RockwellAutomation
Logix5000 provider

Version 1.0.7

User's guide

October 21, 2024

Remarks

This document is translated into English by machine translation.

[Revision history]

Version	Date	Content
1.0.0	31/10/2016	First edition.
1.0.1	15/10/2018	Bug fixed at AddController.
1.0.1	29/10/2018	I listed the Elem upper limit in AddController
1.0.2	04/02/2019	Addition for acquisition of batch of structure. Addition for array division sending and receiving.
	03/04/2019	Notes on BOOL-type arrays and examples of request paths are described.
1.0.3	27/09/2019	Revised internal processing (sequence count).
	23/10/2019	Fixed appendix A.
	28/11/2019	Indication that slot number must be specified for some CompactLogix models.
1.0.4	02/08/2022	Added consistency check for request and response packets.
1.0.5	30/01/2024	Changing ForwardOpen sequence-number assignment process.
1.0.6	21/10/2024	Fixed a problem that caused a lock state when a wire break.
1.0.7	21/10/2024	Modified to specify Vendor ID. Fixed to be able to set RedundantOwner ForwardOpen.

[Operation check model]

Model	Notes	Verified version of the provider
1769-L33ER (ver.21)	1769-L33ER CompactLogix™ 5370 Controller	1.0.6
1756-L71 (ver.21)	1756-L71 ControllLogix® 5570 Controller	1.0.6
1769-L35E	1769-L35E CompactLogix SlotNo option must be specified.	1.0.3
1756-L81E (ver.33.01)	1756-L81E ControlLogix® 5580 Controller (Use EtherNetIP unit of 1756-EN4TR, 1756-EN2TR)	1.0.7
1756-L75 (ver.33.01)	1756-L75 ControlLogix® 5570 Controller (Use EtherNetIP unit of 1756-EN4TR, 1756-EN2TR)	1.0.7

Contents

1. Introduction

This document is a user's guide of CAO provider that reads and writes data from/to a controller "Control Logix series" or "Compact Logix series" manufactured by Rockwell Automation.

CAO provider (CaoProvRockwellLogix5000.dll) used in this guide is called Logix5000 provider.

Chapter 2 describes the outline of the Logix5000 provider and the command reference.

Availability of communication commands and data columns implemented in Logix5000 provider depend on the controller (ControlLogix / CompactLogix series) to communicate.

For details about communication, refer to "1756-pm020c-en-p.pdf" of RockwellAutomation and Vol2_1.4.pdf of CIP specification of ODVA.

2. Outline of this provider

2.1. Outline

Logix 5000 provider is a CAO provider that reads and writes data from/to a RockwellAutomation's controller "Control Logix series" or "Compact Logix series" through Ethernet/IP (TCP) connection by using CIP format-based access commands. The file format is DLL (Dynamic Link Library) and it is dynamically uploaded from the CAO engine. To use Logix5000 provider, you need to install ORiN2SDK, or, complete the registration manually with the procedure shown below.

Table 2-1 Logix5000 provider

File name	CaoProvRockwellLogix5000.dll
ProgID	CaoProv. Rockwell. Logix5000
Registration	regsvr32 CaoProvRockwellLogix5000.dll
De-registration	regsvr32 /u CaoProvRockwellLogix5000.dll

2.2. Method and Properties

2.2.1. CaoWorkspace::AddController method

Logix5000 provider establishes communication by referring to the connection parameters for communication when AddController is executed. (This serves as a TCP client.)

Syntax AddController(<bstrCtrlName:BSTR>,<bstrProvName:BSTR>,
<bstrPCName:BSTR>,<bstrOption:BSTR>))

bstrCtrlName : [in] Controller name
bstrProvName : [in] Provider name. Fixed to "CaoProv. Rockwell. Logix5000"
bstrPCName : [in] Computer name where provider runs.
bstrOption: [in] Option character strings

The following shows a list of option character string items.

Table 2-2 Option strings of CaoWorkspace::AddController

Option ([1])	Description
Conn=<connection parameter>	Required. Communication configuration and connection parameter (Refer to 2.2.1.1)
MyIP[=<own IP address>]	Own IP address (Intended to several NICs) (Default : not specified)
ConnTimeout[=<Timeout period>]	Timeout period at TCP connection (millisecond) (Default: 3000)
Timeout[=<Timeout period>]	Communication timeout at the receiving and sending (millisecond) (Default: 3000 ms)
PathCheck[=TRUE / FALSE]	Specify if the request path is checked at AddVariable TRUE : Path is checked. FALSE : Path is not checked. (Default: FALSE)
SlotNo[=<Slot number>]	Slot number of the access target unit. (Default : not specified) Example 1)

[1] Items enclosed with square brackets ("[]") are omittable. Underlined part shows the default value when the option is not specified.

	CompactLogix ⇒Basically unnecessary[2] Example 2) CompactLogix(CPU=0) ⇒ SlotNo=0 Example 3) ControlLogix(CPU=0) ⇒ SlotNo=0
EnableRedundantOwner	Set the RedundantOwner setting in the ForwardOpen command of the Ethernet/IP CIP at the time of connection. TRUE: Redundant Owner FALSE: Non-Redundant Owner (Default: TRUE)

[2] The specification of SlotNo is basically unnecessary in CompactLogix, but it is necessary for some models. Check the manual of each model to see if you need SlotNo.

2.2.1.1. Conn option

The following shows connection parameter strings for Conn option.

Ethernet device

"Conn=ETH:<Dest IP Address>[:<Dest Port No>[:<Src IP Address>[:<Src Port No>]]]"

"Conn=TCP:<Dest IP Address>[:<Dest Port No>[:<Src IP Address>[:<Src Port No>]]]"

< Dest IP Address >	:	TCP/IP connection destination IP address Example : "127.0.0.1", "192.168.0.1"
<Dest Port No>	:	TCP/IP connection port number Example : 44818,5006,5007, ... Enter desired port number/
<Src IP Address>	:	Own IP address (Intended to several NICs) ([3])
<Src Port No>	:	Own port number (Intended to several NICs)

[3] If own IP address is specified from both Conn option and MyIP option, an error occurs. Be sure to specify own IP address in one option only.

2.2.2. CaoController::VariableNames property

Get a list of variable names that can be connected. The variable name obtained with this property can be used as the first argument of the AddVariable method described later.[4]

2.2.3. CaoController::AddVariable method

AddVariable method of CaoController class is a method to create variable objects. These variable objects are used to read/write data from/to the tag definition of the controller (ControlLogix/CompactLogix series).

This method determines the request path, data type, and the number of data elements to be accessed by entering options.

When it accesses the variable first time making it (Reading, write it), the size of the type is acquired.
It acquires it at AddVariable when PathCheck=True is set.

Syntax AddVariable(<bstrVariableName:VT_BSTR>[,<bstrOption:VT_BSTR>])

<bstrVariableName> : [In] Variable name

<bstrOption> : [in] Option character strings

The following shows a list of option character string items.

Table 2-3 Option character strings of CaoController::AddVariable

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>Path=<request path></td><td>Required.
Specify the request path of data to be accessed.
(Reference 2.2.3.1)</td></tr>
<tr><td>DataFormat[=<data type>]</td><td>Specify the data type of the end of the request path (data type of the actual access destination) specified by Path option. Decimal
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Numeric type, numeric type array</td></tr>
<tr><td>1</td><td>Character string type, character string type array</td></tr>
<tr><td>2</td><td>Structure type and structure type array</td></tr>
</table>
(Default: 0)</td></tr>
</table>

[4] To get a list of tags defined on the target device, the firmware version must be 21 or later

	(Reference 2.2.3.2)
Elem[=<number of element>]	Specify the number of data elements to put/get. Decimal This option is available only when the data type of the end of the request path specified by Path option is array. (Reference 2.2.3.4) Please set a number of element in less than 504 Byte. (Default: 1)

2.2.3.1. Path option

Specify the request path of data to be accessed with character string type.

To access data that locates deeper layer, such as a structured member, specify a request path with entering a dot between paths to connect them. If you specify a path as array, add an element number enclosed by square brackets in the end of path. (Example: [1]) For details about how to specify, refer toAppendix A[5].

Please note that the data type of the end of request path must be numeric type, character string type, numeric type array, or character string type array. If you specify any structure data definition or any array definition without elements number specification as a request path, and then perform reading/writing operation, it may lead to unexpected result.

2.2.3.2. DataFormat option

Specify the data type of the end of the request path (data type of the actual access destination) specified by Path option.

VARIANT data types handled by provider and data types inside of controller are interconnected as the following table shows.

Table 2 - 4 Data type corresponding table

Data type of VARIANT([6])	Data type inside of controller	Size	Description
VT_BOOL	BOOL	1Byte	Boolean (TRUE/FALSE)
VT_I1	SINT	1Byte	1-byte integer (signed)
VT_I2	INT	2Byte	2-byte integer (signed)

[5] Access to BOOL-type arrays is done on a per DWORD basis. When the element number is 0, access is made to elements 0 to 31, and when the element number is 1, access is made to elements 32 to 63.

6 When DataFormat = 2, only (VT_ARRAY | VT_UI1) is specifiedWhen DataFormat=0,1, (VT_ARRAY | VT_UI1) cannot be specified.

VT_I4	DINT		4Byte	4-byte integer (signed)
VT_R4	REAL		4Byte	Single precision real number
VT_UI4	DWORD		4Byte	4-byte integer (unsigned)
VT_I8	LINT		8Byte	8-byte integer (signed)
VT_BSTR	STRING	LEN	4Byte	String length (DINT type)
		DATA[n]	1 byte X n	ASCII array of character data (SINT[n] type)
VT_ARRAY | VT_UI1	All Type		1Byte×n	BYTE ARRAY

2.2.3.3. Request path availability check

If TRUE is selected for PathCheck of AddController option, the availability of a path, which is specified by Path option, is checked at the variable creation.

The path availability is checked by internally executing get_Value-equivalent operation that checks if the result value is successfully read without any errors.

2.2.3.4. Elem option

When the tail is only array specification after passing of the request, it is effective according to optional Path. The number of arrays that can be sent and received depends on the tag name length though the data size treated by the variable at a time is about 500 bytes. The number of arrays that can be sent and received by shortening the tag name length increases. The array is sent and received by dividing into the unit installed on one packet.

Please refer to Appendix C for how to request the number of arrays that can be sent and received at a time and the array size that can be sent and received at a time.

2.2.4. CaoVariable::put_Value property

Based on the specified option, this property sends a packet that incorporates CIP format-based SendRRData encapsulating WriteTagService(0x4d), and then writes it to the tag definition's value in the controller.

For arrays, you can write several elements at the same time by specifying the writing start element number with Path option and specifying the number of elements to write with Elem option.

Unlike the access to numeric type data, when character string type data (STRING) is accessed, writing packet is internally sent several times. For details, refer to Appendix B.

If you specify any other data type written in Table 2 - 4 as a data type of writing value, an argument error will occur. Also when values are written in array if the number of writing elements specified by Elem option and the number of elements to write are different, an argument error occurs.

2.2.5. CaoVariable::get_Value property

Based on the specified option, this property sends a packet that incorporates CIP format-based SendRRData encapsulating ReadTagService(0x4c), and then reads the tag definition's value from the controller.

For arrays, you can read several elements at the same time by specifying the writing start element number with Path option and specifying the number of elements to write with Elem option.

Unlike the access to numeric type data, when character string type data (STRING) is accessed, reading packet is internally sent several times. For details, refer to Appendix B.

2.3. Variable list

2.3.1. CaoController class

Table 2-5 CaoController class system variable list

Variable name	Data type	Description	Attribution	
			get	put
@MAKER_NAME	VT_BSTR	RockwellAutomation (manufacturer's name) is returned.	✓	―
@VERSION	VT_BSTR	Provider version information	✓	―
@ERROR_STATUS	VT_I4 | VT_ARRAY	Detail information of error response that occurred immediate before. ([7]) [0] : Error Code [1] : Extended Error	✓	―
@WROTE_STRING	VT_I4	Number of elements that has been successfully written at the writing of character string type array.	✓	―

Table 2-6 CaoController class user variable list

Variable name	Data type	Description	Attribution	
			get	put
Any name	Depends on a target variable	Access the tag definition in the controller (ControlLogix / CompactLogix series).	✓	✓

[7] For information about error contents, refer to "Read Tag Service Error Codes" and "Write Tag Service Error Codes" in "1756-pm020c-en-p (Data Access).pdf" from RockwellAutomation.

2.4. Error code

The following original error codes are defined in Logic5000 provider.

For about ORiN2 common errors, refer to the chapter of Error code in ORiN2 programming guide.

Table 2-7 Original error code

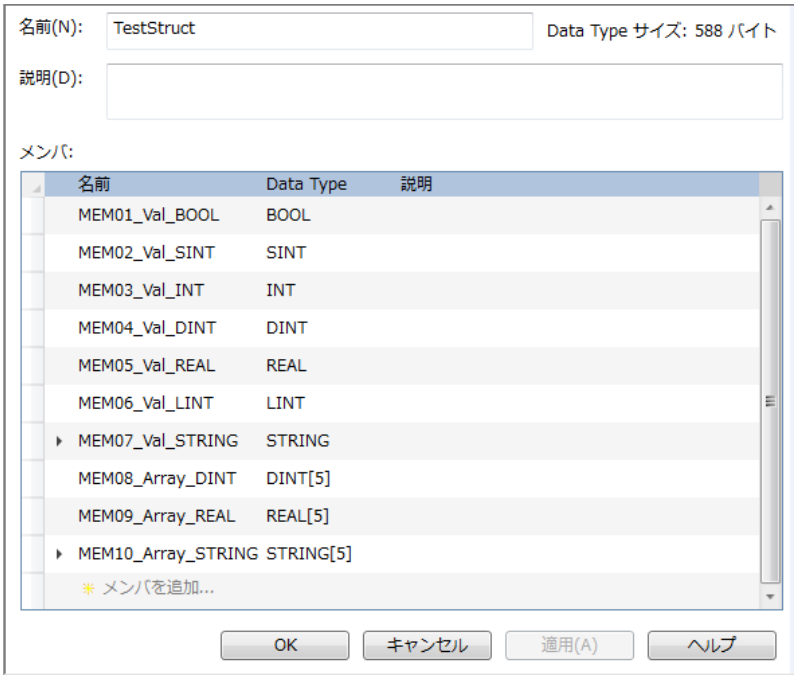
Error name	Error number	Description
CIP response error (Format error)	0x80100000	This error occurs when a response packet of CIP communication is unexpected abnormal format.
CIP response error (Error Status Code)	0x80100001	This error occurs when a response packet of CIP communication shows abnormal state.
CIP response error (Error General Status)	0x80100002	This error occurs when a response packet of CIP communication shows an error.[8] This error occurs if any of request path, data type, or number of access elements at the request is incorrect. When this error occurs, you can obtain the detailed error code by reading @ERROR_STATUS of system variable. Also, if this error occurs at the writing of character string type array, you can identify which element number's writing generated the error by reading @WROTE_STRING of system variable.
RequestPath error	0x80100003	This error occurs when passing specified when structure information is acquired is not found.
The size of the structure is too large.	0x80100004	This error occurs when the size of the structure is too large. Please reduce the size of the structure.
CIP response packet mismatch	0x80100005	This message is returned when CIP communication response packet has occurred a certain number of times and is not the expected packet.

[8] This error also occurs for models that require SlotNo (including CompactLogix). If this error occurs even though the settings seem to be correct, try specifying SlotNo. Check the manual of each model to see if you need SlotNo.

Appendix A. Specifying a request path

This chapter shows how to specify a request path to access each data when the following definitions are adopted in the controller.

[User-defined structure defined by LogixDesigner]



[Controller tag defined by LogixDesigner]

名前	値	強制マスク	スタイル	Data Type
Val1	123		Decimal	DINT
Val2	'Test'	{...}		STRING
Val2.LEN	4		Decimal	DINT
Val2.DATA	{...}	{...}	ASCII	SINT[82]
Val3	{...}	{...}	Float	REAL[3]
Val3[0]	1.234		Float	REAL
Val3[1]	0.555		Float	REAL
Val3[2]	0.0		Float	REAL
Val4	0		Decimal	BOOL
Val5	{...}	{...}	Decimal	BOOL[160]
Val6	{...}	{...}		STRING[3]
Val6[0]	'Str1'	{...}		STRING
Val6[1]	'Str2'	{...}		STRING
Val6[2]	'Str3'	{...}		STRING
Val7	{...}	{...}		TestStruct
Val8	{...}	{...}		TestStruct[2]

Example 1) Numeric type definition (Val1)

Path=Val1, DataFormat=0

Example 2) Character string type definition (Val2)

Path=Val2, DataFormat=1

Example 3) Numeric type array (Element 0 to 2 of Val3)

Path=(Val3[0]), DataFormat=0, Elem=3

Example 4) BOOL type (Val4)

Path=(Val4), DataFormat=0

Example 5) BOOLtype array (Element 32 to 95 of Val5)

Path=(Val5[1]) , DataFormat=0, Elem=2

Example 6) Character string type array (Element 1 to 2 of Val6)

Path=(Val6[1]), DataFormat=1, Elem=2

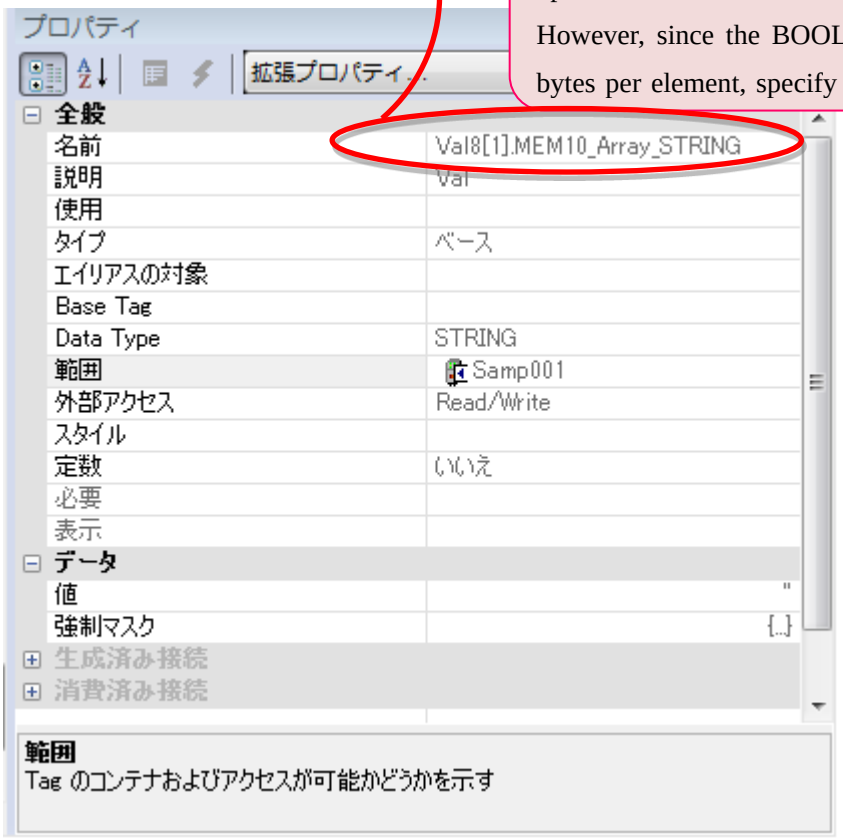
Example 7) Structure member (MEM03_Val_INT of Val5)

Path=Val5.MEM03_Val_INT, DataFormat=0

Example 8) Example where an array is included in the middle of a path. (Element 0 to 2 of MEM10_Array_STRING of Val8[1])

Path=(Val8[1].MEM10_Array_STRING[0]), DataFormat=1, Elem=3

Enter a Tag name displayed on LogixDesigner in the Path option as-is.
However, since the BOOL type array is accessed with 32 bytes per element, specify the last element number divided



Example 9) structure (Val7)

Path=Val7, DataFormat=2

Example 10) structure Array(Element 0 to 1 of Val8)

Path=(Val8[0]), DataFormat=2, Elem=2

Example 11) Numeric type (Val1)

Path=Val1, DataFormat=2

Example 12) String type(Val2)

Path=Val2, DataFormat=2

Example 13) structure member (MEM03_Val_INT of Val7)

Path=Val7.MEM03_Val_INT, DataFormat=2

Example 14) It patterns including the array while passing. (Val8[1] MEM10_Array_STRING 0～2)

Path=(Val8[1].MEM10_Array_STRING[0]) , DataFormat=2, Elem=3

Appendix B. Access to the character string type

In LogixDesigner, character string type (STRING) is defined as a structure type that comprises two members; XXX.LEN and XXX.DATA[n].

LEN stores character string length (byte number) in DINT type and DATA stores character string data converted to ASCII in SINT type array. (Areas where DATA character is not stored are initialized with NULL = 0x00.)

String	'Test'	{...}		STRING
String.LEN	4		Decimal	DINT
String.DATA	{...}	{...}	ASCII	SINT[82]
String.DATA[0]	'T'		ASCII	SINT
String.DATA[1]	'e'		ASCII	SINT
String.DATA[2]	's'		ASCII	SINT
String.DATA[3]	't'		ASCII	SINT
String.DATA[4]	'$00'		ASCII	SINT
String.DATA[5]	'$00'		ASCII	SINT
String.DATA[6]	'$00'		ASCII	SINT

To write values with put_Value, create a WriteTagService packet that corresponds to "Path=(String.DATA[0]), DataFormat=0, Elem= number of character string to write", set character strings specified in VT_BSTR as ASCII type data and then send it. Once DATA part is written successfully, create a WriteTagService packet that corresponds to "Path=String.LEN, DataFormat=0", set the character string length, and then send it. One WriteTagService packet is sent in two transmissions. LEN is sent first, and then DATA is sent next.

To read values with get_Value, create a ReadTagService packet that corresponds to "Path=String.LEN, DataFormat=0" and send it in order to obtain the character string length. Once LEN is successfully read, create a ReadTagService packet that corresponds to "Path=(String.DATA[0]), DataFormat=0, Elem=character string length" and send it. Obtained data is connected and returned in VT_BSTR. One ReadTagService packet is sent in two transmissions. LEN is sent first, and then DATA is sent next.

For both the data writing and reading, to access character string type array, you need to repeat the process above by the number of elements to access. (Therefore, the data synchronism is not guaranteed even if you access several array elements at the same time.)

Appendix C. About the array size that can be sent and received at a time

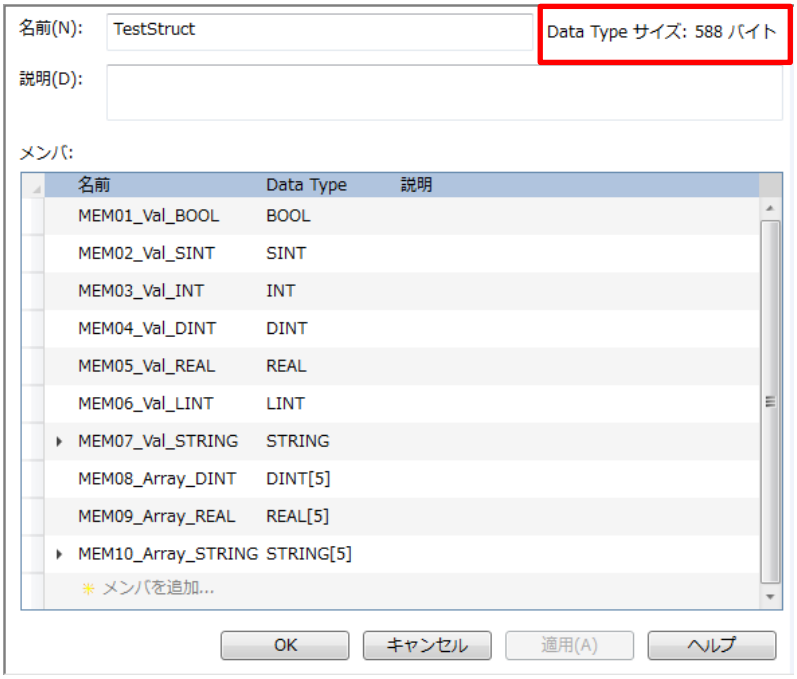
Sending and receiving is done by the variable according to one with small possible number of elements the number of elements that can be sent and received at a time.

To become below these units of the element number, the packet is divided and transmitted.

It becomes an error when the size of the structure is included for the structure and it doesn't install on one packet or less. The character string type becomes a treatment similar to the structure, too.

The data size of the structure can be confirmed from LogixDesigner.

[User definition structure defined with LogixDesigner]



The size that can be sent and received becomes a variable size because it becomes a fixed-size, and it writes and it uses the tag name because it doesn't use the tag name at the time of reading.
It becomes as shown in the following tables.

Data Type	Size	Number of maximum reading elements[9]	Number of maximum writing elements[9]
BOOL	4 Byte	123	(504 - Request passing size) / 4
SINT	1 Byte	492	(504 - Request passing size) / 1
INT	2 Byte	246	(504 - Request passing size) / 2
DINT	4 Byte	123	(504 - Request passing size) / 4
REAL	4 Byte	123	(504 - Request passing size) / 4
DWORD	4 Byte	123	(504 - Request passing size) / 4
LINT	8 Byte	61	(504 - Request passing size) / 8
STRING[10]	1 Byte[11]	492	(504 - Request passing size[12]) / 1
Structure[10]	x Byte[13]	492 / Size of structure	(502 - Request passing size) / Size of structure

[9] The table is Compact Logix. The Control Logix is 10 bytes smaller than the Compact Logix.
[10] When DataFormat=2 is specified, STRING becomes a treatment of the structure (LEN (four bytes) and DATA (n byte)).
Please note that it is not very related and acquires all data about effective character string length (LEN) unlike DataFormat=1.
[11] Please note that not the number of elements but the number of definition characters becomes a size as for the STRING type.
When the number of maximum characters is set to 100, the size becomes 100 bytes.
[12] The STRING type is a tag name. The request passing size is length [kunarimasu] of ten bytes unlike other types because of the access to Data n.
[13] Padding is done so that the structure may become four bytes by each member variable.
The size of the structure is eight bytes (INT (two bytes)+ Padding (two bytes)+ DINT (four bytes)) for the structure with two members of the INT type element and the DINT type.

[Method of calculating readable size]

・Data size that can be received

Non-structure:492 bytes

Structure and STRING type:492 bytes

・Number of elements that can be received

Quotient into which data size that can be received is divided by type size

example) DINT type

Element the number of = data size (492) that can be received of can reception type/size (4) = 123

[Method of calculating recordable size]

・Data size that can be transmitted

Non-structure:504 bytes

Structure and STRING:502 byte

・Request passing size

It ..request passing.. resolves it ..each dot (".").. (The resolved element is assumed to be a request passing element).

Request passing element:

[Non-array part]

Request header (2) + element character string length + even number padding (1)

[Array part]

Request header (2) + element character string length + even number padding (1) + array specification size (4)

Request passing size:

Request passing size = request passing element 0 + request passing element 1 +…+ request passing element n

example) DINT type

Path=Val5.MEM03_Val_DIN[1], DataFormat=0, Elem=2

Elements of data 0: Six bytes

Request header (2) + element character string length (4) + even number padding (0)

Elements of data 1: 20 bytes

Request header (2) + element character string length (13) + even number padding (1) + array specification size (4)

Element the number of = data size (502) that can be transmitted of can transmission ? request passing size (26) type/size (4) = 119