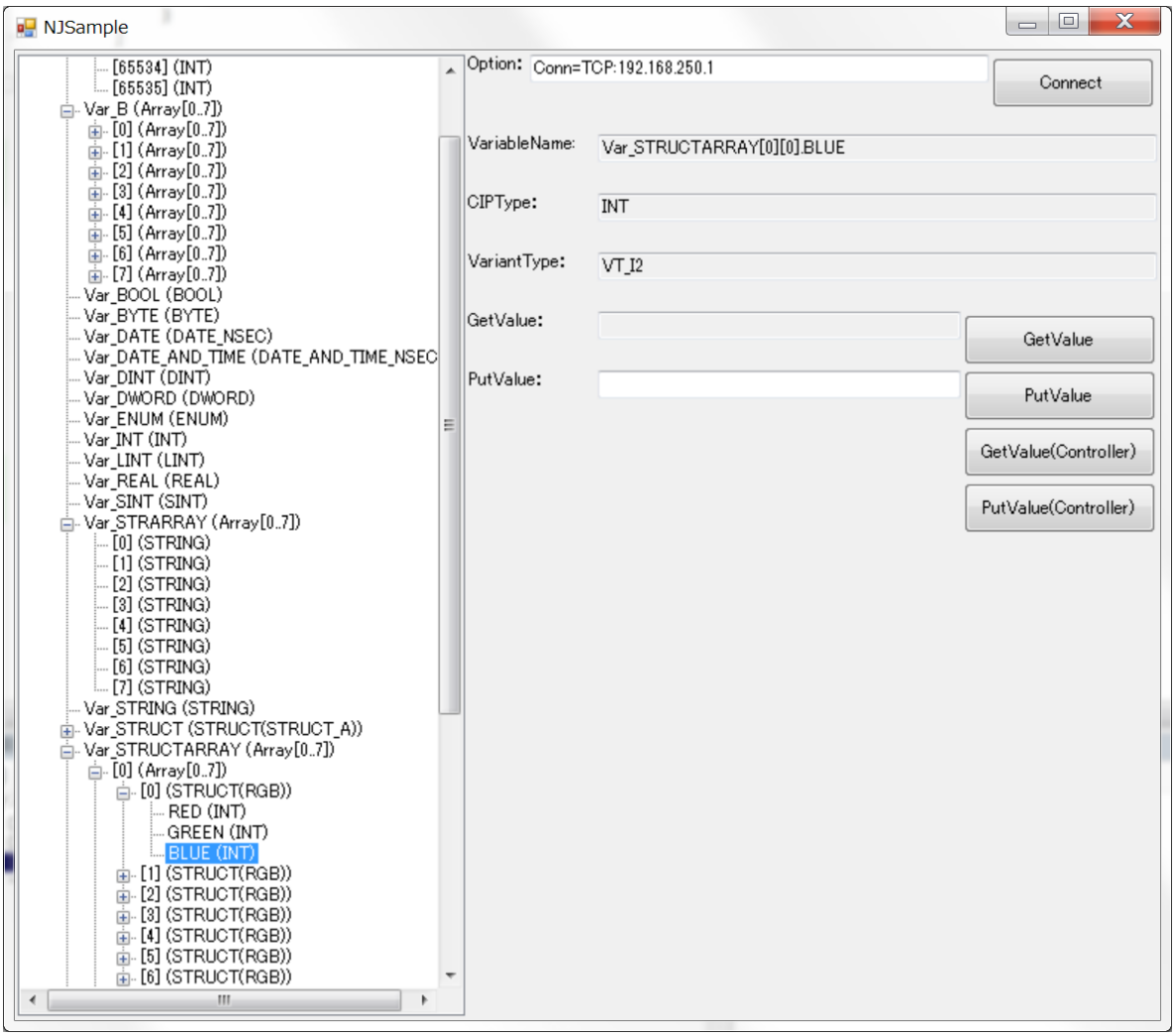


Fig. 8-1 NJ Sample window

9. Method of accessing another station using tag data link

This chapter describes how to use the tag data link function of the NJ series of OMRON to set the value of the child PLC from the parent PLC by the provider for OMRON NJ series. **When accessing other stations, data can be read, but data cannot be written to.**

9.1. Preparation in advance

In the procedure described in this chapter, in addition to the CPU unit, the expansion EtherNet/IP unit must be mounted on both the parent PLC and the child PLC. OMRON's application "Sysmac Studio" is used to set up the NJ series.

Install the expansion EtherNet/IP module in the NJ series as the parent PLC, and install the expansion EtherNet/IP module in the NJ series as the child PLC. Connect the PC and the built-in EtherNet/IP port on the CPU unit of the parent PLC with EtherNet cable, and connect the expansion EtherNet/IP unit of the parent PLC and the expansion EtherNet/IP unit of the extension PLC with EtherNet cable[3].

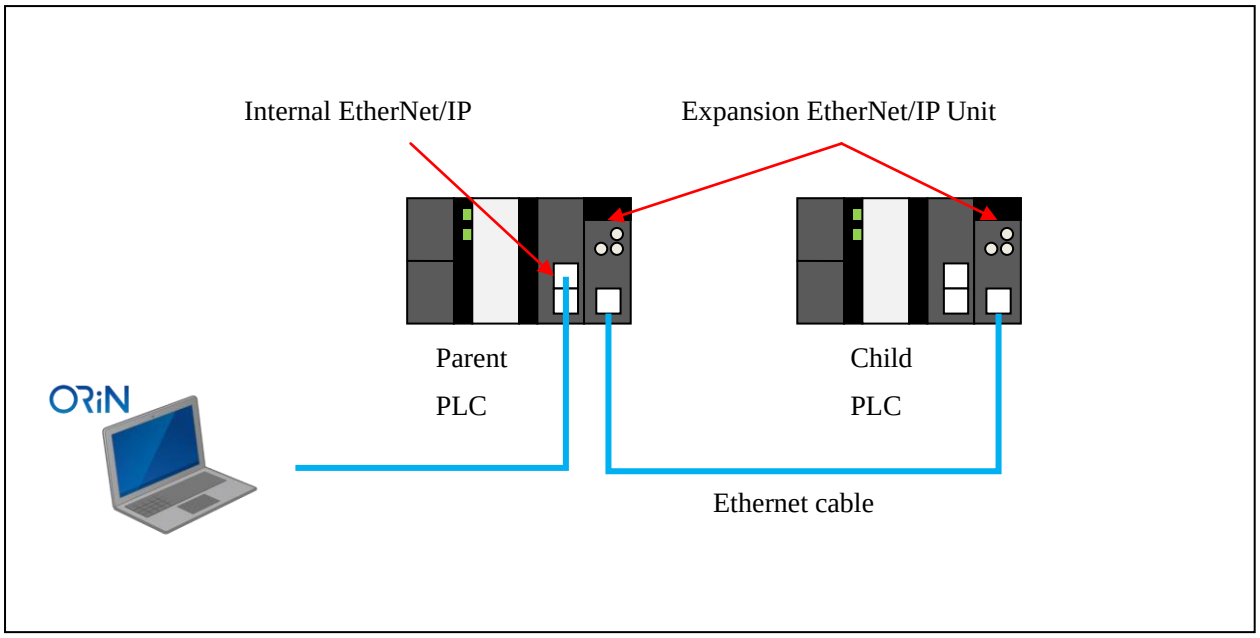


Fig. 9-1 is a schematic diagram of another station access method.

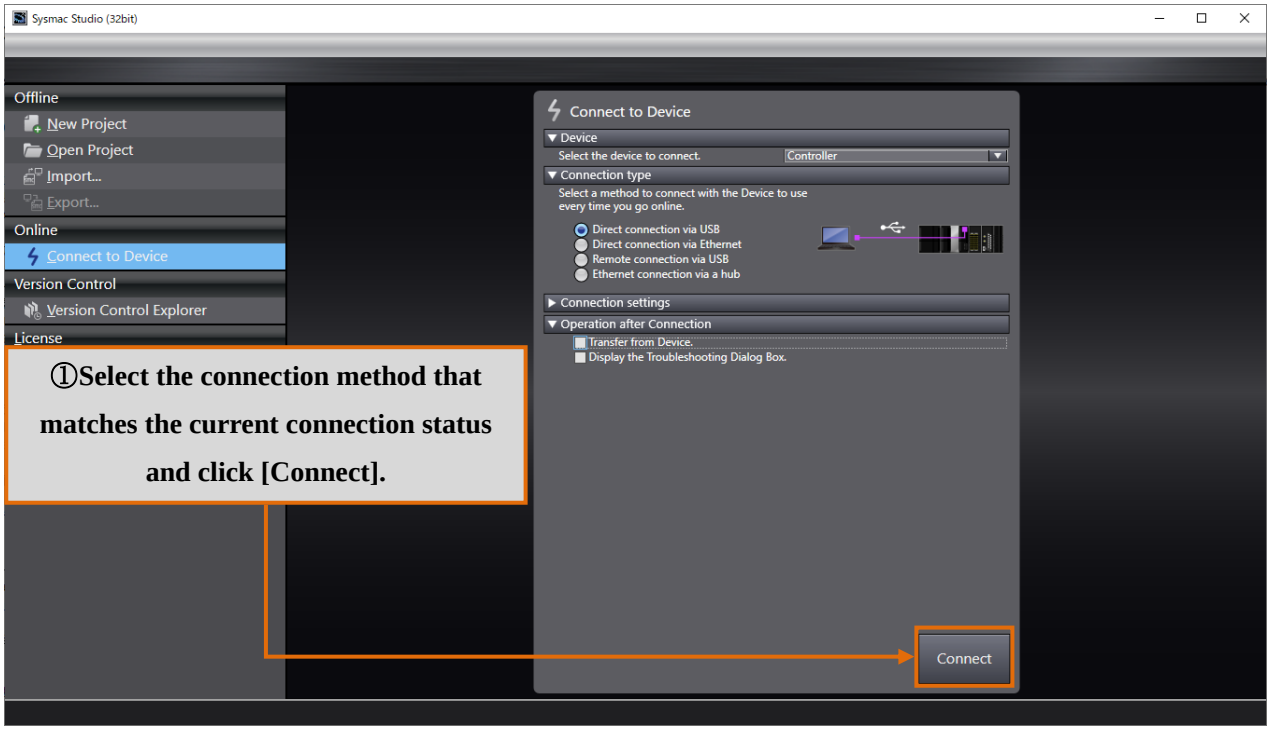
[3] When use CJ1W-EIP21 unit, the unit version must be Ver2.1 or higher.

9.2. Setting the parent PLC

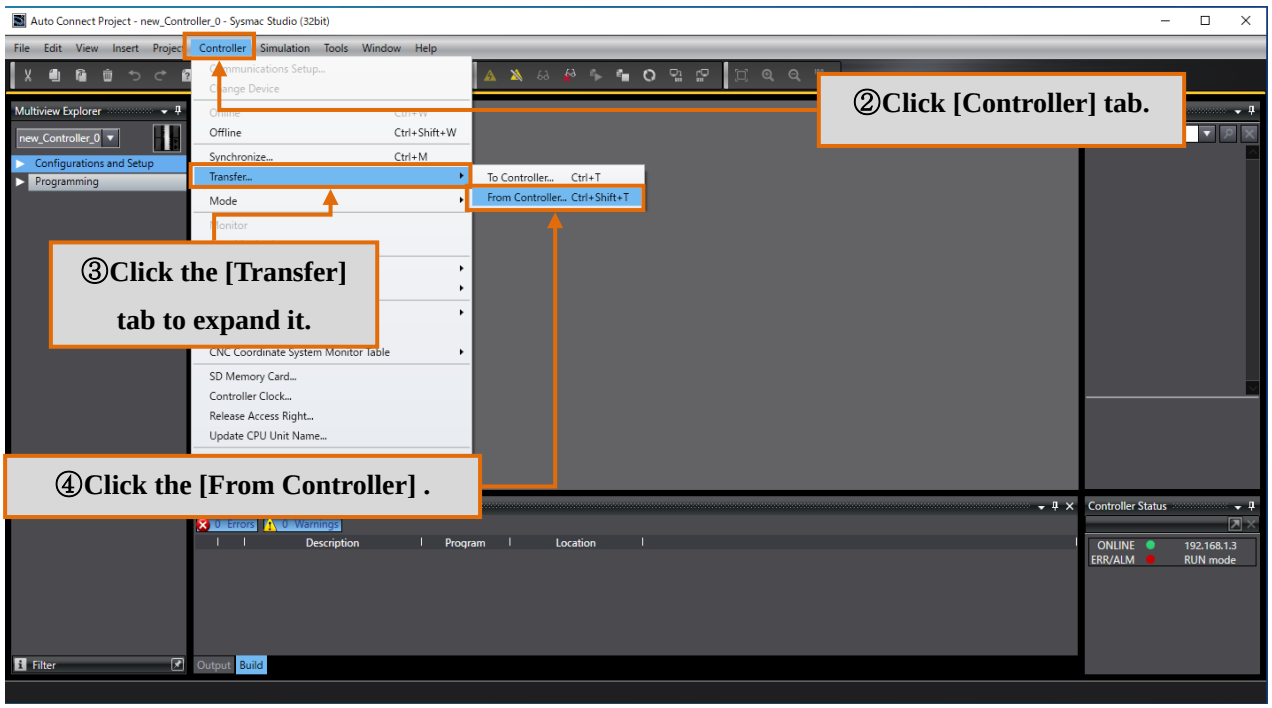
In this section, the settings of the parent PLC are made using Sysmac Studio.

9.2.1. Configuration Settings

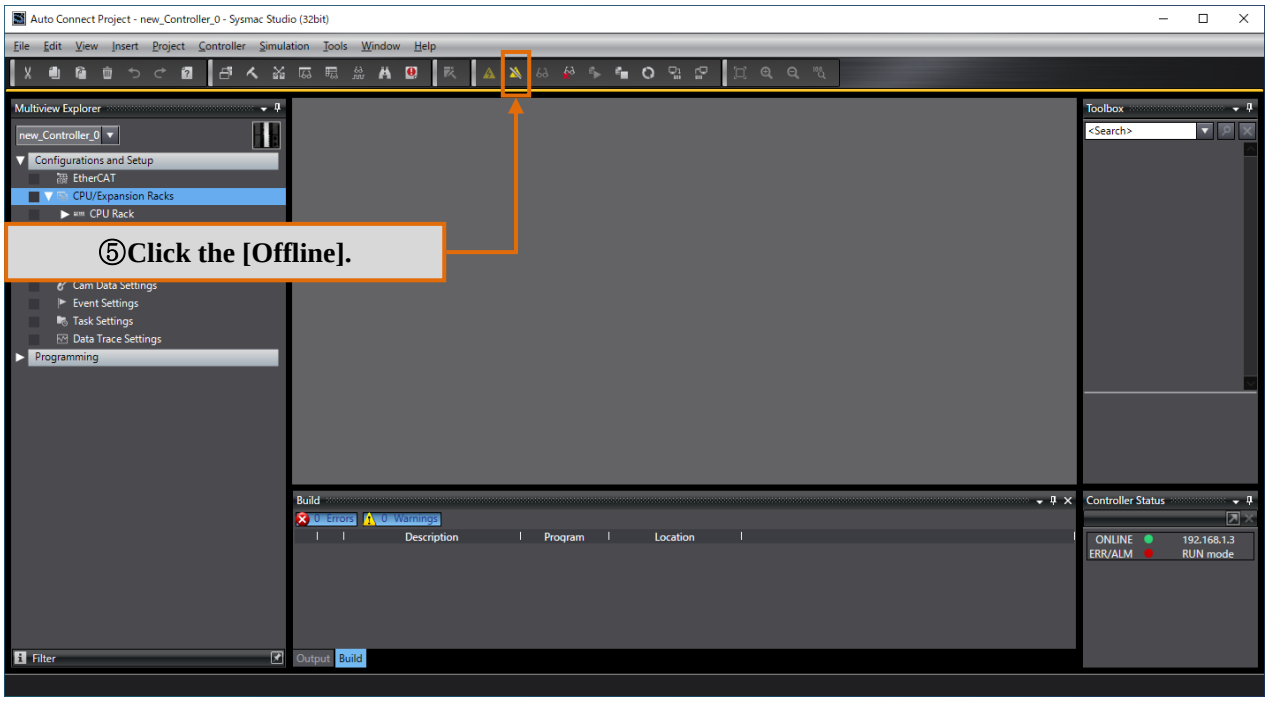
Start Sysmac Studio and connect to the parent PLC.



Gets the contents set in the PLC data.

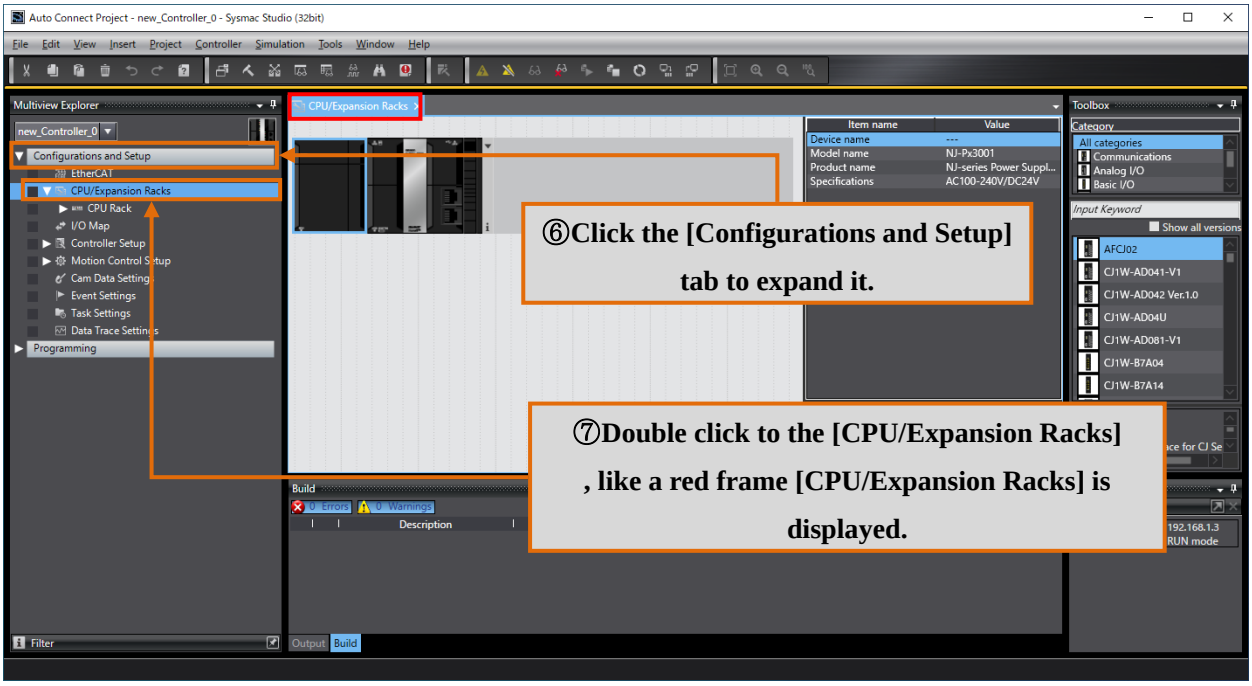


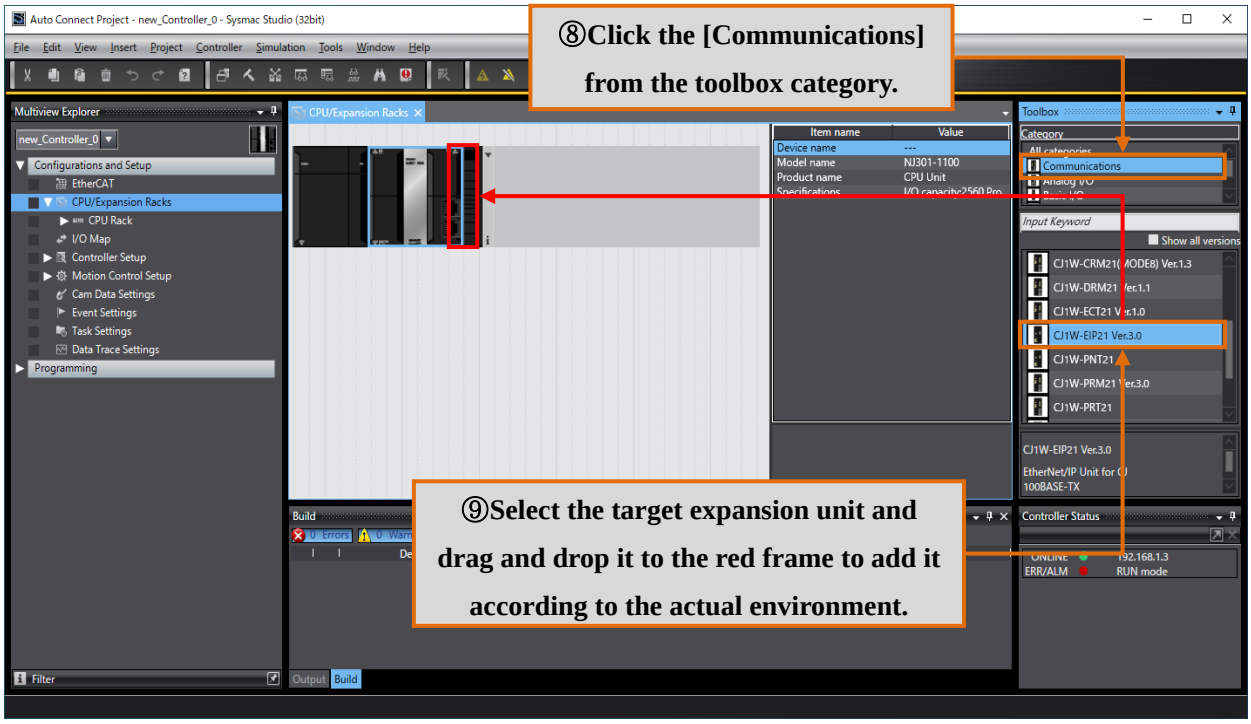
After the transfer is complete, go offline once to edit the settings.

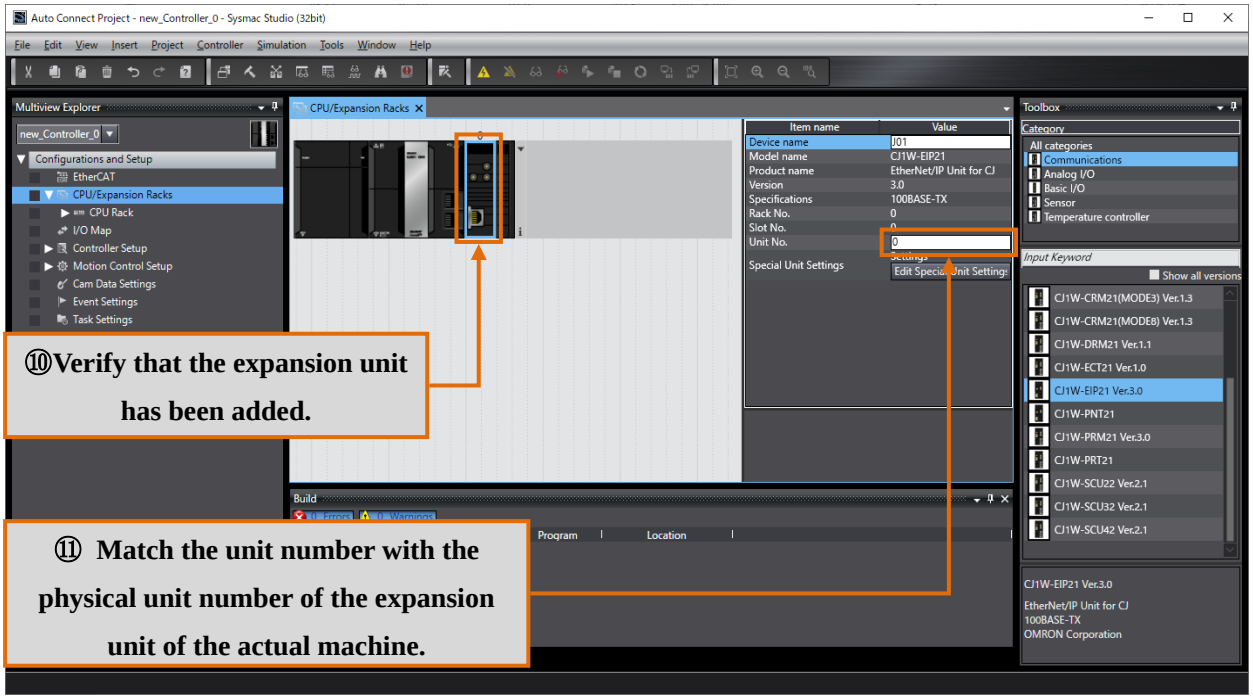


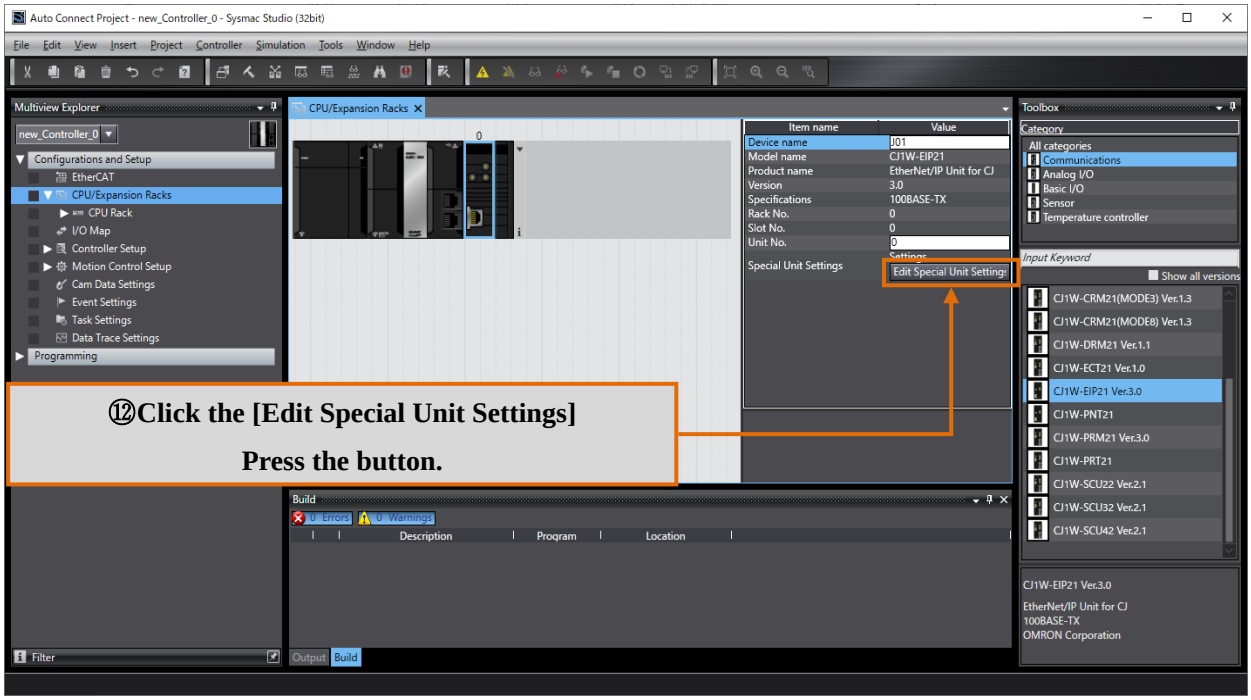
9.2.1.1. How to set up extended EtherNet/IP unit

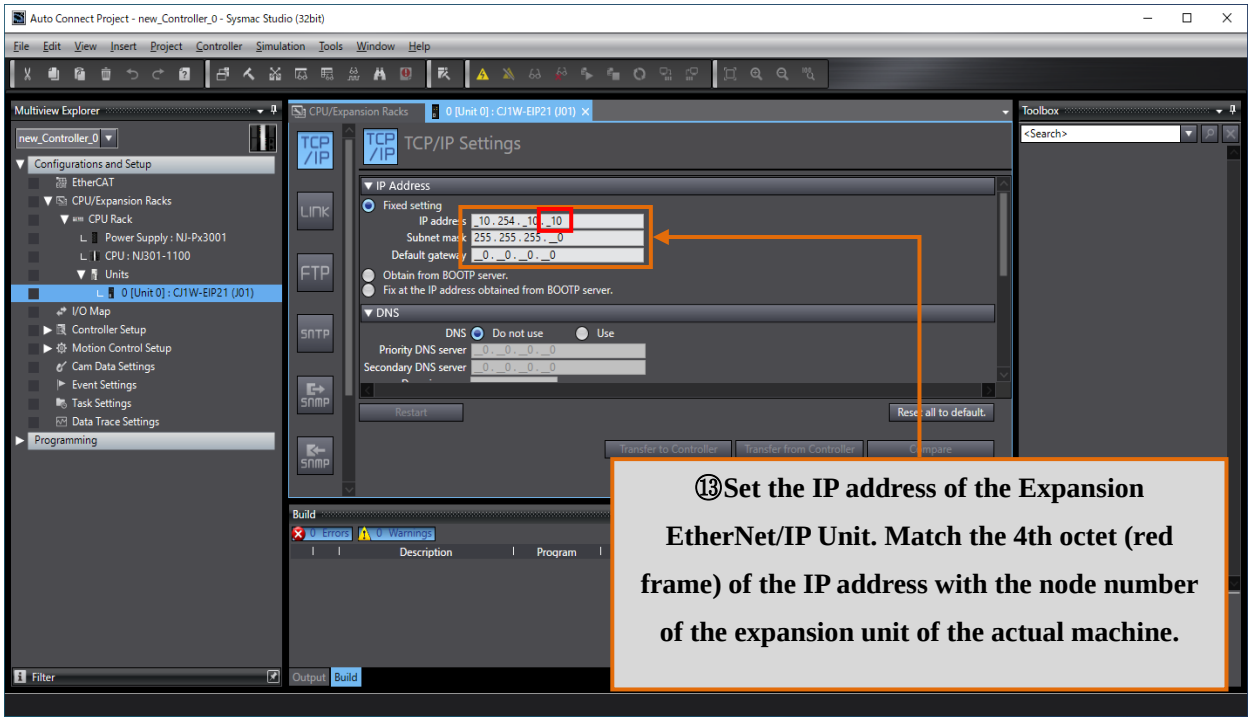
When the Expansion EtherNet/IP Unit is not recognized, set it from the Configuration/Settings tab. When it is recognized, proceed with the setting from the number ⑬ of the procedure.




⑧Click the [Communications] from the toolbox category.
⑨Select the target expansion unit and drag and drop it to the red frame to add it according to the actual environment.

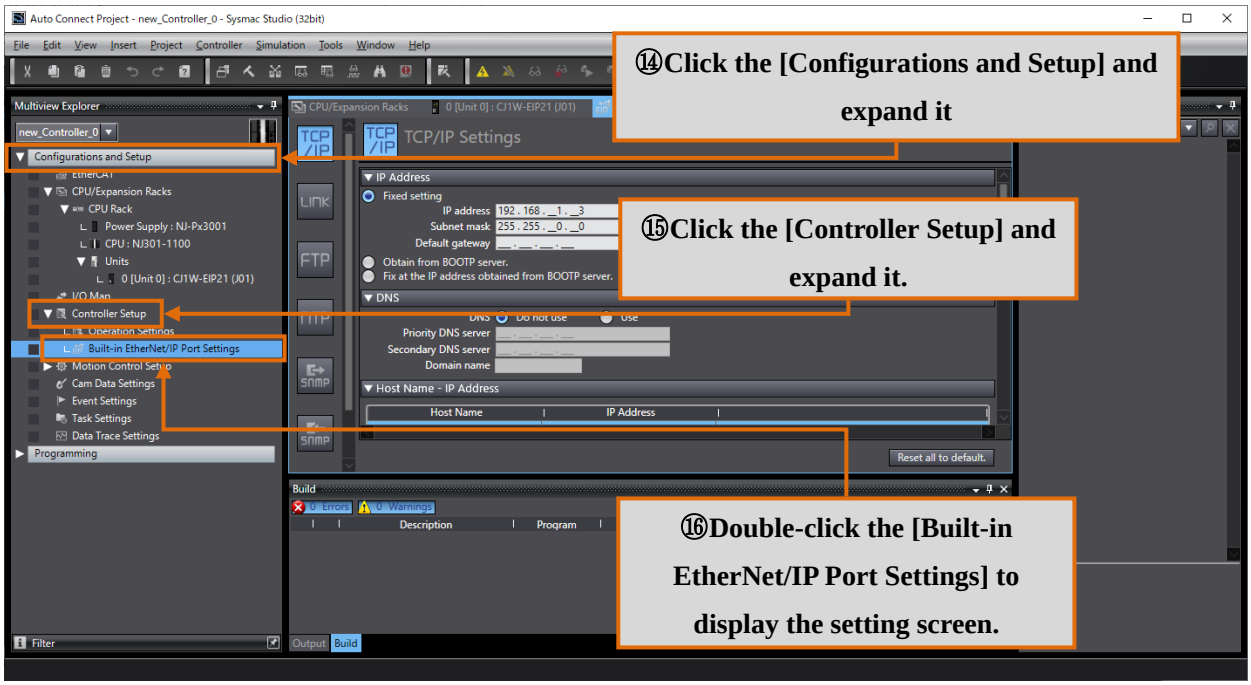

⑩Verify that the expansion unit has been added.
⑪ Match the unit number with the physical unit number of the expansion unit of the actual machine.

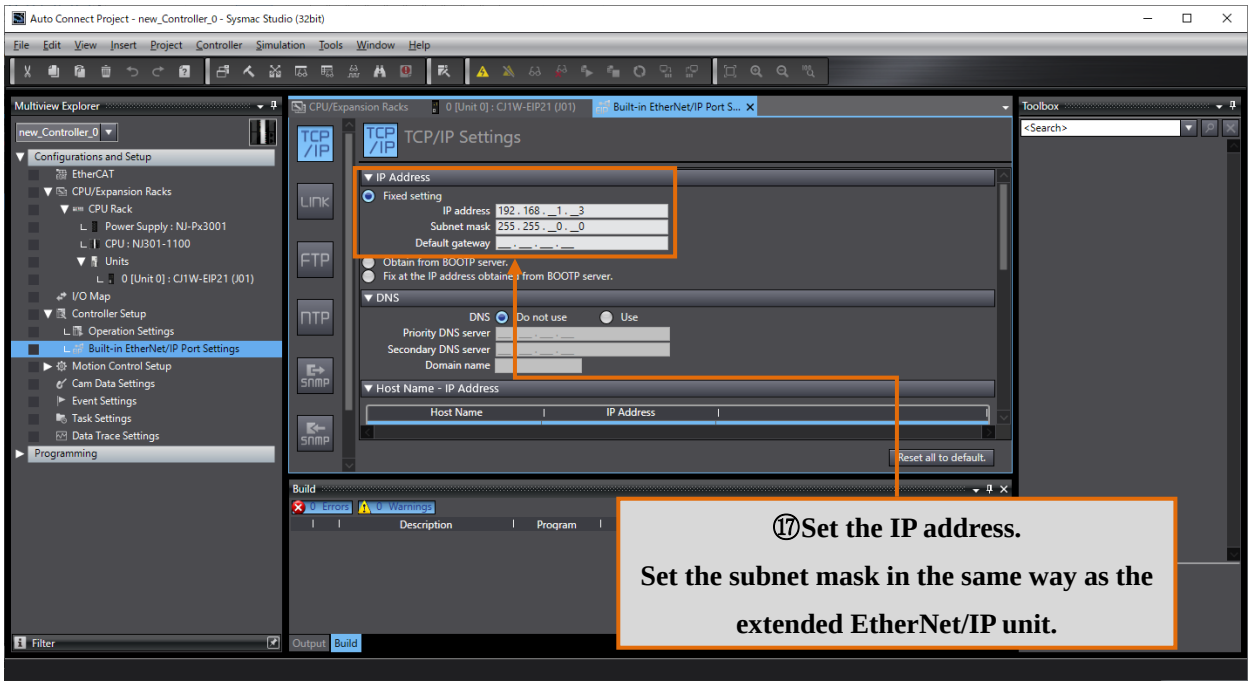

⑫Click the [Edit Special Unit Settings]
Press the button.


⑬Set the IP address of the Expansion EtherNet/IP Unit. Match the 4th octet (red frame) of the IP address with the node number of the expansion unit of the actual machine.

9.2.1.2. Setting the CPU Unit

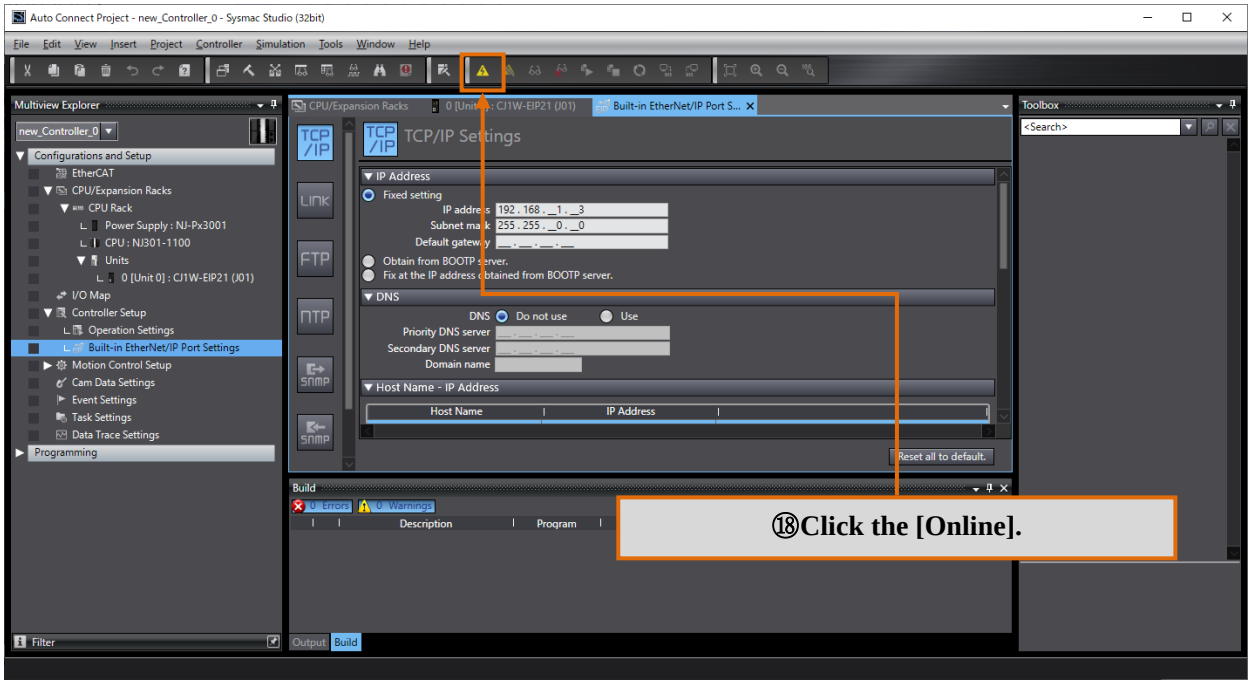
Set the built-in EtherNet/IP of the CPU module.

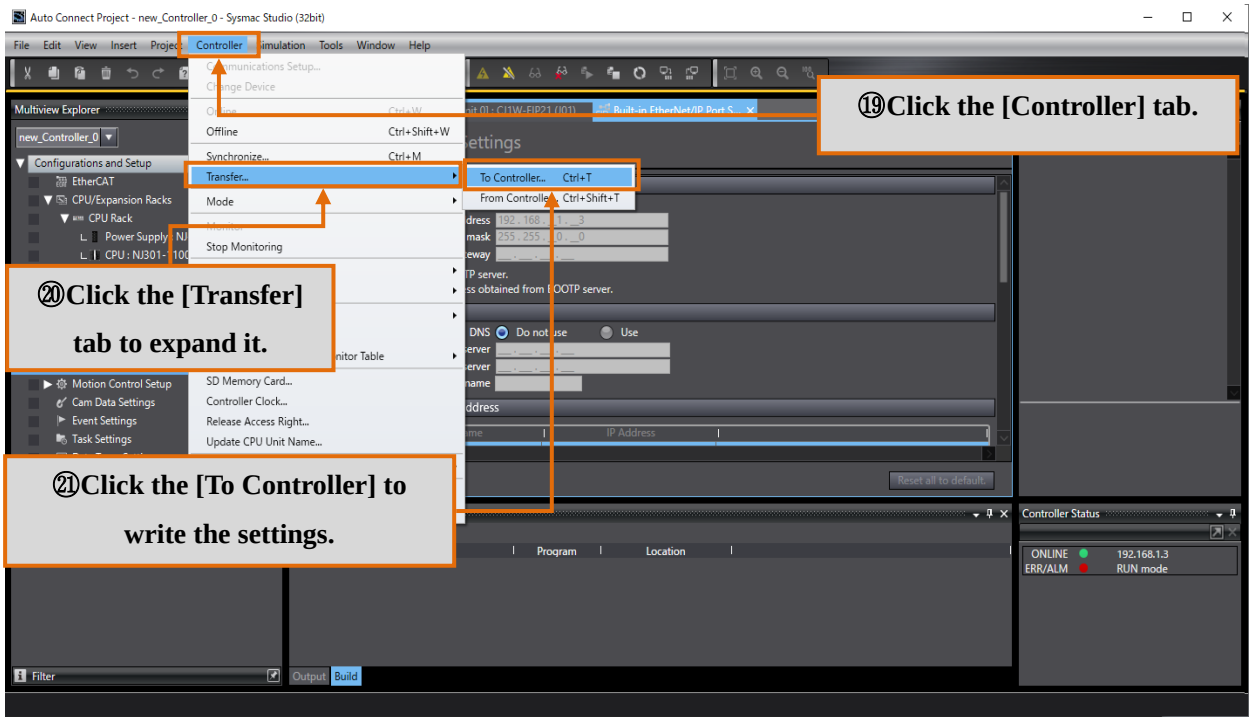




9.2.1.3. Write the settings to the device.

After the setting of each unit is completed, reflect the setting contents to the PLC.



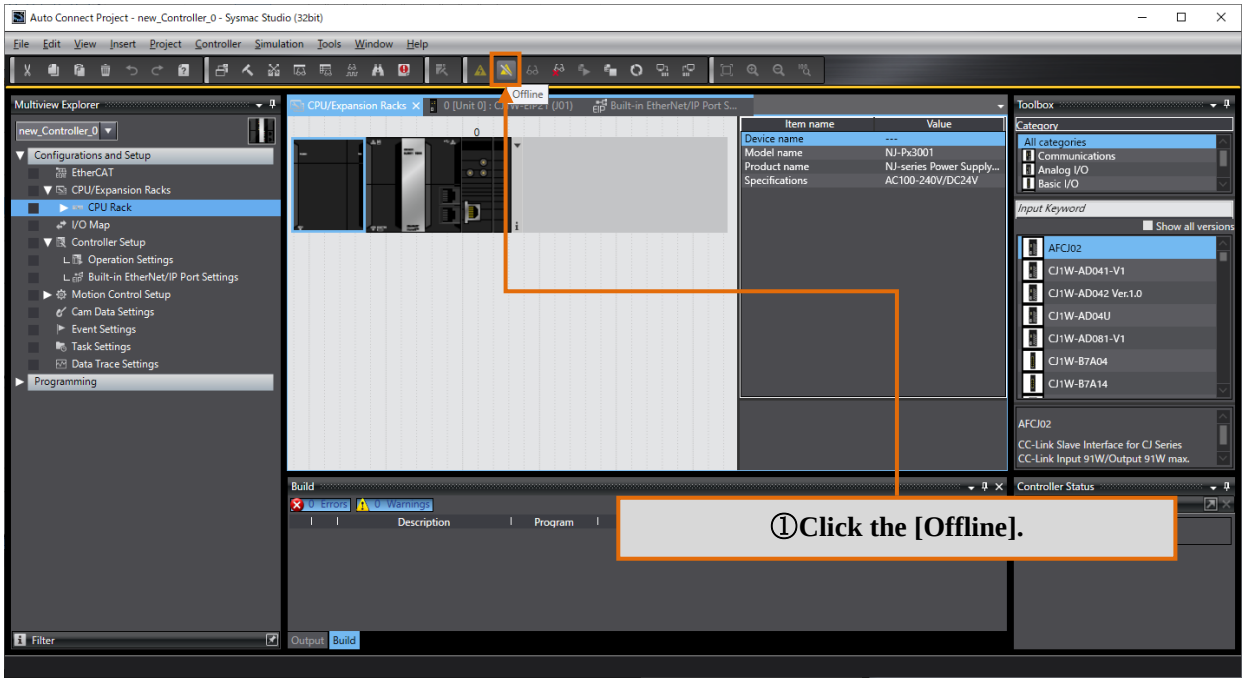


When the transfer to the PLC is completed, the PLC power supply is restarted, and if there is no error in the PLC, the configuration setting is completed.

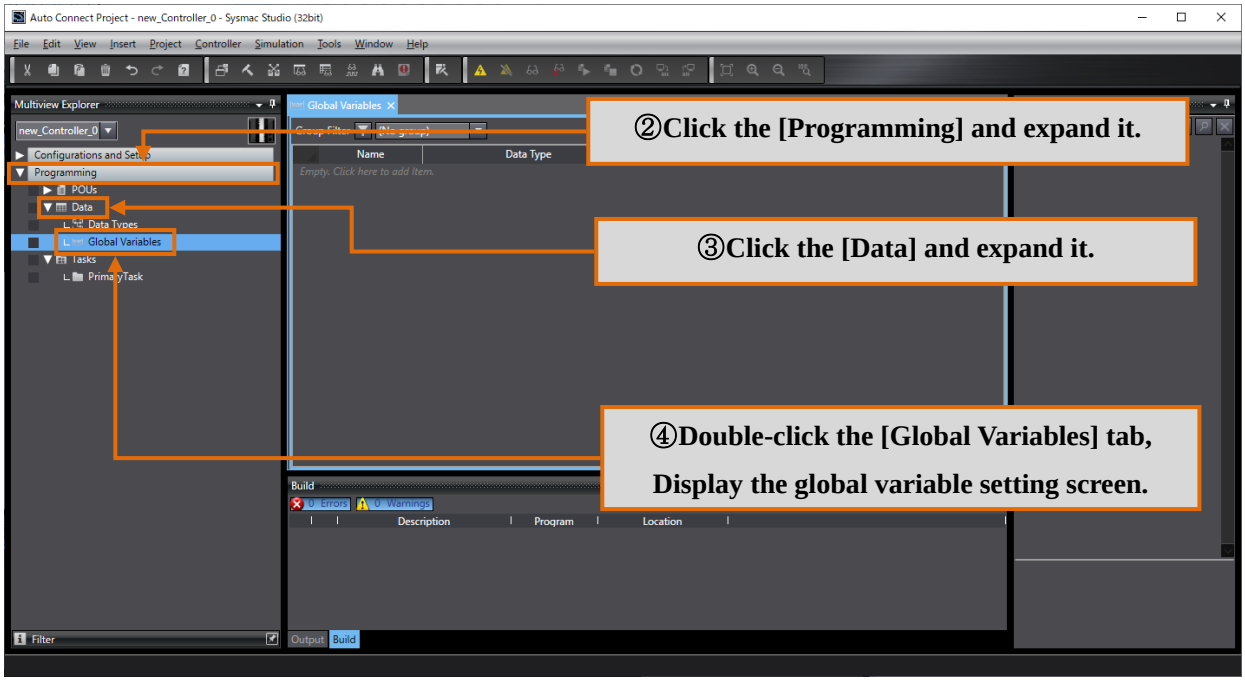
9.2.2. **Variable registration**

Registers the global variable for tag data link to the parent PLC.

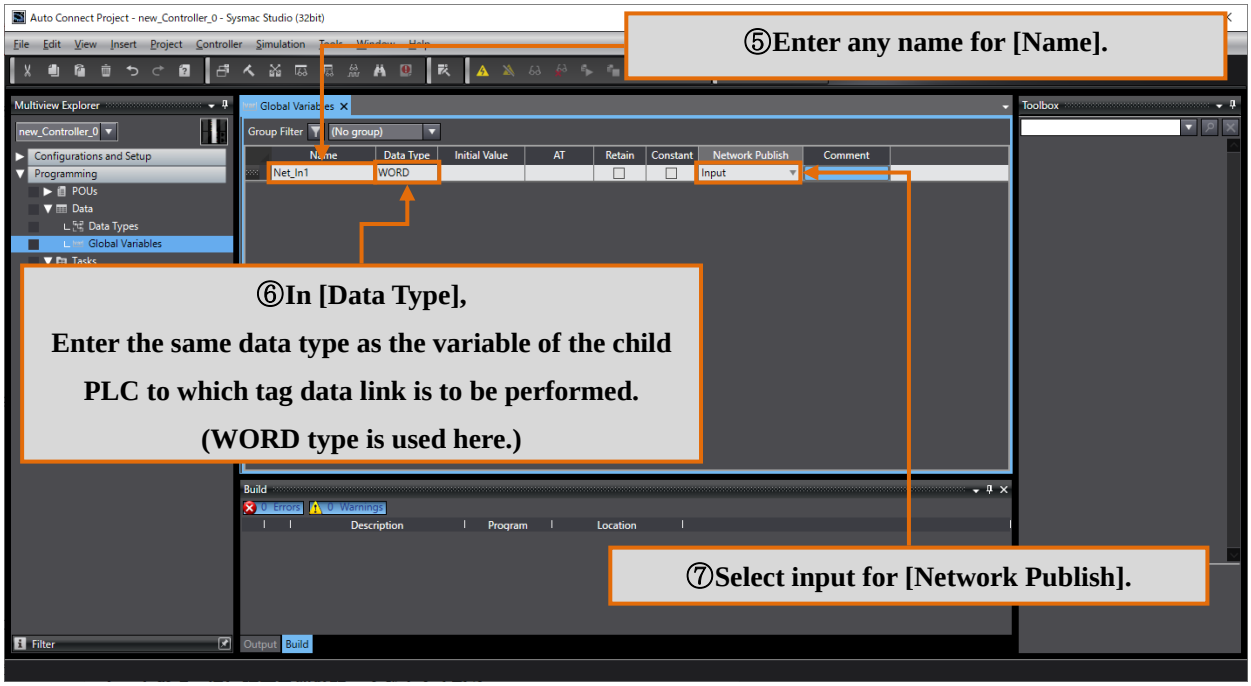
To edit the settings, click the offline when online status.



Next, create a global variable.



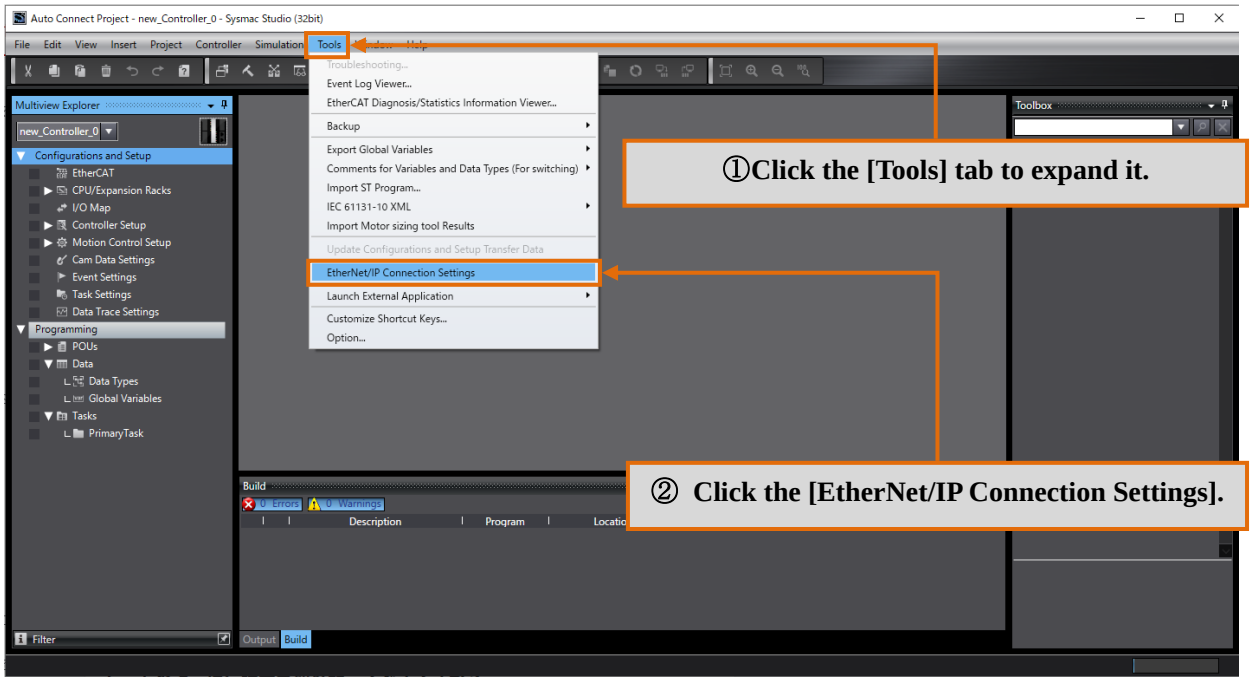
Register the global variable to be linked with the tag data and set it as follows. It is not necessary to set any items other than the following.

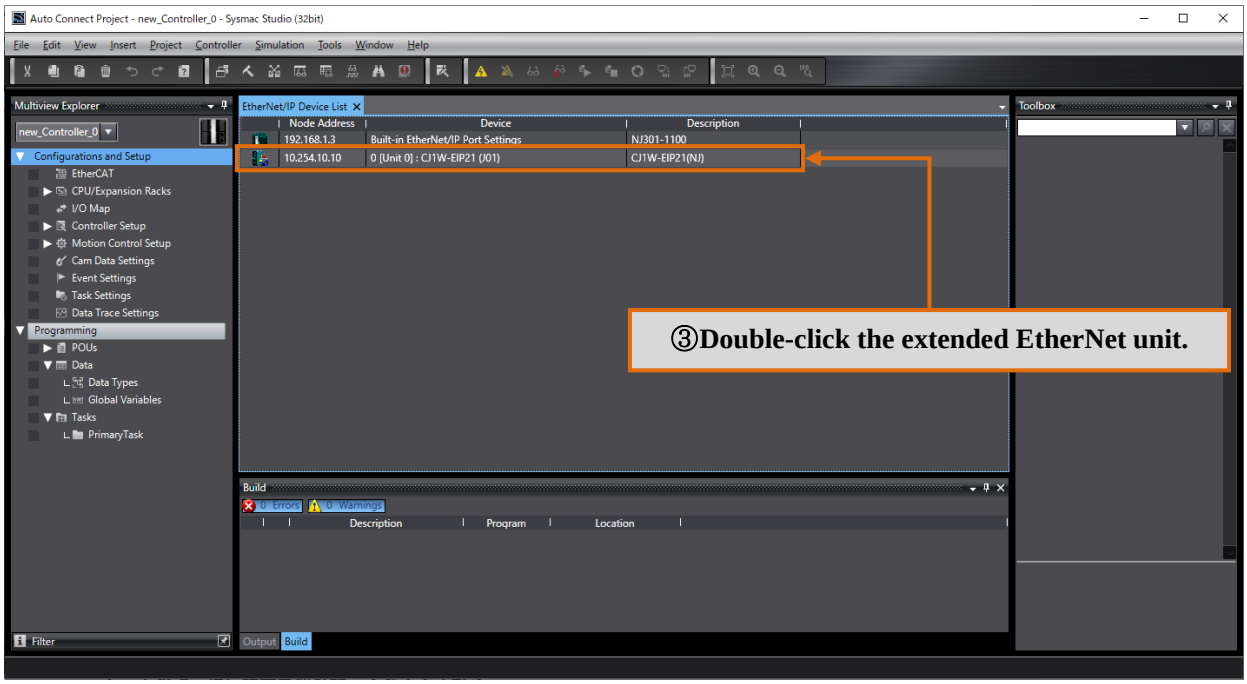


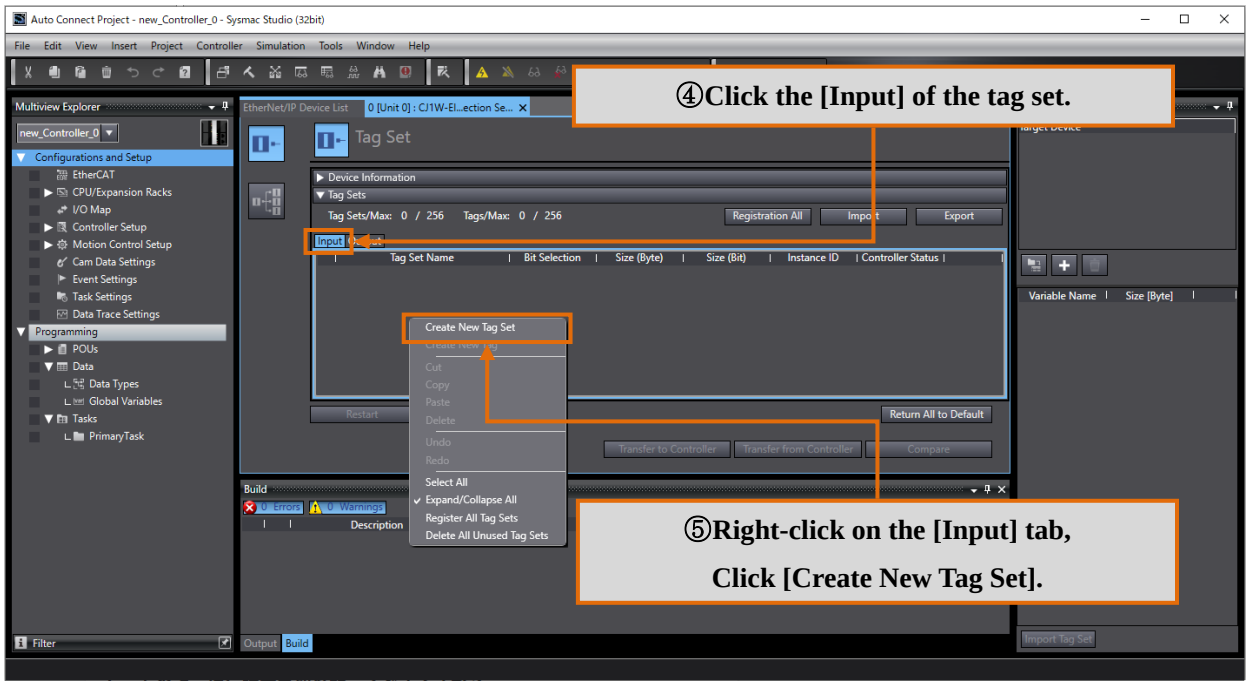
Write the setting to the parent PLC in the same way as 9.2.1.3Write the settings to the device.

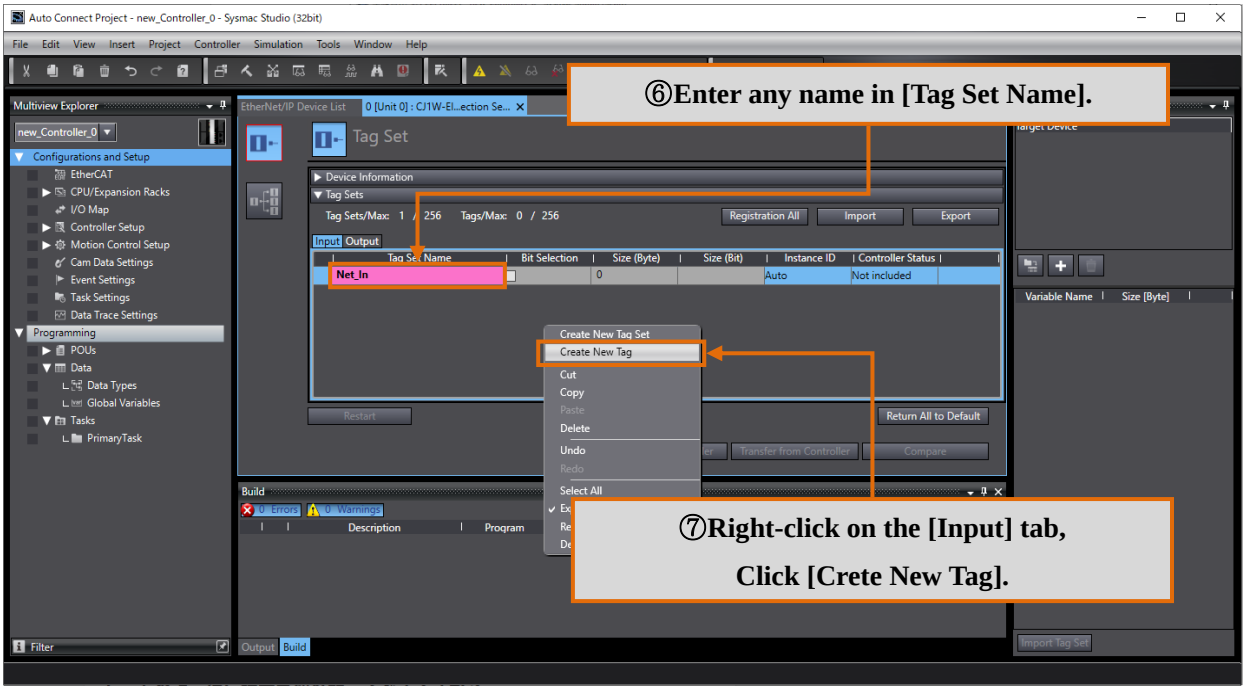
9.2.3. EtherNet/IP connection setting

From EtherNet/IP connection settings, configure the tag set for tag data linking. In this section, Sysmac Studio project for which the child PLC has been set is used. Complete the 9.3 How to set the child PLC, and save Sysmac Studio project used at the time of setting.

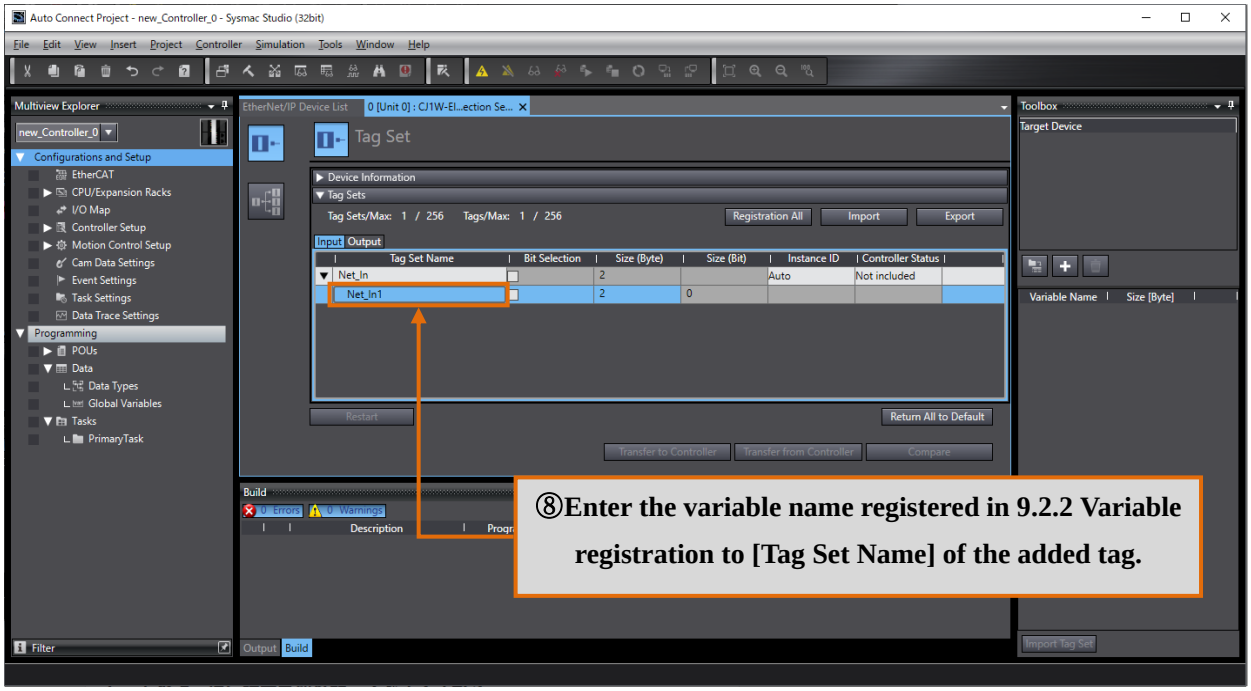




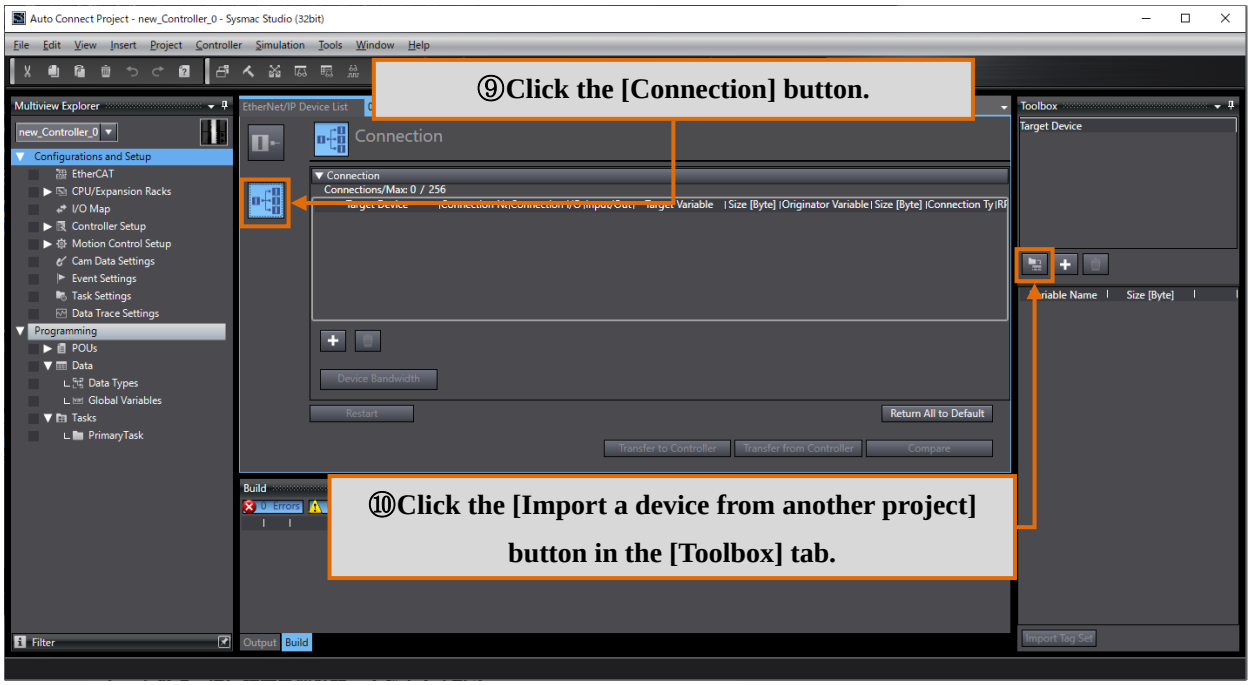

④Click the [Input] of the tag set.
⑤Right-click on the [Input] tab,
Click [Create New Tag Set].
Create New Tag Set
Cut
Copy
Paste
Delete
Undo
Redo
Select All
Expand/Collapse All
Register All Tag Sets
Delete All Unused Tag Sets


⑥Enter any name in [Tag Set Name].
⑦Right-click on the [Input] tab,
Click [Crete New Tag].
Net_In
Create New Tag Set
Create New Tag
Cut
Copy
Paste
Delete
Undo
Redo
Select All

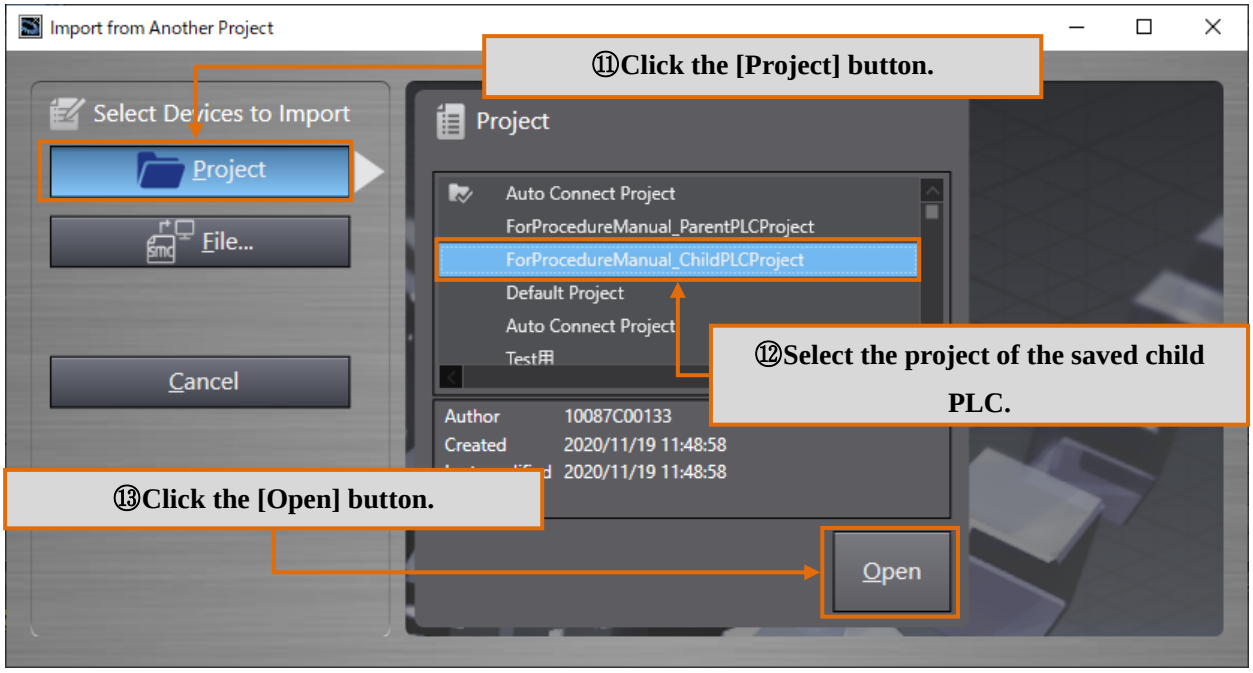
Sets the variable to which the tag data is linked to the tag set. Multiple variables can be registered in the tag set, but the same size, data type, and number of tags must be matched between the parent PLC and the child PLC.

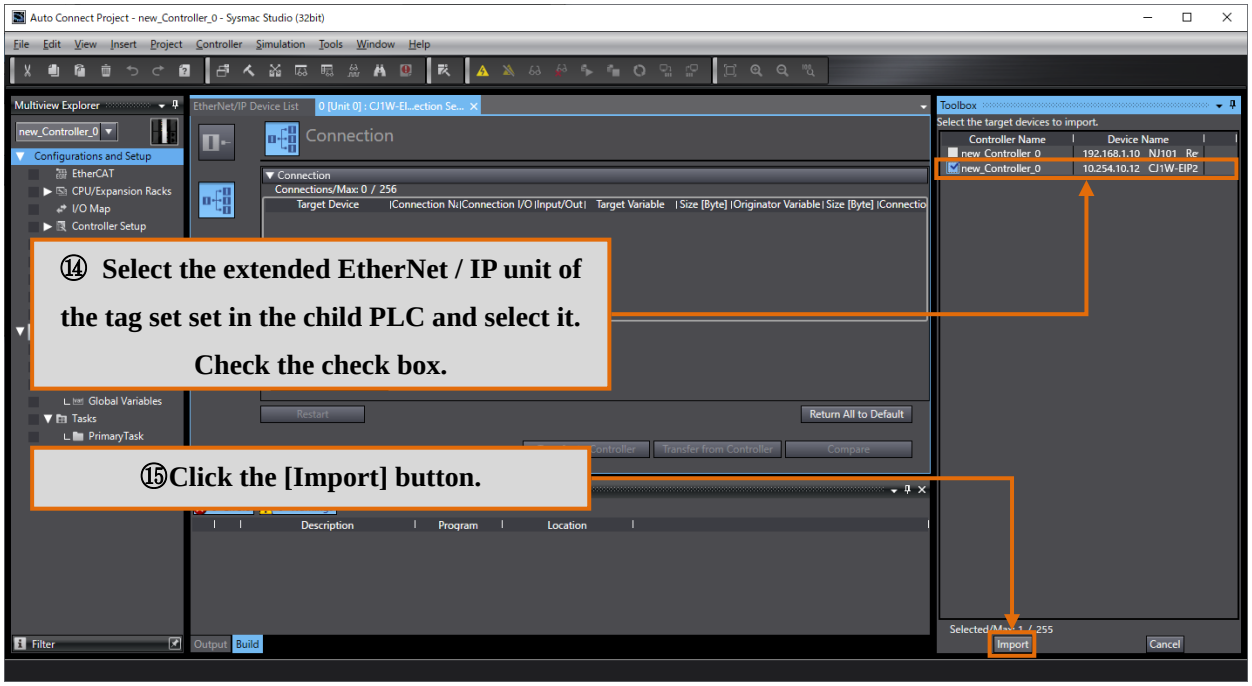


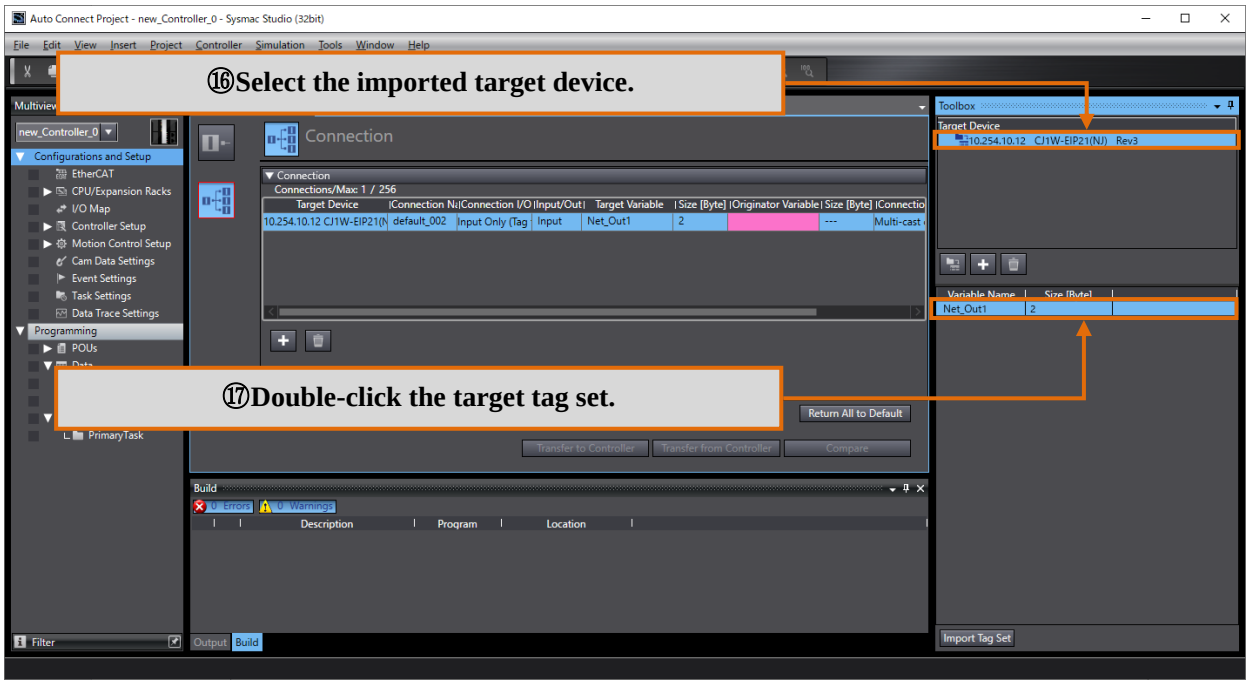
Next, connect the set tag set to the target child PLC.

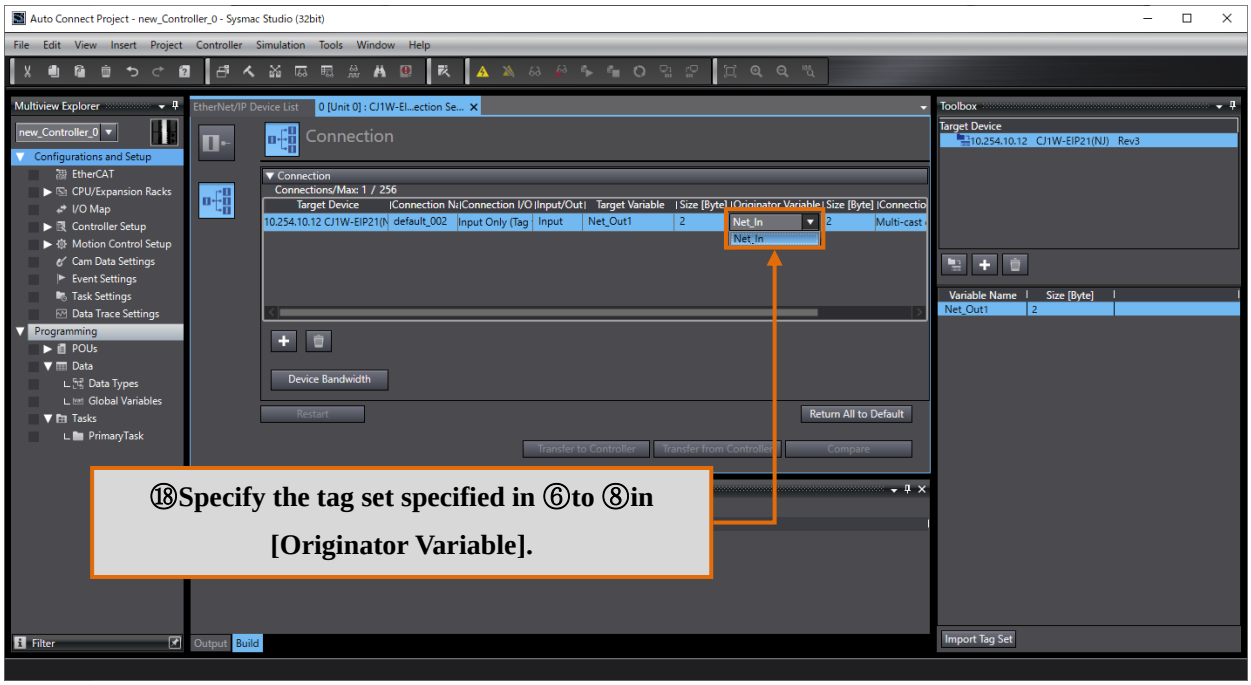


Reads tag set information from the project of the set child PLC.

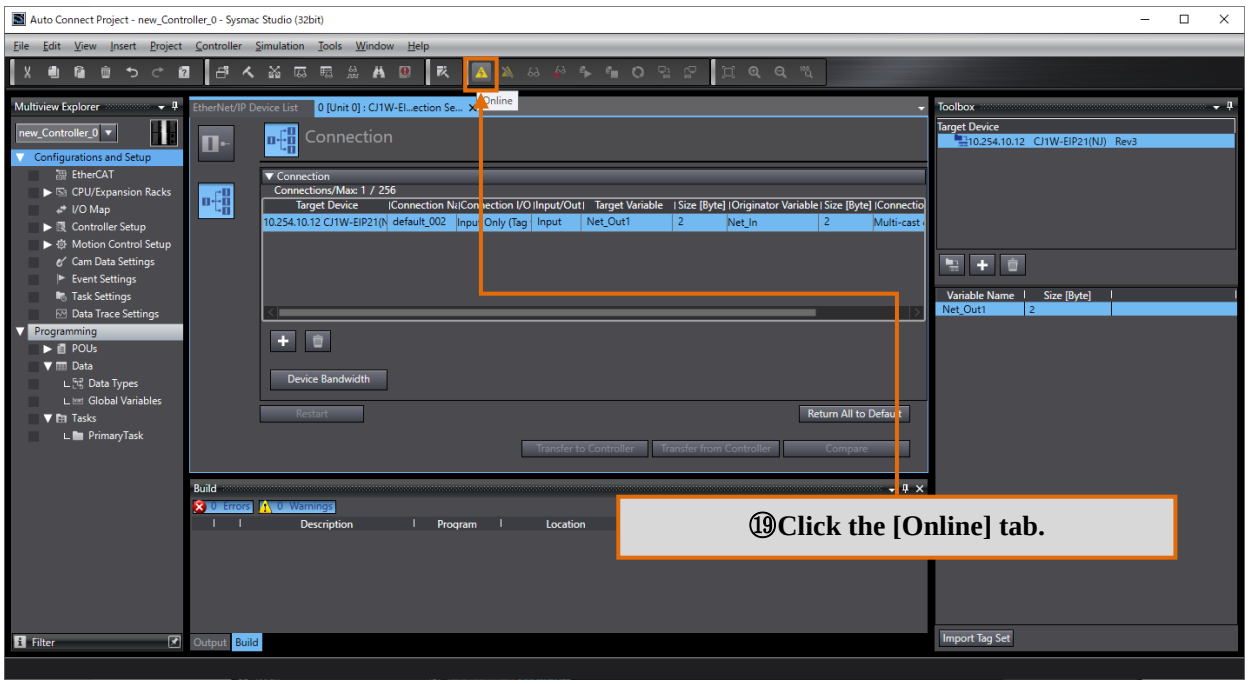


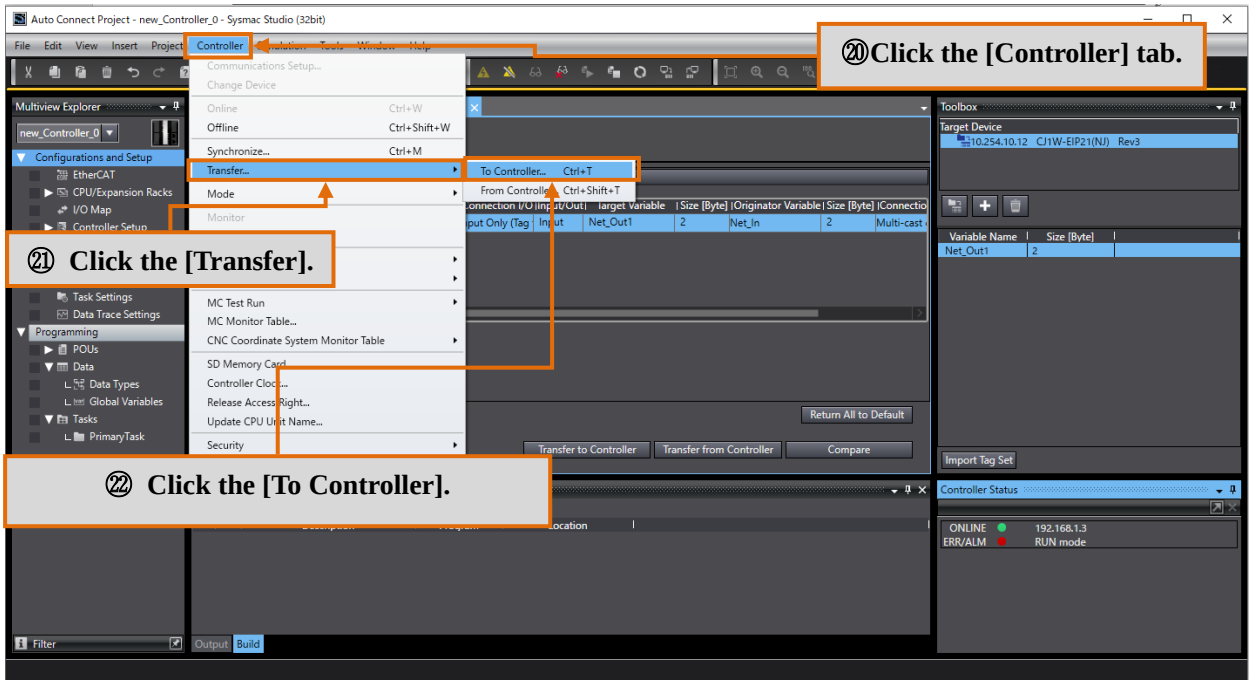



⑯Select the imported target device.
⑰Double-click the target tag set.

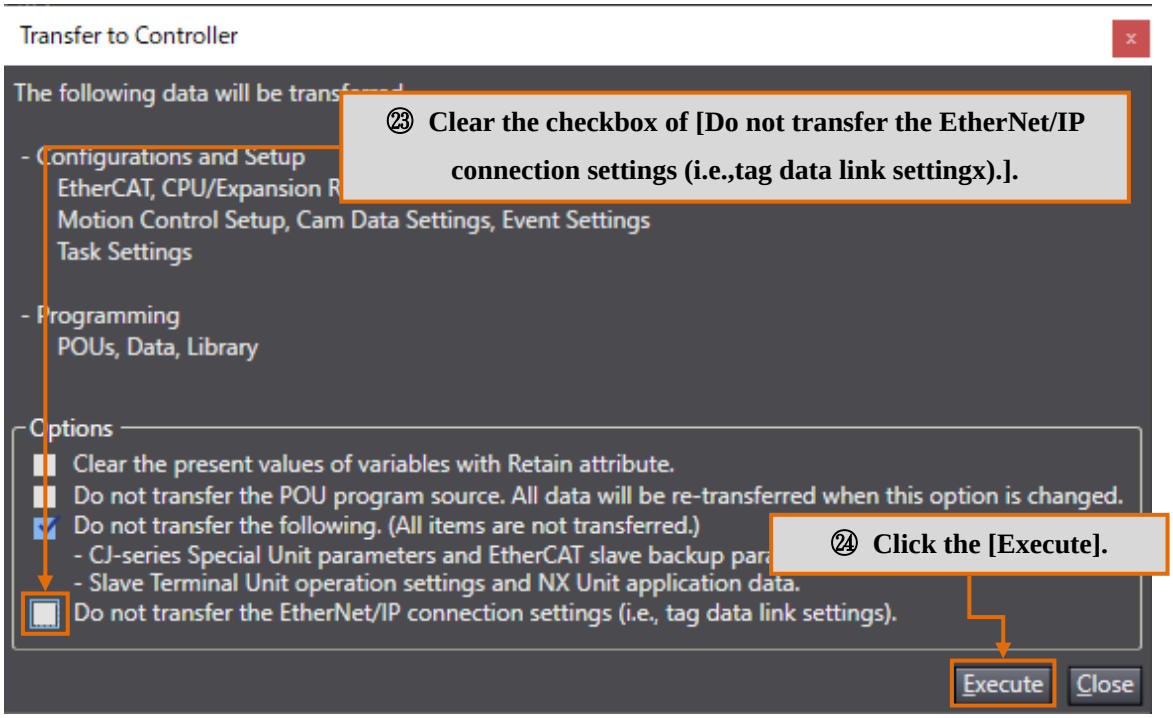

⑱Specify the tag set specified in ⑥to ⑧in [Originator Variable].

After configuring the tag set, apply the setting to the parent PLC.





When transferring data from a PC, the following window will be displayed. Clear the appropriate checkbox and transfer the data.



This completes the setting of the parent PLC.

9.3. How to set the child PLC

In this section, the settings of the PLCs are made using Sysmac Studio.

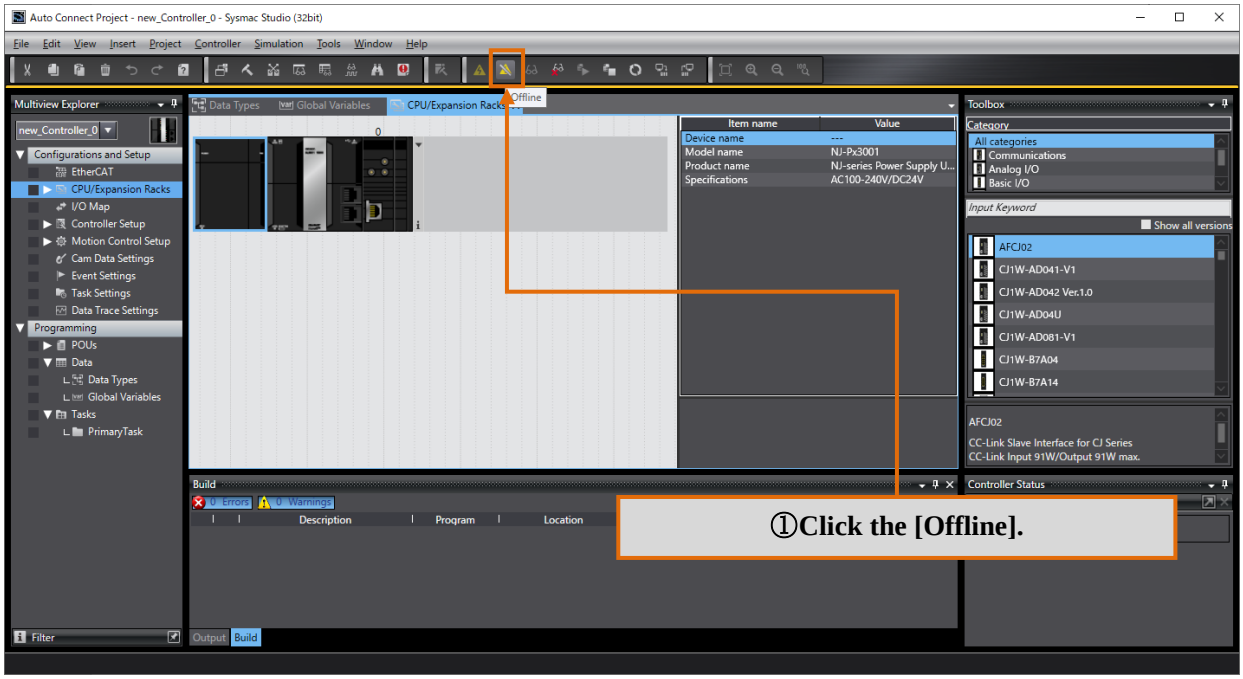
9.3.1. Configuration Settings

The configuration setting method of the child PLC is the same as the configuration setting of the parent PLC. For details, see section 9.2.1, Configuration Settings. However, the IP address must be set so that it does not correspond to the setting of the parent PLC.9.2.1

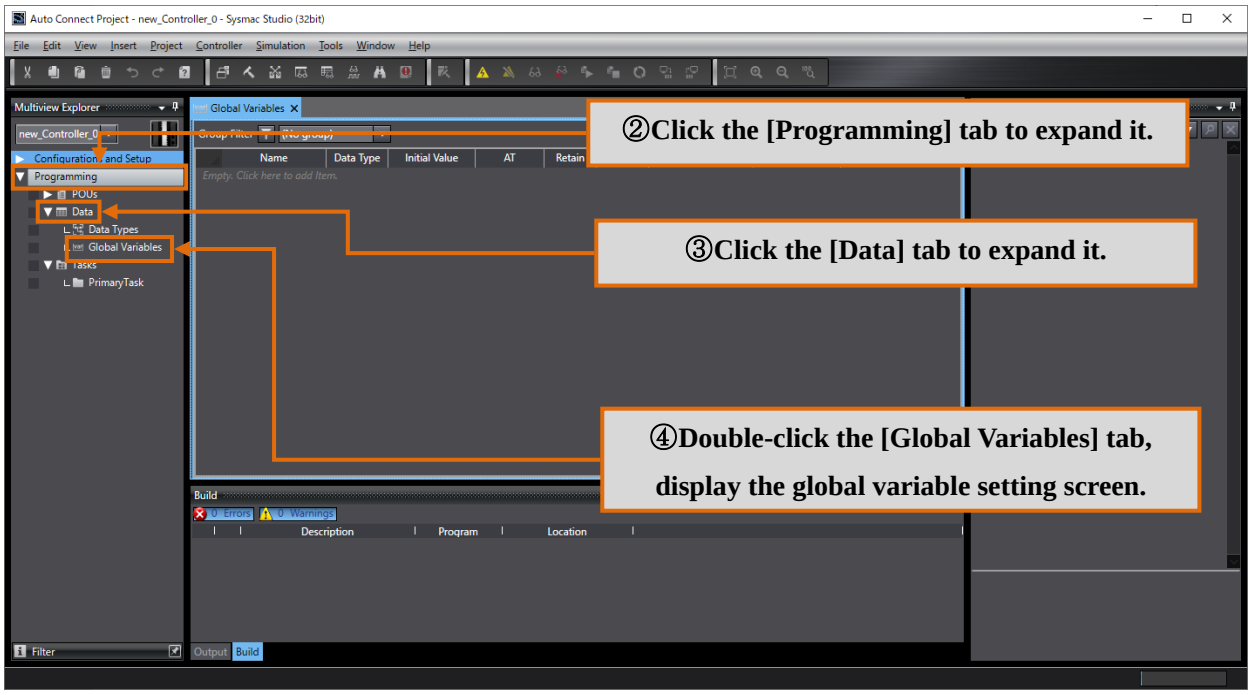
9.3.2. Variable registration

Register and set the global variable for tag data link to the child PLC.

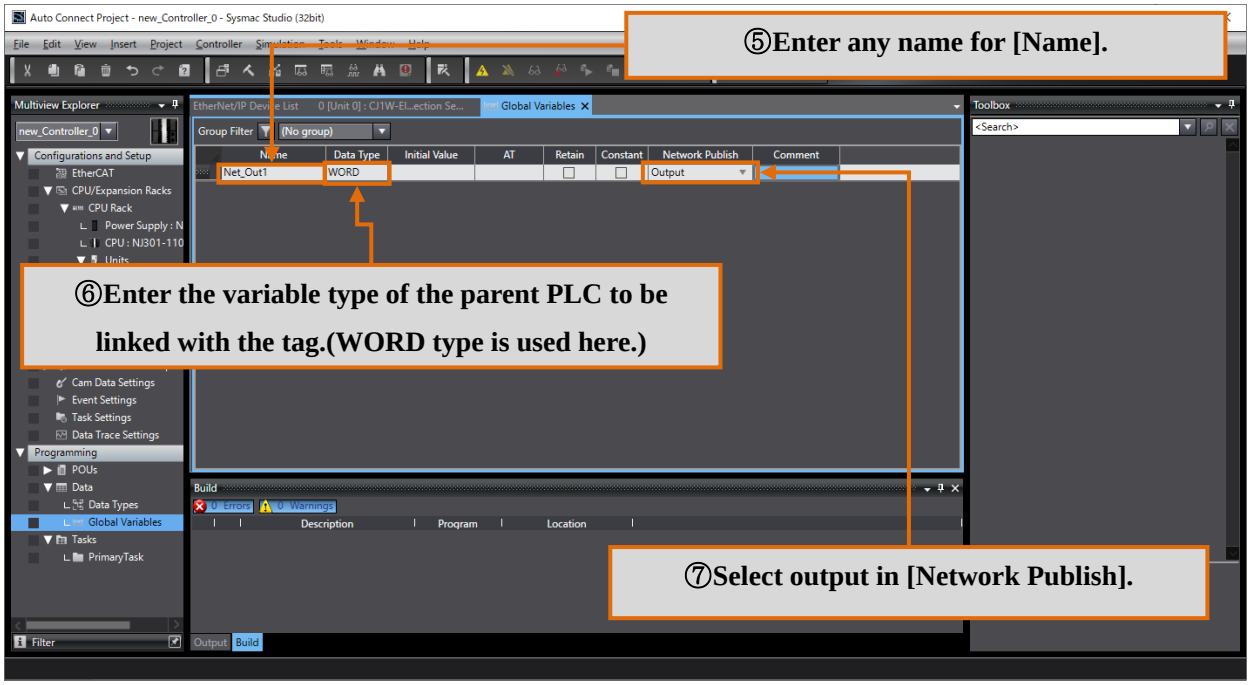
To edit the settings, click offline when it is online status, in the same way as described in "9.2.1 Configuration Settings ".9.2.1



Next, register a global variable.



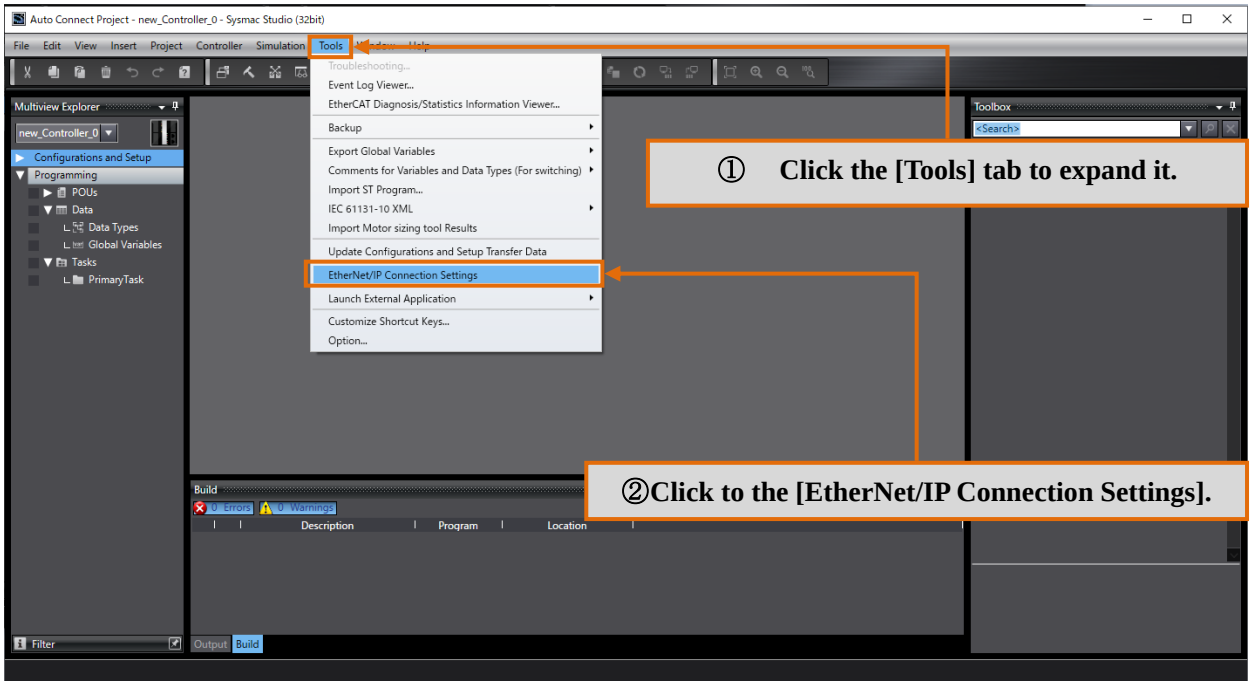
Register the global variable to be linked with the tag data and set it as follows. It is not necessary to set any items other than the following.

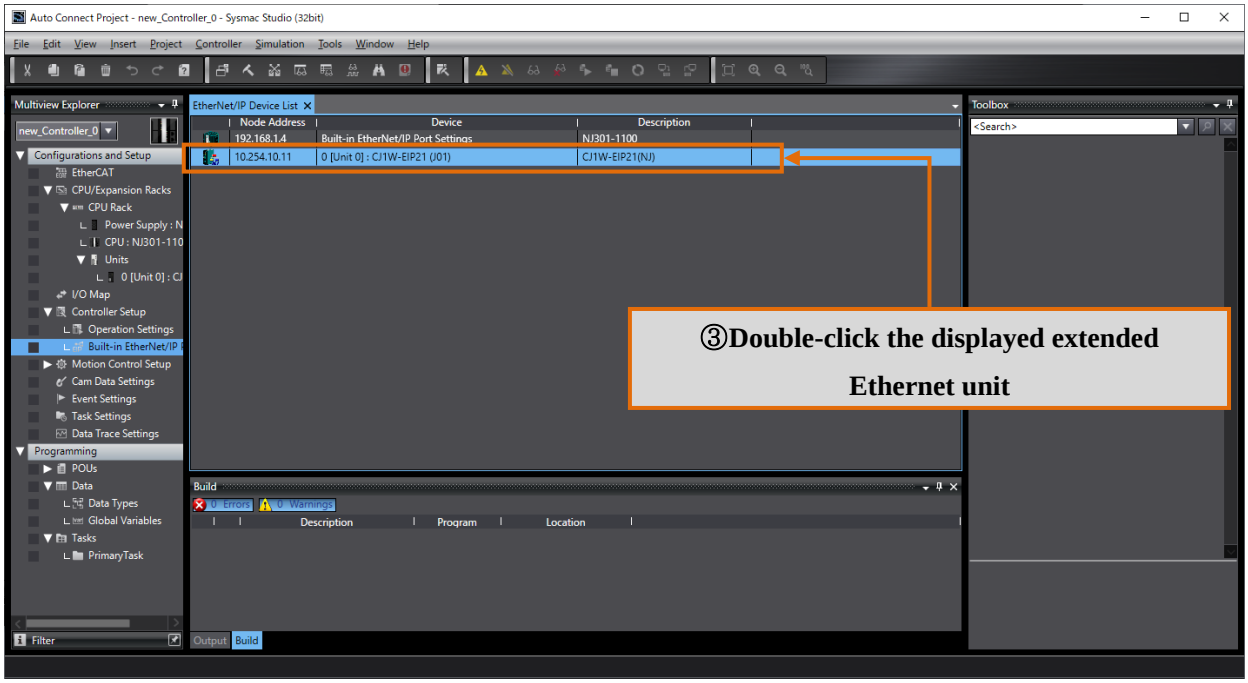


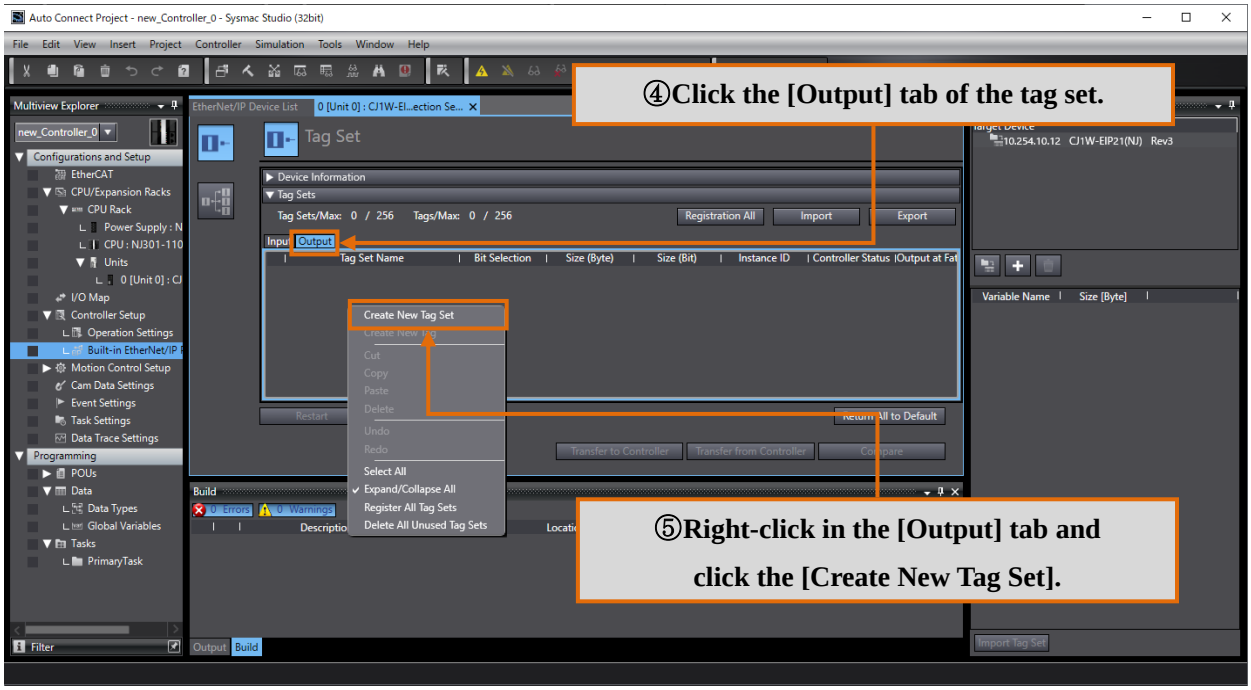
Variable registration is that all. Write the setting to the child PLC.9.2.1.3

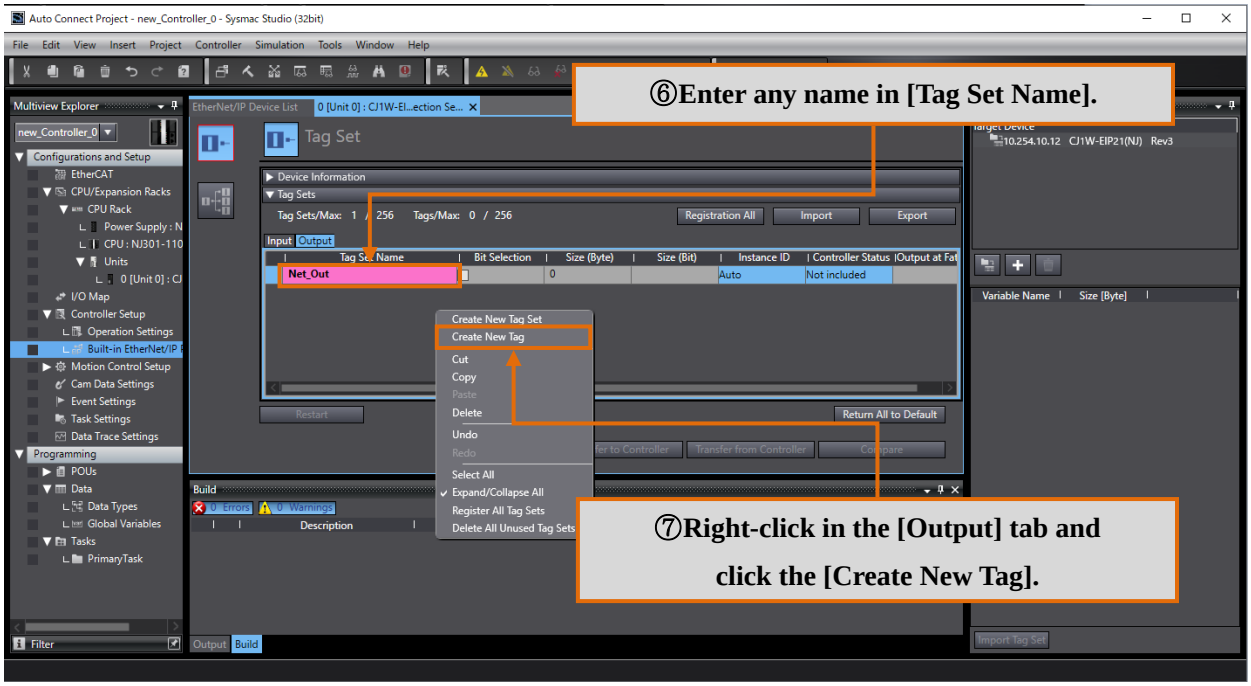
9.3.3. EtherNet/IP connection setting

Sets the tag set for tag data link.

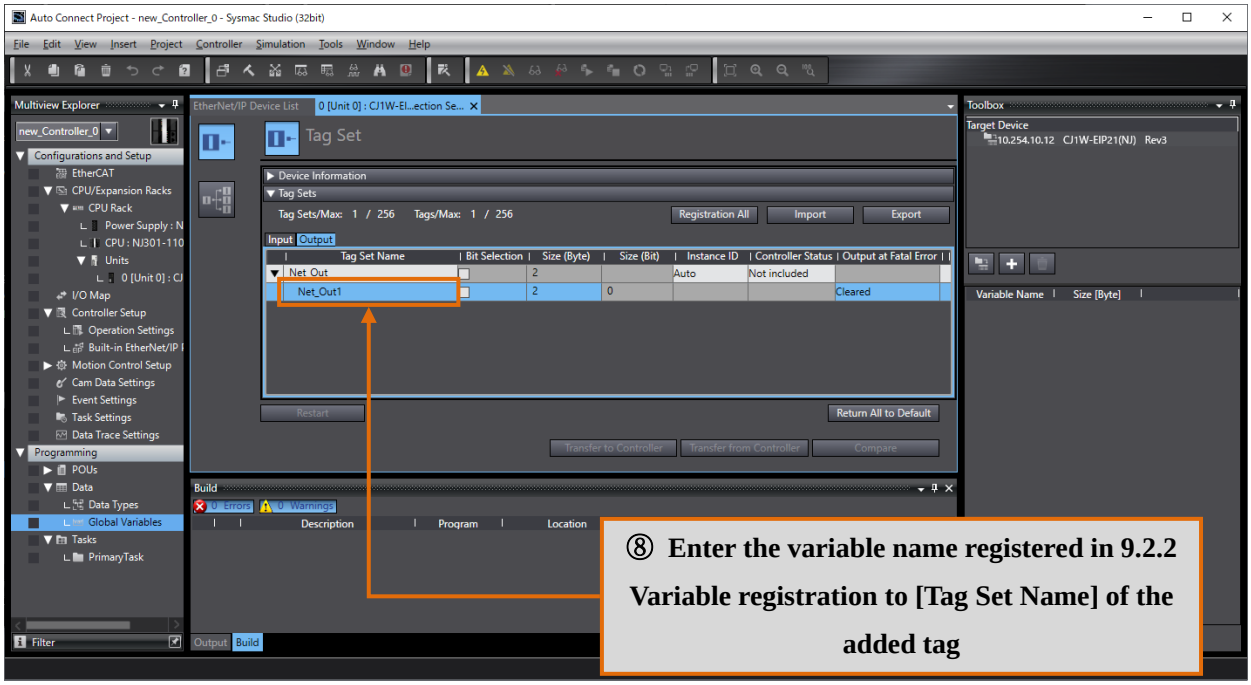




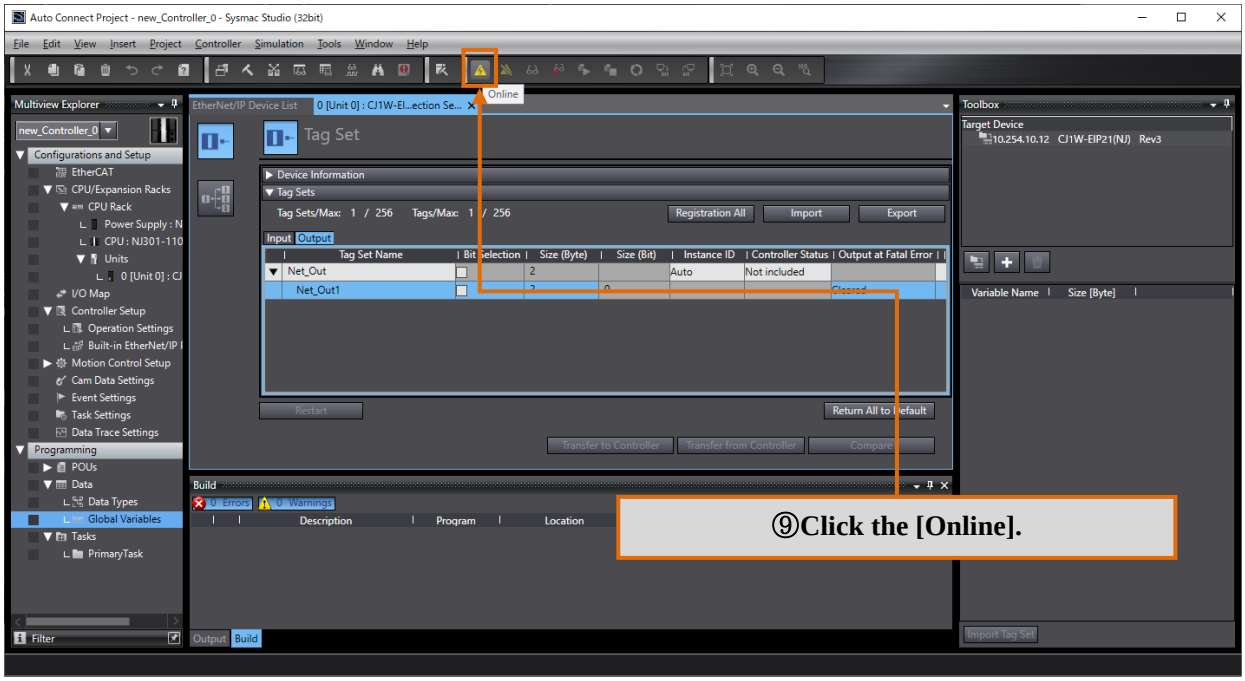

④Click the [Output] tab of the tag set.
⑤Right-click in the [Output] tab and click the [Create New Tag Set].
Create New Tag Set
Create New Tag
Cut
Copy
Paste
Delete
Undo
Redo
Select All
Expand/Collapse All
Register All Tag Sets
Delete All Unused Tag Sets

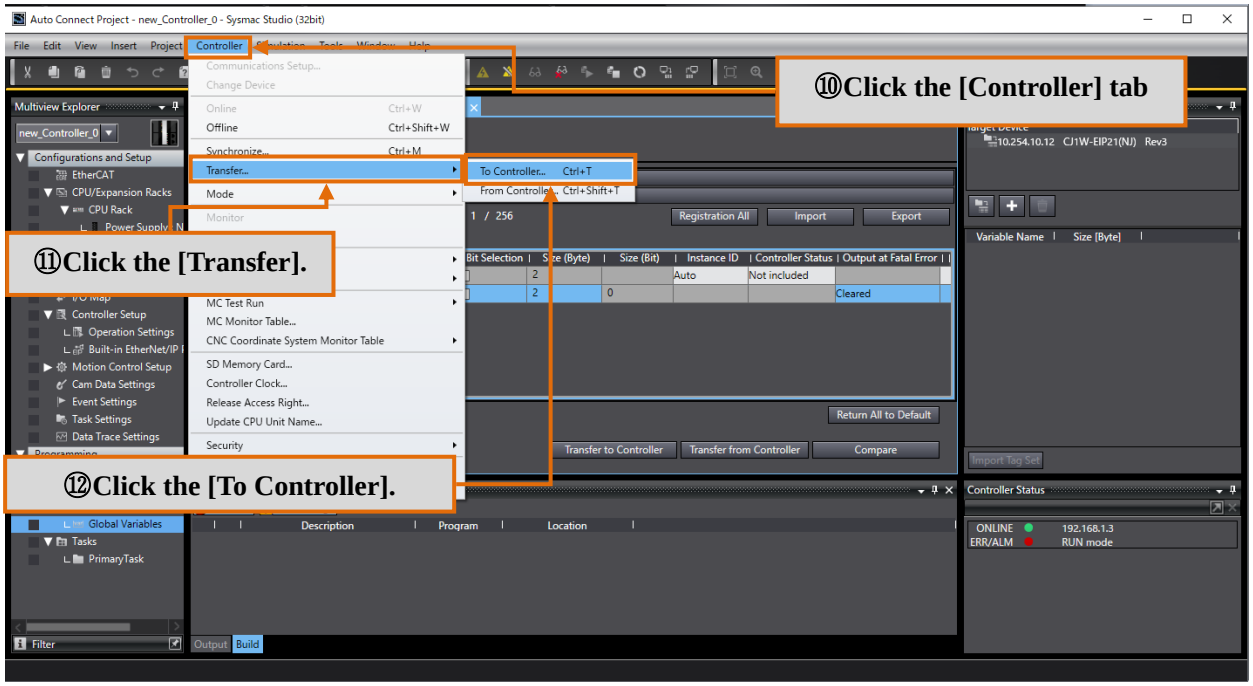

⑥Enter any name in [Tag Set Name].
⑦Right-click in the [Output] tab and click the [Create New Tag].
Net_Out
Create New Tag Set
Create New Tag
Cut
Copy
Paste
Delete
Undo
Redo
Select All
Expand/Collapse All
Register All Tag Sets
Delete All Unused Tag Sets

Sets the variable to which the tag data is linked to the tag set. Multiple variables can be registered in the tag set, but the same size, data type, and number of tags must be matched between the parent PLC and the child PLC.

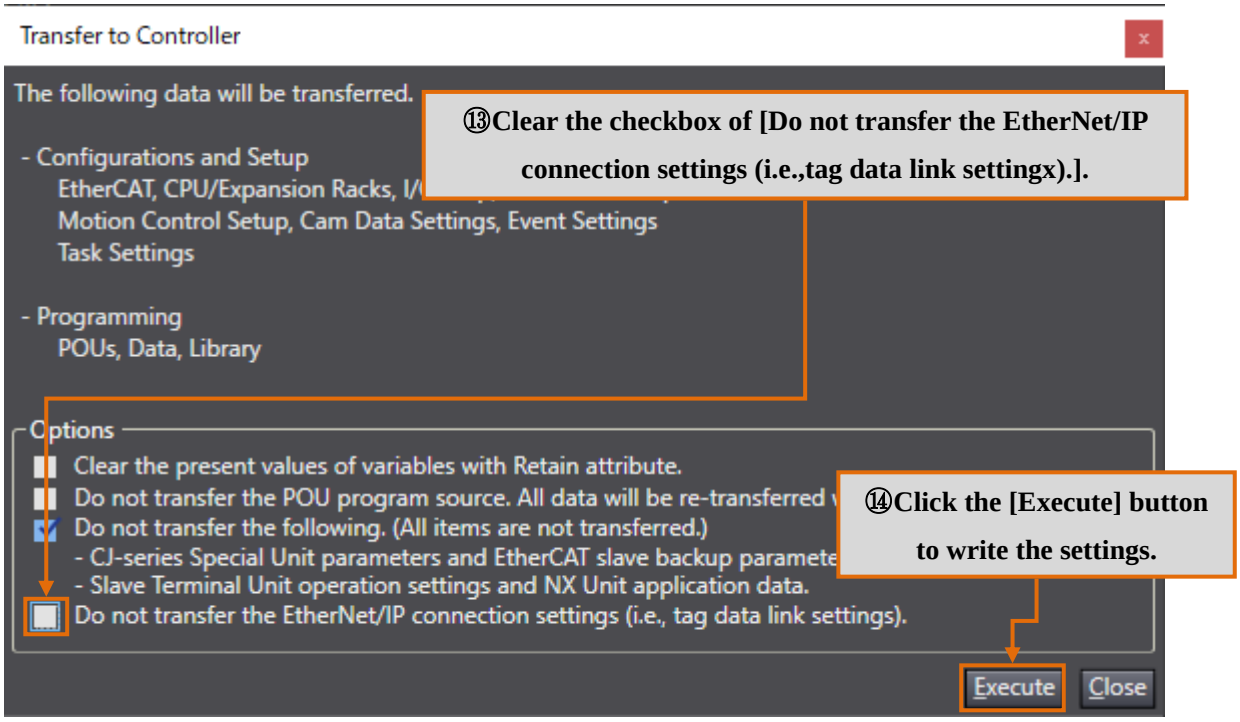


After configuring the tag set, write the settings to the child PLC.





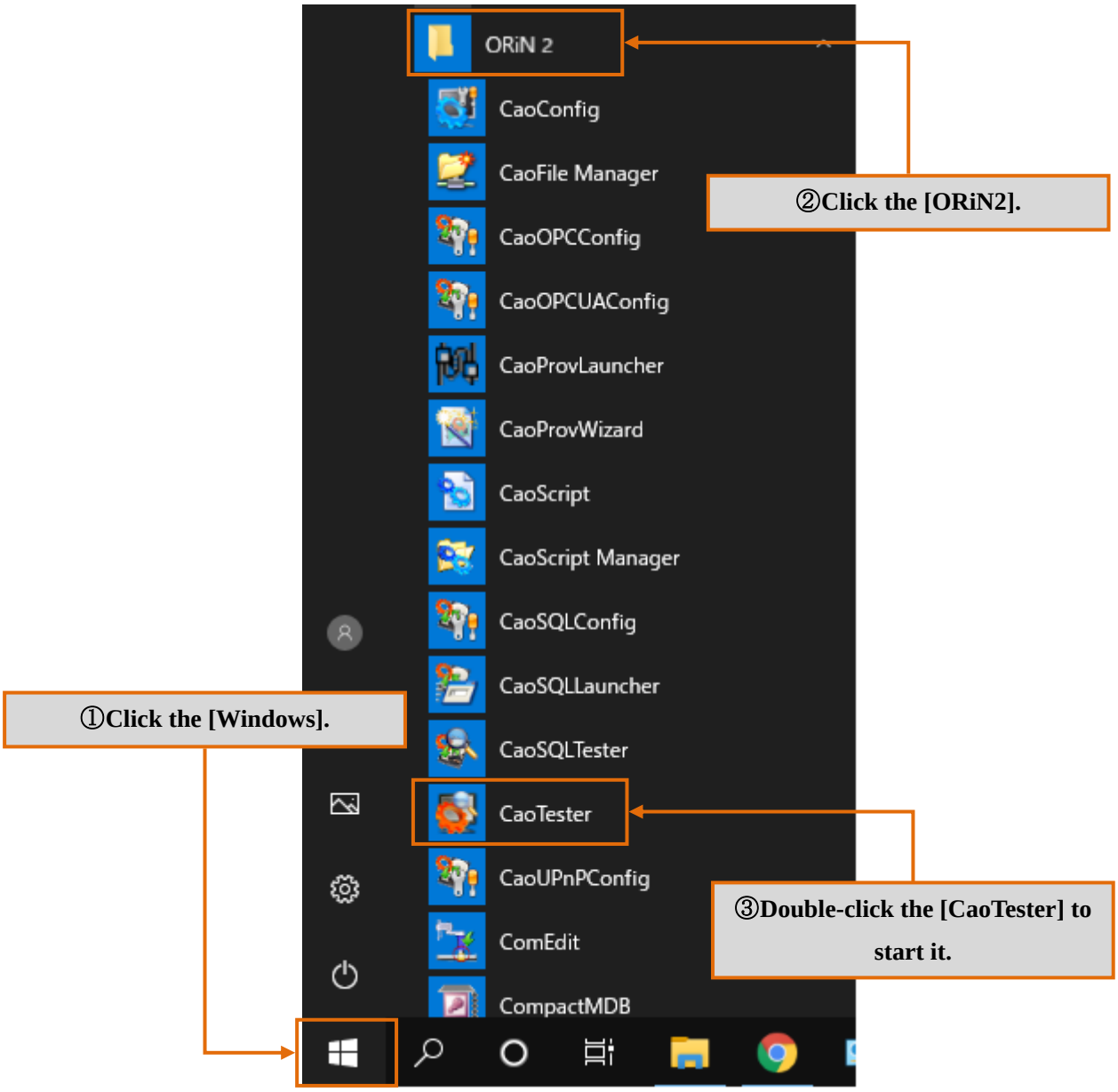
When transferring data from a PC, the following window will be displayed. Clear the appropriate checkbox and transfer the data.



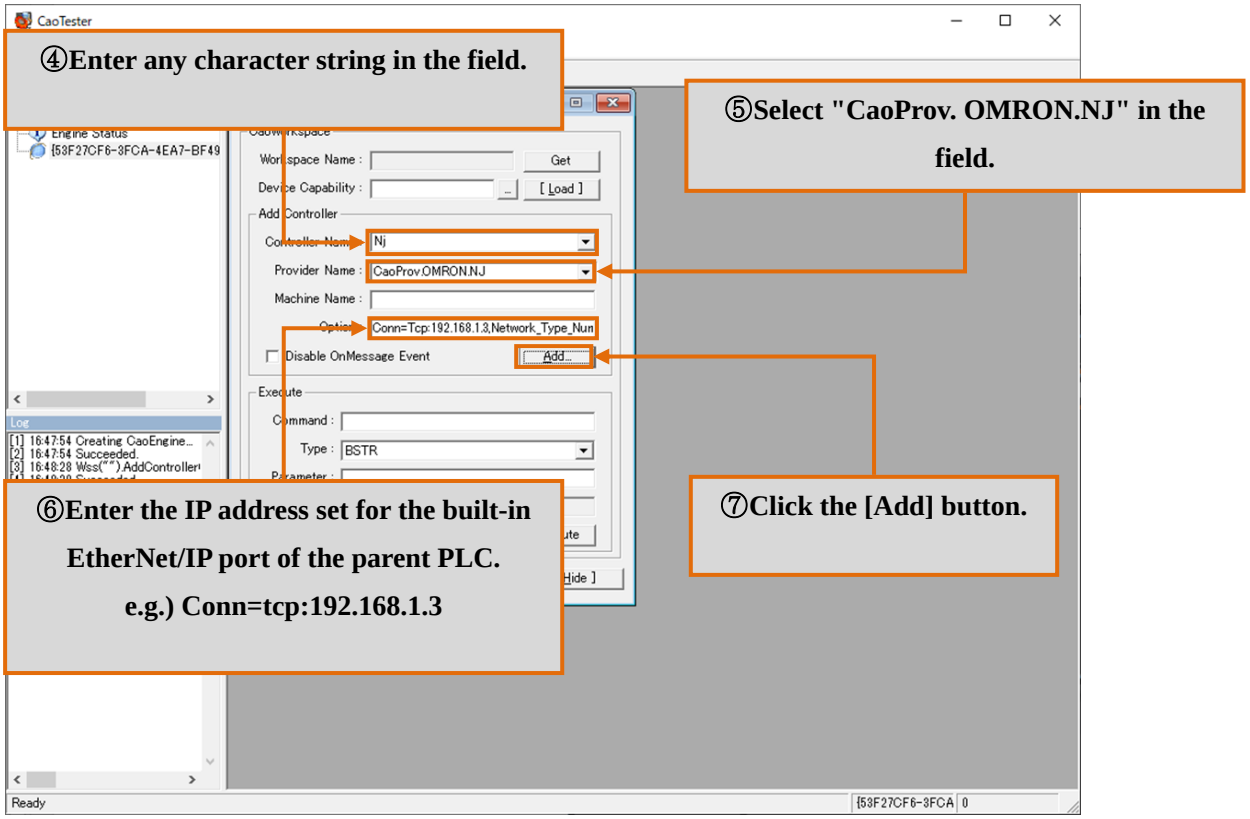
This completes the setting of the child PLC.

9.4. Get data by CaoTester

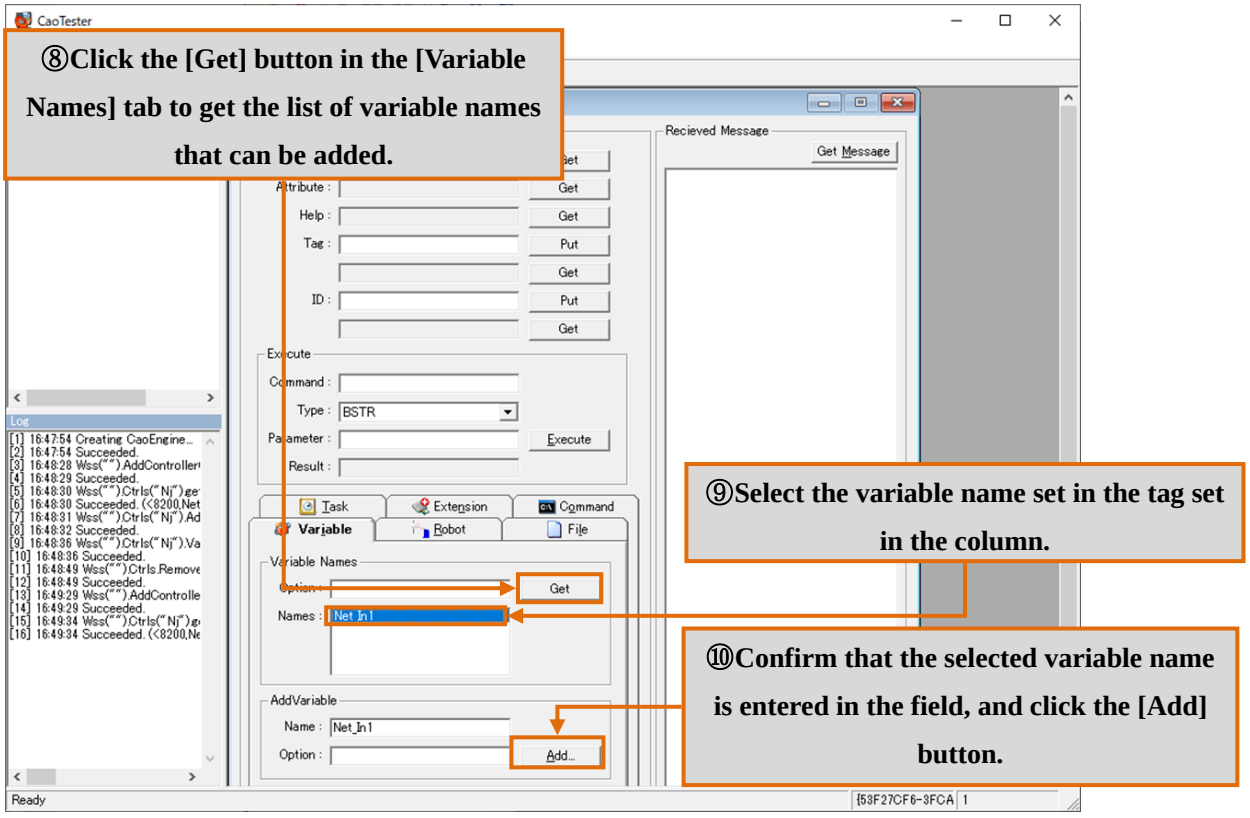
Show the how to get set tag data by CaoTester using OMRON.NJ series provider.9.29.3



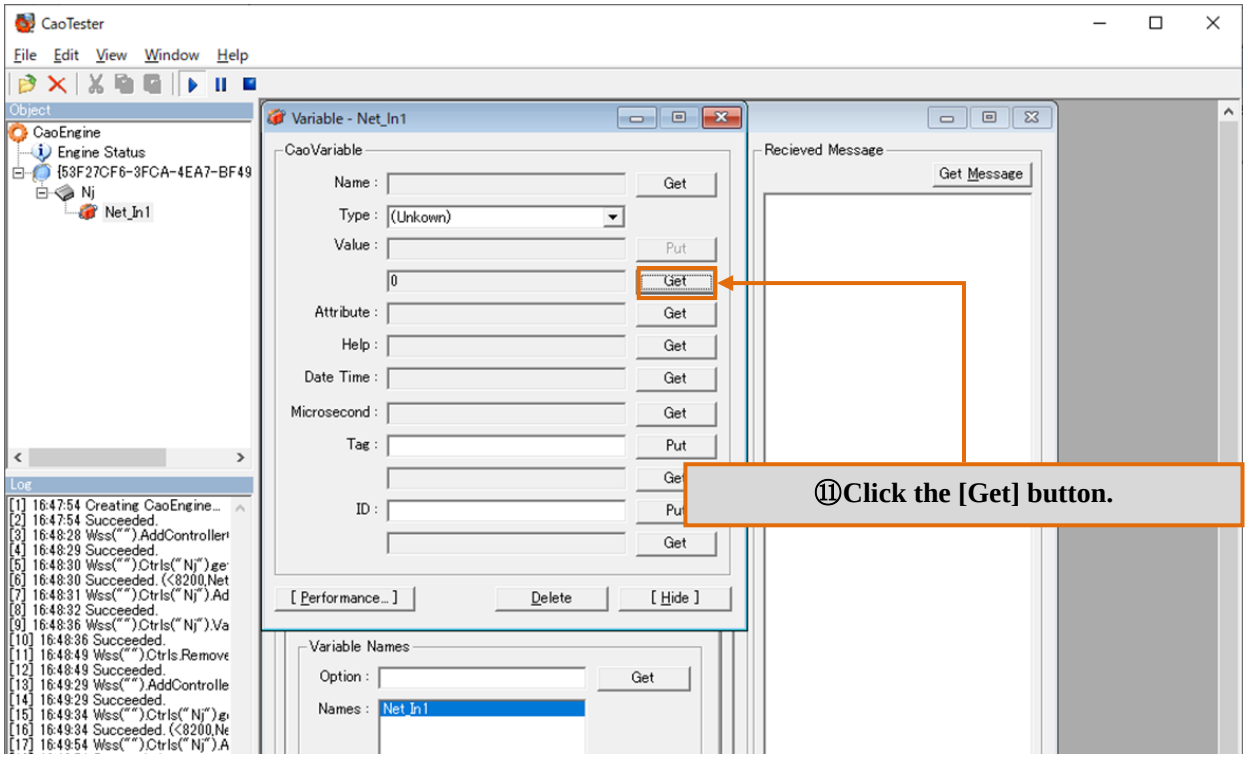
After starting CaoTester, perform AddController.



AddVariable is executed after AddController.



Finally, execute Get button to get value.



The procedure for accessing other stations using the tag data link is that all.