b-CAP server & b-CAP client

User's guide

Version 1.0.0

January 30, 2015

[Remark]

[Revision history]

Date	Version	Content
2015-1-30	1.0.0	First edition

Contents

1. Introduction

This is a user's guide designed for clients who use b-CAP server and b-CAP client for system development.

2. b-CAP server

b-CAP server activates a b-CAP receive thread and calls back a function according to the packet received.

2.1. Function specifications

2.1.1. bCap_Open_Server

Syntax

HRESULT bCap_Open_Server(const char *connect, uint32_t timeout, int *pfd);

Description

Activate a receive thread with s specified communication setting.

Argument

[in]	connect	Communication setting
[in]	timeout	Receive timeout period [ms]
[out]	pfd	File descriptor of activated receive thread

Connection

For Ethernet communication(TCP)

tcp[:DestIP[:DestPort[:SourceIP[:SourcePort]]]]

DestIP	Client IP address	(not used)
DestPort	Client port number	(not used)
SourceIP	Server IP address	(Default :255.255.255.255)
SourcePort	Server port number	(Default :5007)

For Ethernet communication (UDP)

udp[:DestIP[:DestPort[:SourceIP[:SourcePort]]]]

※Meanings of each parameter are same as that of Ethernet communication (TCP).

For serial communication

com[:<COM Port>[:<BaudRate>[:<Parity>:<DataBits>:<StopBits>[:<Flow>]]]]

COM Port	COM port number	(Default :1)
BaudRate	Baud rate	(Default :38400)
Parity	Parity bit	(Default :N)
DataBits	Data bit	(Default :8)
StopBits	Stop bit	(Default :1)
Flow	Flow control	(Default :0)

Return value

S_OK(0) is returned when a receive thread is successfully activated. If it fails, a negative value is returned.

Note

None

2.1.2. bCap_Close_Server

Syntax

HRESULT bCap_Close_Server(int *pfd);

Description

Close a receive thread of specified file descriptor.

Argument

[in,out] pfd File descriptor of receive thread to close

Return value

S_OK(0) is returned if a receive thread is successfully closed. If not, a negative value is returned.

Note

None

2.1.3. bCap_SetCallFunc

Syntax

HRESULT bCap_SetCallFunc(int32_t id, CALL_FUNC_BCAP func);

Description

Specify a function that is called back when b-CAP packet is received.

Argument

[in] id b-CAP function ID

[in] func Callback function

callback function

typedef HRESULT (*CALL_FUNC_BCAP)(VARIANT *vntArgs, int16_t Argc, VARIANT *vntRet);

[in] vntArgs Argument array

[in] Argc Number of elements of argument array

[out] vntRet Execution result

Return value

S_OK(0) is returned if a function is successfully specified. If not, a negative value is returned.

Note

For details about arguments of callback functions, refer to 4.Appendix A.b-CAP function ID and

Callback function"

3. b-CAP client

b-CAP client establishes connection with a specified b-CAP server in order to provide an interface that sends a given b-CAP function.

3.1. Function specifications

3.1.1. bCap_Open_Client

Syntax

HRESULT bCap_Open_Client(const char *connect, uint32_t timeout, unsigned int retry, int *pfd);

Description

Establish communication with a b-CAP server that has a specified communication setting.

Argument

[in]	connect	Communication setting
[in]	timeout	Receive timeout period [ms]
[in]	retry	Send retry count
[out]	pfd	File descriptor of socket whose communication has been established

Connect

For Ethernet communication(TCP)

tcp[:DestIP[:DestPort[:SourceIP[:SourcePort]]]]

DestIP	Server IP address	(Default :127.0.0.1)
DestPort	Server port number	(Default :5007)
SourceIP	Client IP address	(not used)
SourcePort	Client port number	(not used)

For Ethernet communication (UDP)

udp[:DestIP[:DestPort[:SourceIP[:SourcePort]]]]

※Meanings of each parameter are same as that of Ethernet communication (TCP).

For serial communication

com[:<COM Port>[:<BaudRate>[:<Parity>:<DataBits>:<StopBits>[:<Flow>]]]]

COM Port	COM port number	(Default :1)
BaudRate	Baud rate	(Default :38400)
Parity	Parity bit	(Default :N)
DataBits	Data bit	(Default :8)
StopBits	Stop bit	(Default :1)
Flow	Flow control	(Default :0)

Return value

S_OK(0) if communication is successfully established. If not, a negative value is returned.

Note

None

3.1.2. bCap_Close_Client

Syntax

HRESULT bCap_Close_Client(int *pfd);

Description

Close communication with a specified file descriptor.

Argument

[in,out] pfd File descriptor of communication to closed

Return value

S_OK(0) is returned if communication is closed successfully. If not, a negative value is returned.

Note

None

3.1.3. bCap_SetTimeout

Syntax

HRESULT bCap_SetTimeout(int fd, uint32_t timeout);

Description

Set a receive timeout period of specified file descriptor.

Argument

[in] fd File descriptor of communication whose receive timeout period is set.

[in] timeout Receive timeout period[ms]

Return value

S_OK(0) is returned if a receive timeout period is successfully set. If not, a negative value is returned.

Note

None

3.1.4. bCap_GetTimeout

Syntax

HRESULT bCap_GetTimeout(int fd, uint32_t *timeout);

Description

Obtain a receive timeout period of specified file descriptor.

Argument

[in] fd File descriptor of communication which receive timeout period is obtained.

[out] timeout Receive timeout period[ms]

Return value

S_OK(0) is returned if a receive timeout period is successfully obtained. If not, a negative value is returned.

Note

None

3.1.5. bCap_SetRetry

Syntax

HRESULT bCap_SetRetry(int fd, unsigned int retry);

Description

Set a send retry count of specified file descriptor.

Argument

[in] fd File descriptor of communication whose send retry count is set.

[in] retry Send retry count

Return value

S_OK(0) is returned if a send retry count is successfully set. If not, a negative value is returned.

Note

・Valid range is from 1 to 7.

・Any values less than 1 will be treated as 1, values over 7 will be treated as 7.

・This will be treated 1 at any time under the Ethernet communication (TCP).

3.1.6. bCap_GetRetry

Syntax

HRESULT bCap_GetRetry(int fd, unsigned int *retry);

Description

Obtain a send retry count of specified file descriptor..

Argument

[in] fd File descriptor of communication whose send retry count is obtained.

[out] retry Send retry count

Return value

S_OK(0) is returned if a send retry count I successful obtained. If not, a negative value is returned.

Note

None

4. Control flow of the b-CAP communication

This section describes the control flow of the b-CAP communication. Figure 4-1 shows the process flow between the server and client.

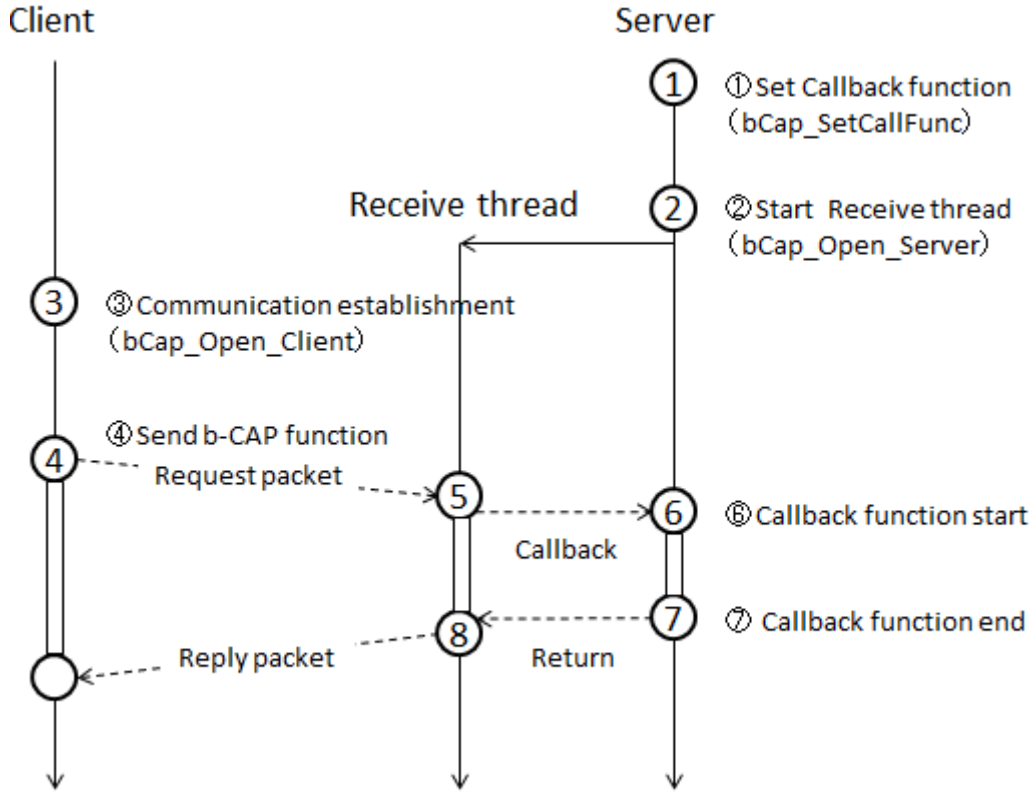

Figure 4−1 Process flow between the serve and client

The following explains each process.

① Set a callback function.

② Activate a receive thread. To activate miltiple receive threads, implement any exclusive control required for the server. If the communication method is Ethernet (TCP), please note that receive threads will be activated as much as the number of clients to be connected.

③ Establish a communication between the server and client.

④ Use a b-CAP function transmission interface that is provided by client.

⑤ The receive thread checks the b-CAP function ID of received packet. If the b-CAP function ID is the same as the one set in ①, execute ⑥. If the ID has not been set, the receive thread will automatically send the response packet of E_NOTIMPL(0x80004001).

⑥ According to the received b-CAP function ID, a callback function will be executed.

⑦ The process finishes when it reaches the return code. Set the execution result in vntRet ,if necessary.

⑧ The receive thread will send the return value of the callback function and execution result as a response packet.

Appendix A. b-CAP function ID and Callback function

The following table shows arguments for b-CAP function ID and callback function.

Table A-1 Arguments for b-CAP function ID and callback function

b-CAP function ID	vntArgs	vntRet
ID_SERVICE_START	[0] Option (VT_EMPTY or VT_BSTR)	None
ID_SERVICE_STOP	None	None
ID_CONTROLLER_CONNECT	[0] Name (VT_BSTR) [1] Provider (VT_BSTR) [2] Machine(VT_BSTR) [3] Option (VT_BSTR)	Handle number (VT_I4)
ID_CONTROLLER_DISCONNECT	[0] Handle number (VT_I4)	None
ID_CONTROLLER_GETEXTENSION	[0] Handle number (VT_I4) [1] Name (VT_BSTR) [2] Option (VT_BSTR)	Handle number (VT_I4)
ID_CONTROLLER_GETFILE	[0] Handle number (VT_I4) [1] Name (VT_BSTR) [2] Option (VT_BSTR)	Handle number (VT_I4)
ID_CONTROLLER_GETROBOT	[0] Handle number (VT_I4) [1] Name (VT_BSTR) [2] Option (VT_BSTR)	Handle number (VT_I4)
ID_CONTROLLER_GETTASK	[0] Handle number (VT_I4) [1] Name (VT_BSTR) [2] Option (VT_BSTR)	Handle number (VT_I4)
ID_CONTROLLER_GETVARIABLE	[0] Handle number (VT_I4) [1] Name (VT_BSTR) [2] Option (VT_BSTR)	Handle number (VT_I4)
ID_CONTROLLER_GETCOMMAND	[0] Handle number (VT_I4) [1] Name (VT_BSTR) [2] Option (VT_BSTR)	Handle number (VT_I4)
ID_CONTROLLER_GETEXTENSIONNAMES	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	Array of available Extension names (VT_BSTR | VT_ARRAY)
ID_CONTROLLER_GETFILENAMES	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	Array of available File names (VT_BSTR | VT_ARRAY)

ID_CONTROLLER_GETROBOTNAMES	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	Array of available Robot names (VT_BSTR | VT_ARRAY)
ID_CONTROLLER_GETTASKNAMES	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	Array of available Task names (VT_BSTR | VT_ARRAY)
ID_CONTROLLER_GETVARIABLENAMES	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	Array of available Variable names (VT_BSTR | VT_ARRAY)
ID_CONTROLLER_GETCOMMANDNAMES	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	Array of available Command names (VT_BSTR | VT_ARRAY)
ID_CONTROLLER_EXECUTE	[0] Handle number (VT_I4) [1] Command (VT_BSTR) [2] Option (VT_VARIANT)	Execution result (VT_VARIANT)
ID_CONTROLLER_GETMESSAGE	[0] Handle number (VT_I4)	Handle number (VT_I4)
ID_CONTROLLER_GETATTRIBUTE	[0] Handle number (VT_I4)	Attribute (VT_I4)
ID_CONTROLLER_GETHELP	[0] Handle number (VT_I4)	Help (VT_BSTR)
ID_CONTROLLER_GETNAME	[0] Handle number (VT_I4)	Name (VT_BSTR)
ID_CONTROLLER_GETTAG	[0] Handle number (VT_I4)	Tag (VT_VARIANT)
ID_CONTROLLER_PUTTAG	[0] Handle number (VT_I4) [1] Tag (VT_VARIANT)	None
ID_CONTROLLER_GETID	[0] Handle number (VT_I4)	ID(VT_VARIANT)
ID_CONTROLLER_PUTID	[0] Handle number (VT_I4) [1] ID(VT_VARIANT)	None
ID_EXTENSION_GETVARIABLE	[0] Handle number (VT_I4) [1] Name (VT_BSTR) [2] Option (VT_BSTR)	Handle number (VT_I4)
ID_EXTENSION_GETVARIABLENAMES	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	Array of available Variable names (VT_BSTR | VT_ARRAY)
ID_EXTENSION_EXECUTE	[0] Handle number (VT_I4) [1] Command (VT_BSTR) [2] Option (VT_VARIANT)	Execution result(VT_VARIANT)
ID_EXTENSION_GETATTRIBUTE	[0] Handle number (VT_I4)	Attribute (VT_I4)
ID_EXTENSION_GETHELP	[0] Handle number (VT_I4)	Help (VT_BSTR)
ID_EXTENSION_GETNAME	[0] Handle number (VT_I4)	Name (VT_BSTR)
ID_EXTENSION_GETTAG	[0] Handle number (VT_I4)	Tag (VT_VARIANT)
ID_EXTENSION_PUTTAG	[0] Handle number (VT_I4) [1] Tag (VT_VARIANT)	None

ID_EXTENSION_GETID	[0] Handle number (VT_I4)	ID(VT_VARIANT)
ID_EXTENSION_PUTID	[0] Handle number (VT_I4) [1] ID(VT_VARIANT)	None
ID_EXTENSION_RELEASE	[0] Handle number (VT_I4)	None
ID_FILE_GETFILE	[0] Handle number (VT_I4) [1] Name (VT_BSTR) [2] Option (VT_BSTR)	Handle number (VT_I4)
ID_FILE_GETVARIABLE	[0] Handle number (VT_I4) [1] Name (VT_BSTR) [2] Option (VT_BSTR)	Handle number (VT_I4)
ID_FILE_GETFILENAMES	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	Array of available File names (VT_BSTR | VT_ARRAY)
ID_FILE_GETVARIABLENAMES	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	Array of available Variable names (VT_BSTR | VT_ARRAY)
ID_FILE_EXECUTE	[0] Handle number (VT_I4) [1] Command (VT_BSTR) [2] Option (VT_VARIANT)	Execution result (VT_VARIANT)
ID_FILE_COPY	[0] Handle number (VT_I4) [1] Name (VT_BSTR) [2] Option (VT_BSTR)	None
ID_FILE_DELETE	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	None
ID_FILE_MOVE	[0] Handle number (VT_I4) [1] Name (VT_BSTR) [2] Option (VT_BSTR)	None
ID_FILE_RUN	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	File name after compile (VT_BSTR)
ID_FILE_GETDATECREATED	[0] Handle number (VT_I4)	File creation date (VT_VARIANT)
ID_FILE_GETDATELASTACCESSED	[0] Handle number (VT_I4)	Date of the last access to file (VT_VARIANT)
ID_FILE_GETDATELASTMODIFIED	[0] Handle number (VT_I4)	Date of the last file update (VT_VARIANT)
ID_FILE_GETPATH	[0] Handle number (VT_I4)	File path (VT_BSTR)
ID_FILE_GETSIZE	[0] Handle number (VT_I4)	File size (VT_I4)
ID_FILE_GETTYPE	[0] Handle number (VT_I4)	File type (VT_BSTR)

ID_FILE_GETVALUE	[0] Handle number (VT_I4)	Contents of file (VT_VARIANT)
ID_FILE_PUTVALUE	[0] Handle number (VT_I4) [1] File content (VT_VARIANT)	None
ID_FILE_GETATTRIBUTE	[0] Handle number (VT_I4)	Attribute (VT_I4)
ID_FILE_GETHELP	[0] Handle number (VT_I4)	Help (VT_BSTR)
ID_FILE_GETNAME	[0] Handle number (VT_I4)	Name (VT_BSTR)
ID_FILE_GETTAG	[0] Handle number (VT_I4)	Tag (VT_VARIANT)
ID_FILE_PUTTAG	[0] Handle number (VT_I4) [1] Tag (VT_VARIANT)	None
ID_FILE_GETID	[0] Handle number (VT_I4)	ID(VT_VARIANT)
ID_FILE_PUTID	[0] Handle number (VT_I4) [1] ID(VT_VARIANT)	None
ID_FILE_RELEASE	[0] Handle number (VT_I4)	None
ID_ROBOT_GETVARIABLE	[0] Handle number (VT_I4) [1] Name (VT_BSTR) [2] Option (VT_BSTR)	Handle number (VT_I4)
ID_ROBOT_GETVARIABLENAMES	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	Array of available Variable names (VT_BSTR | VT_ARRAY)
ID_ROBOT_EXECUTE	[0] Handle number (VT_I4) [1] Command (VT_BSTR) [2] Option (VT_VARIANT)	Execution result (VT_VARIANT)
ID_ROBOT_ACCELERATE	[0] Handle number (VT_I4) [1] Axis number(VT_I4) [2] Acceleration (VT_R4) [3] Deceleration (VT_R4)	None
ID_ROBOT_CHANGE	[0] Handle number (VT_I4) [1] Name (VT_BSTR)	None
ID_ROBOT_CHUCK	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	None
ID_ROBOT_DRIVE	[0] Handle number (VT_I4) [1] Axis number(VT_I4) [2] Distance (VT_R4) [3] Option (VT_BSTR)	None
ID_ROBOT_GOHOME	[0] Handle number (VT_I4)	None
ID_ROBOT_HALT	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	None

ID_ROBOT_HOLD	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	None
ID_ROBOT_MOVE	[0] Handle number (VT_I4) [1] Interpolation specification (VT_I4) [2] Posed row(VT_VARIANT) [3] Option (VT_BSTR)	None
ID_ROBOT_ROTATE	[0] Handle number (VT_I4) [1] Rotation surface (VT_VARIANT) [2] Angle (VT_R4) [3] Rotation center (VT_VARIANT) [4] Option (VT_BSTR)	None
ID_ROBOT_SPEED	[0] Handle number (VT_I4) [1] Axis number(VT_I4) [2] Speed (VT_R4)	None
ID_ROBOT_UNCHUCK	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	None
ID_ROBOT_UNHOLD	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	None
ID_ROBOT_GETATTRIBUTE	[0] Handle number (VT_I4)	Attribute (VT_I4)
ID_ROBOT_GETHELP	[0] Handle number (VT_I4)	Help (VT_BSTR)
ID_ROBOT_GETNAME	[0] Handle number (VT_I4)	Name (VT_BSTR)
ID_ROBOT_GETTAG	[0] Handle number (VT_I4)	Tag (VT_VARIANT)
ID_ROBOT_PUTTAG	[0] Handle number (VT_I4) [1] Tag (VT_VARIANT)	None
ID_ROBOT_GETID	[0] Handle number (VT_I4)	ID(VT_VARIANT)
ID_ROBOT_PUTID	[0] Handle number (VT_I4) [1] ID(VT_VARIANT)	None
ID_ROBOT_RELEASE	[0] Handle number (VT_I4)	None
ID_TASK_GETVARIABLE	[0] Handle number (VT_I4) [1] Name (VT_BSTR) [2] Option (VT_BSTR)	Handle number (VT_I4)
ID_TASK_GETVARIABLENAMES	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	Array of available Variable names (VT_BSTR | VT_ARRAY)
ID_TASK_EXECUTE	[0] Handle number (VT_I4) [1] Command (VT_BSTR) [2] Option (VT_VARIANT)	Execution result (VT_VARIANT)

ID_TASK_START	[0] Handle number (VT_I4) [1] Start mode (VT_I4) [2] Option (VT_BSTR)	None
ID_TASK_STOP	[0] Handle number (VT_I4) [1] Stop mode (VT_I4) [2] Option (VT_BSTR)	None
ID_TASK_DELETE	[0] Handle number (VT_I4) [1] Option (VT_BSTR)	None
ID_TASK_GETFILENAME	[0] Handle number (VT_I4)	File name (VT_BSTR)
ID_TASK_GETATTRIBUTE	[0] Handle number (VT_I4)	Attribute (VT_I4)
ID_TASK_GETHELP	[0] Handle number (VT_I4)	Help (VT_BSTR)
ID_TASK_GETNAME	[0] Handle number (VT_I4)	Name (VT_BSTR)
ID_TASK_GETTAG	[0] Handle number (VT_I4)	Tag (VT_VARIANT)
ID_TASK_PUTTAG	[0] Handle number (VT_I4) [1] Tag (VT_VARIANT)	None
ID_TASK_GETID	[0] Handle number (VT_I4)	ID(VT_VARIANT)
ID_TASK_PUTID	[0] Handle number (VT_I4) [1] ID(VT_VARIANT)	None
ID_TASK_RELEASE	[0] Handle number (VT_I4)	None
ID_VARIABLE_GETDATETIME	[0] Handle number (VT_I4)	Current time (VT_VARIANT)
ID_VARIABLE_GETVALUE	[0] Handle number (VT_I4)	Variable value (VT_VARIANT)
ID_VARIABLE_PUTVALUE	[0] Handle number (VT_I4) [1] Variable value (VT_VARIANT)	None
ID_VARIABLE_GETATTRIBUTE	[0] Handle number (VT_I4)	Attribute (VT_I4)
ID_VARIABLE_GETHELP	[0] Handle number (VT_I4)	Help (VT_BSTR)
ID_VARIABLE_GETNAME	[0] Handle number (VT_I4)	Name (VT_BSTR)
ID_VARIABLE_GETTAG	[0] Handle number (VT_I4)	Tag (VT_VARIANT)
ID_VARIABLE_PUTTAG	[0] Handle number (VT_I4) [1] Tag (VT_VARIANT)	None
ID_VARIABLE_GETID	[0] Handle number (VT_I4)	ID (VT_VARIANT)
ID_VARIABLE_PUTID	[0] Handle number (VT_I4) [1] ID(VT_VARIANT)	None
ID_VARIABLE_GETMICROSECOND	[0] Handle number (VT_I4)	Time stamp (VT_I4)
ID_VARIABLE_RELEASE	[0] Handle number (VT_I4)	None
ID_COMMAND_EXECUTE	[0] Handle number (VT_I4) [1] Mode (VT_I4)	Execution result (VT_VARIANT)

ID_COMMAND_CANCEL	[0] Handle number (VT_I4)	None
ID_COMMAND_GETTIMEOUT	[0] Handle number (VT_I4)	Timeout time (VT_I4)
ID_COMMAND_PUTTIMEOUT	[0] Handle number (VT_I4) [1] Timeout time (VT_I4)	None
ID_COMMAND_GETSTATE	[0] Handle number (VT_I4)	State (VT_I4)
ID_COMMAND_GETPARAMETERS	[0] Handle number (VT_I4)	Command parameter (VT_VARIANT)
ID_COMMAND_PUTPARAMETERS	[0] Handle number (VT_I4) [1] Command parameter (VT_VARIANT)	None
ID_COMMAND_GETRESULT	[0] Handle number (VT_I4)	Execution result (VT_VARIANT)
ID_COMMAND_GETATTRIBUTE	[0] Handle number (VT_I4)	Attribute (VT_I4)
ID_COMMAND_GETHELP	[0] Handle number (VT_I4)	Help (VT_BSTR)
ID_COMMAND_GETNAME	[0] Handle number (VT_I4)	Name (VT_BSTR)
ID_COMMAND_GETTAG	[0] Handle number (VT_I4)	Tag (VT_VARIANT)
ID_COMMAND_PUTTAG	[0] Handle number (VT_I4) [1] Tag (VT_VARIANT)	None
ID_COMMAND_GETID	[0] Handle number (VT_I4)	ID(VT_VARIANT)
ID_COMMAND_PUTID	[0] Handle number (VT_I4) [1] ID(VT_VARIANT)	None
ID_COMMAND_RELEASE	[0] Handle number (VT_I4)	None
ID_MESSAGE_REPLY	[0] Handle number (VT_I4) [1] Reply message (VT_VARIANT)	None
ID_MESSAGE_CLEAR	[0] Handle number (VT_I4)	None
ID_MESSAGE_GETDATETIME	[0] Handle number (VT_I4)	Message creation date (VT_VARIANT)
ID_MESSAGE_GETDESCRIPTION	[0] Handle number (VT_I4)	Message description(VT_BSTR)
ID_MESSAGE_GETDESTINATION	[0] Handle number (VT_I4)	Message destination (VT_BSTR)
ID_MESSAGE_GETNUMBER	[0] Handle number (VT_I4)	Message number (VT_I4)
ID_MESSAGE_GETSERIALNUMBER	[0] Handle number (VT_I4)	Message serial number (VT_I4)
ID_MESSAGE_GETSOURCE	[0] Handle number (VT_I4)	Message source (VT_BSTR)
ID_MESSAGE_GETVALUE	[0] Handle number (VT_I4)	Message content (VT_VARIANT)
ID_MESSAGE_RELEASE	[0] Handle number (VT_I4)	None

Appendix A.1. Handle number

This section describes a handle number that is used for arguments of callback function and execution results. Figure A-1-1 shows the flow of the handle number creation.

A function that provides a handle number to an execution result (e.g; ID_CONTROLLER_CONNECT, ID_CONTROLLER_GETVARIABLE) reserves the memory area when a callback function is executed.

The server must retain the memory until a memory release function (e.g; ID_CONTROLLER_DISCONNEC, ID_VARIABLE_RELEASE) is called.

To control these memory, create a memory allocation table. At the same time, create a key that is unique to a memory in order to call each memory from the table. The key will be informed to clients as a handle.

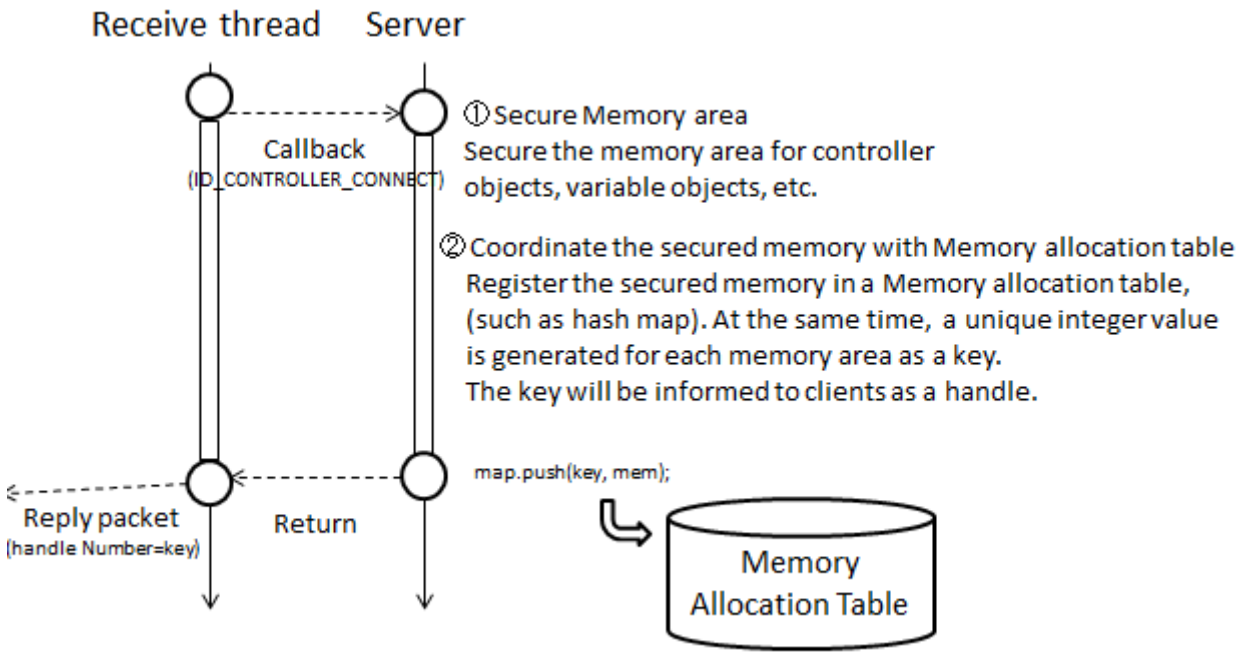

Figure A-1-1 Creating a handle number

When a function that includes a handle number (e.g: ID_CONTROLLER_EXECUTE) is executed as an argument, obtain a memory from the memory allocation table based on the handle number.

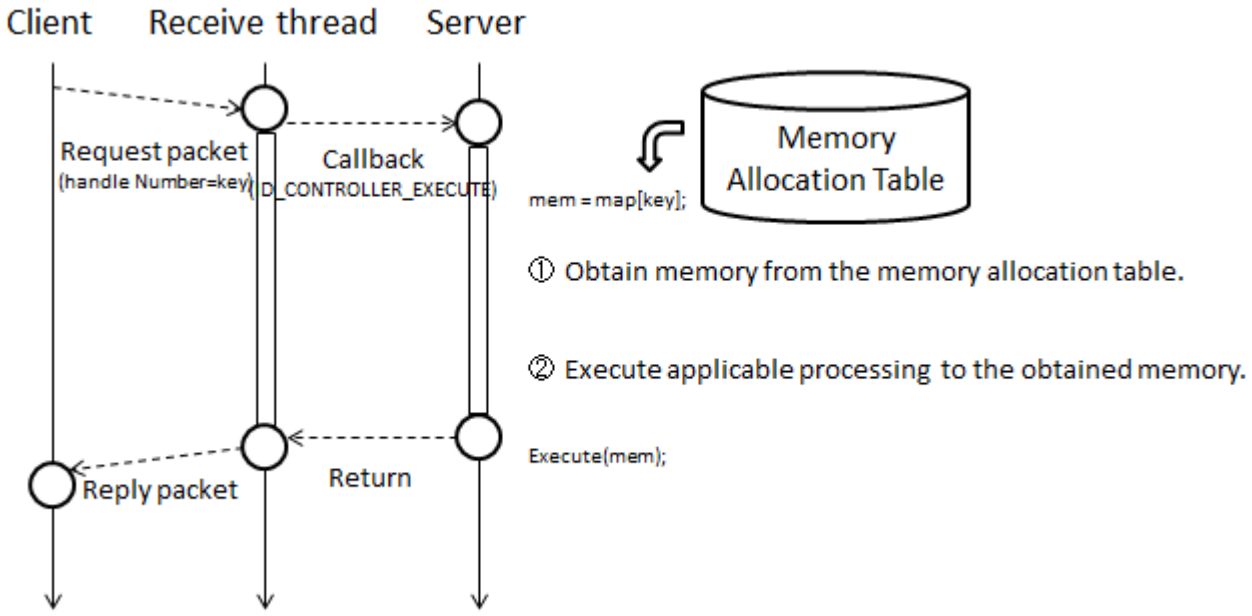

Figure A-1-2 Using a handle number