e-CAP specifications

Version 1.0

October 16, 2008

【Remarks】
This document was automatically translated from the Japanese document by the machine translation system, and no one has been modified manually. This translation version is offered for the customer who uses English, and DENSO WAVE doesn't guarantee the quality of this version at all. Moreover DENSO WAVE doesn't assume the responsibility of all problems caused by the mistranslation of this document.

【Revision History】

Date	versions	Content
2008-10-16	1.0.0	First edition

Contents

1 Introduction

This specifications provide for the communication protocol of e-CAP(Embedded CAP).

There is CAP(Controller Access Protocol) as a means to send and receive the command remotely as well as this e-CAP. However, it takes time to mounting when the server side is not Windows machine because it adopts SOAP in CAP.

To become simpler than CAP, e-CAP is designed. As a result, e-CAP has the following features.

Simplification only by HTTP without using XML.

A complex fact sheet reality (multi-dimensional structure array etc.) is not supported.

Other functions are equal to CAP.

2 Structure of e-CAP

e-CAP has the service structure similar to the object model of the CAO provider, and one e-CAP message corresponds to one service (function). This structure is shown in figure below.

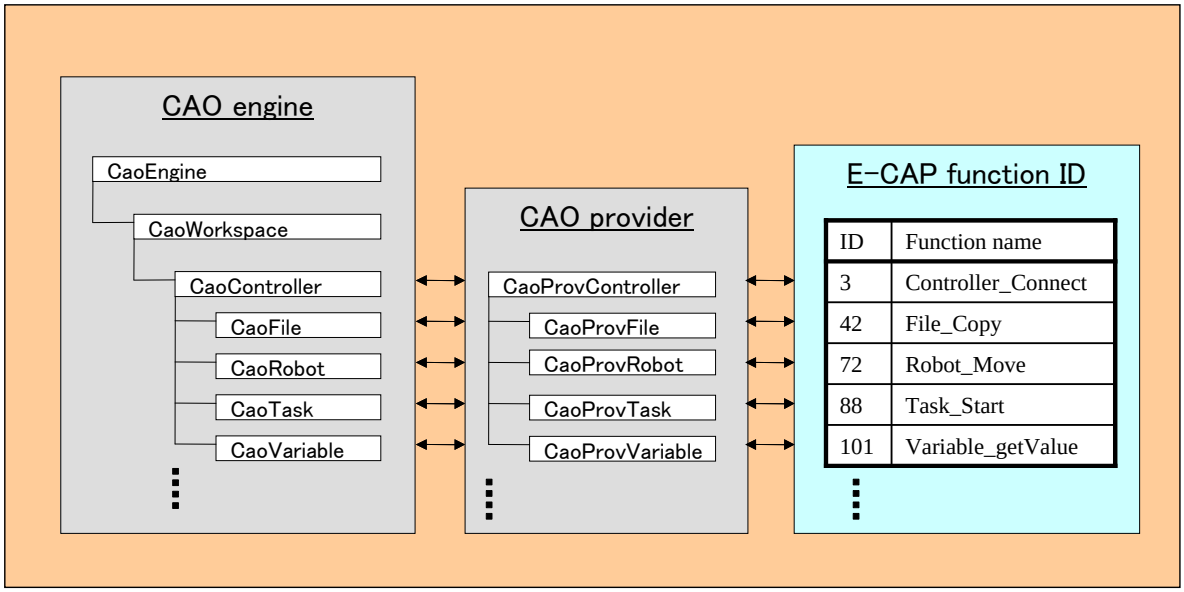

Figure2-1 Structure of e-CAP

To reduce the load of the analysis of the command on the server side by expressing the function that uses HTTP and uses like CGI, and to simplify mounting compared with CAP, e-CAP is designed.

E-CAP executes the e-CAP client and service that demands service and is composed of two programs of the e-CAP server that returns the result.

The e-CAP client makes the HTTP request message to store information necessary for the demanded service, transmits to the server side, receives the HTTP response message, and confirms the execution result.

In the e-CAP server, the HTTP request message from the client is received, and service corresponding to function ID is executed. After service is executed, the result and the value are stored in the HTTP response message, and it transmits to the client side.

The connection example by b-CAP is shown as follows.

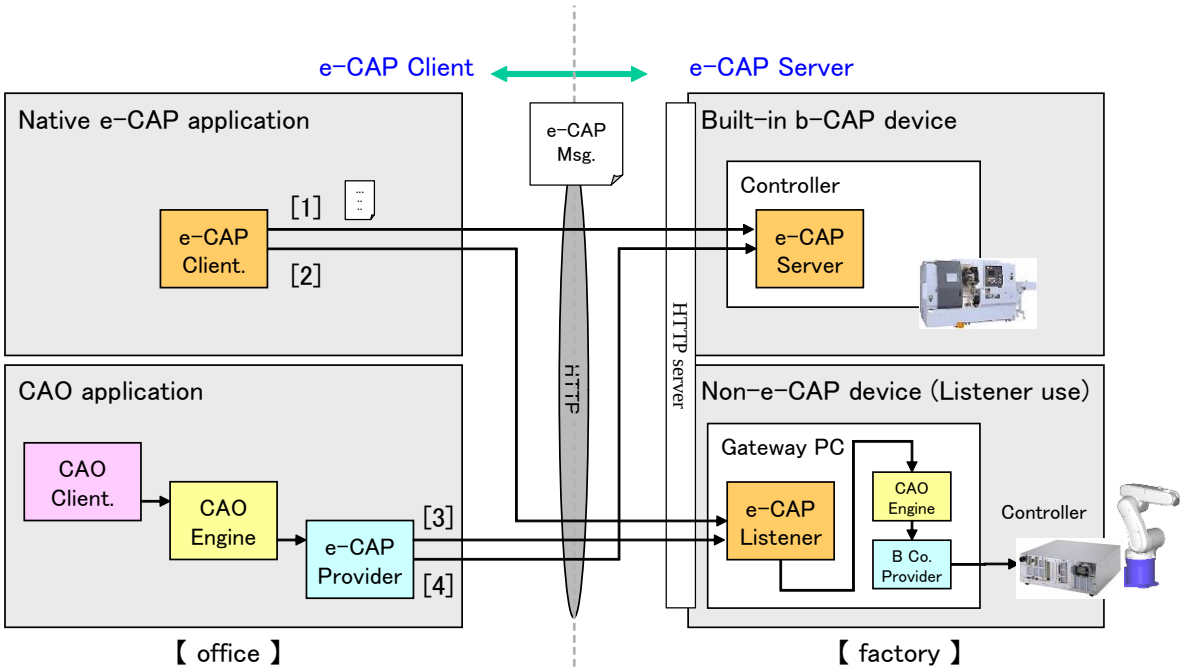

Figure2-2 Example of connecting e-CAP

3 E-CAP message

3.1. Message structure

E-CAP does the sending and receiving message by using HTTP like CGI. At this time, the mark rule of the e-CAP message follows it at the following.

3.2. Message rule

E-CAP does the sending and receiving message by using HTTP like CGI. At this time, the mark rule of the e-CAP message follows it at the following.

Please transmit the “POST" command to the HTTP request message.

Please add the following meta data to HTTP header of the HTTP request message.

“Content-Type:application/x-www-form-urlencoded"

The HTTP body of the HTTP request message becomes the following structures.

Arg5=< the fifth argument Arg4=< the fourth argument > Arg3=< the third argument > Arg2=< the second argument > Arg1=< the first ..Func=< function ID >.. argument >>

Function ID	Function ID . About details3.3. Please refer to [wo].
The 1-5th argument	Parameter of execution function. When the type of the argument is VARIANT typeVARIANT data formatIt is necessary to describe it according to [sho] type drinking. The input value is set by the character string, except for the VARIANT type. Please twine and set [bun] [azaretsu] to the argument that the execution function doesn't use.

The HTTP body of the HTTP response message becomes the following structures.

< return code > and < result data >

Return code	Return code of execution function. About detailsReturn codePlease refer to [wo].
Result data	Result of execution function data When the execution function doesn't return data, the result data is omitted. The result data is acquired in the VARIANT type. About the format of the VARIANT typeVARIANT data formatPlease refer to [wo].

One communication example is shown as follows.

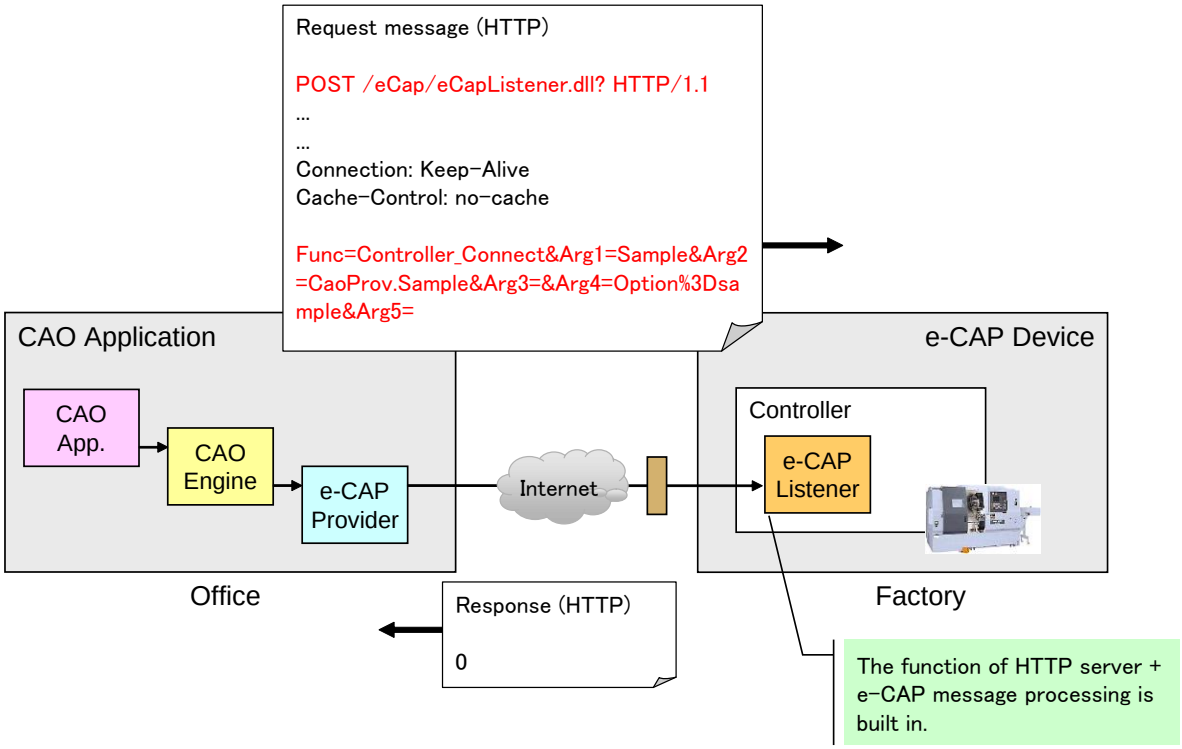

Figure3-1 Example of e-CAP message

3.3. Function ID

Function ID is allotted in b-CAP as follows.

Table1 Function ID allocation

Function ID	Explanation
1～137	Predetermined function
138～255	Reservation area
255～	User function

It is possible to use it by allocating an arbitrary function in the user function to use functions other than a predetermined function.

The list of a predetermined function of b-CAP is shown below.

Table2 Predetermined function list

Function ID	Function name	Explanation

1	Service_Start	Beginning of server service
2	Service_Stop	Stop of server service
3	Controller_Connect	Connection with controller
4	Controller_Disconnect	Cutting with controller
5	Controller_GetExtension	The controller's extension board acquisition
6	Controller_GetFile	The controller's file acquisition
7	Controller_GetRobot	The controller's robot acquisition
8	Controller_GetTask	The controller's task acquisition
9	Controller_GetVariable	The controller's variable acquisition
10	Controller_GetCommand	The controller's command acquisition
11	Controller_GetExtensionNames	The controller's extension board name list acquisition
12	Controller_GetFileNames	The controller's file name list acquisition
13	Controller_GetRobotNames	The controller's robot name list acquisition
14	Controller_GetTaskNames	The controller's task name list acquisition
15	Controller_GetVariableNames	The controller's variable identifier list acquisition
16	Controller_GetCommandNames	The controller's command name list acquisition
17	Controller_Execute	Execution of controller's enhancing function
18	Controller_GetMessage	The controller's event message acquisition
19	Controller_GetAttribute	The controller's attribute value acquisition
20	Controller_GetHelp	The controller's help character string acquisition
21	Controller_GetName	The controller's name acquisition
22	Controller_GetTag	The controller's tag information acquisition
23	Controller_PutTag	The controller's tag information setting
24	Controller_GetID	The controller's ID acquisition
25	Controller_PutID	The controller's ID setting
26	Extension_GetVariable	Acquisition of variable of extension board
27	Extension_GetVariableNames	Acquisition of list of variable identifier of extension board
28	Extension_Execute	Execution of enhancing function of extension board
29	Extension_GetAttribute	Attribute value acquisition of extension board
30	Extension_GetHelp	Acquisition of help character string of extension board
31	Extension_GetName	Acquisition of name of extension board
32	Extension_GetTag	Acquisition of tag information on extension board
33	Extension_PutTag	Setting of tag information on extension board
34	Extension_GetID	ID acquisition of extension board

35	Extension_PutID	ID setting of extension board
36	Extension_Release	Liberating of extension board
37	File_GetFile	Another file acquisition of file
38	File_GetVariable	Acquisition of variable of file
39	File_GetFileNames	Acquisition of list of another file name of file
40	File_GetVariableNames	Acquisition of list of variable identifier of file
41	File_Execute	Execution of enhancing function of file
42	File_Copy	Copy of file
43	File_Delete	Deletion of file
44	File_Move	Movement of file
45	File_Run	Execution of file
46	File_GetDateCreated	Acquisition at the date of file
47	File_GetDateLastAccessed	Acquisition at the final access date of file
48	File_GetDateLastModified	Acquisition at last updated date and time of file
49	File_GetPath	Passing acquisition of file
50	File_GetSize	Size acquisition of file
51	File_GetType	File type acquisition of file
52	File_GetValue	Acquisition of content of file
53	File_PutValue	Setting of content of file
54	File_GetAttribute	Attribute acquisition of file
55	File_GetHelp	Acquisition of help character string of file
56	File_GetName	Acquisition of name of file
57	File_GetTag	Acquisition of tag information on file
58	File_PutTag	Setting of tag information on file
59	File_GetID	ID acquisition of file
60	File_PutID	ID setting of file
61	File_Release	Liberating of file
62	Robot_GetVariable	Acquisition of variable of robot
63	Robot_GetVariableNames	Acquisition of list of variable identifier of robot
64	Robot_Execute	Execution of enhancing function of robot
65	Robot_Accelerate	Execution of ACCEL sentence of robot
66	Robot_Change	Execution of CHANGE sentence of robot
67	Robot_Chuck	Execution of GRASP sentence of robot
68	Robot_Drive	Execution of DRIVE sentence of robot
69	Robot_GoHome	Execution of GOHOME sentence of robot

70	Robot_Halt	Execution of HALT sentence of robot
71	Robot_Hold	Execution of HOLD sentence of robot
72	Robot_Move	Execution of MOVE sentence of robot
73	Robot_Rotate	Execution of ROTATE sentence of robot
74	Robot_Speed	Execution of SPEED/JSPEED sentence of robot
75	Robot_Unchuck	Execution of REELASE sentence of robot
76	Robot_Unhold	Release of HOLD sentence of robot
77	Robot_GetAttribute	Attribute value acquisition of robot
78	Robot_GetHelp	Acquisition of help character string of robot
79	Robot_GetName	Acquisition of name of robot
80	Robot_GetTag	Acquisition of tag information on robot
81	Robot_PutTag	Setting of tag information on robot
82	Robot_GetID	ID acquisition of robot
83	Robot_PutID	ID setting of robot
84	Robot_Release	Liberating of robot
85	Task_GetVariable	Acquisition of variable of task
86	Task_GetVariableNames	Acquisition of list of variable identifier of task
87	Task_Execute	Execution of enhancing function of task
88	Task_Start	Beginning of task
89	Task_Stop	Stop of task
90	Task_Delete	Deletion of task
91	Task_GetFileName	Former file name of task
92	Task_GetAttribute	Attribute acquisition of task
93	Task_GetHelp	Acquisition of help character string of task
94	Task_GetName	Acquisition of name of task
95	Task_GetTag	Acquisition of tag information on task
96	Task_PutTag	Setting of tag information on task
97	Task_GetID	ID acquisition of task
98	Task_PutID	ID setting of task
99	Task_Release	Liberating of task
100	Variable_GetDateTime	Stamp acquisition of time of variable
101	Variable_GetValue	Value acquisition of variable
102	Variable_PutValue	Value setting of variable
103	Variable_GetAttribute	Attribute value acquisition of variable
104	Variable_GetHelp	Acquisition of help character string of variable

105	Variable_GetName	Acquisition of name of variable
106	Variable_GetTag	Acquisition of tag information on variable
107	Variable_PutTag	Setting of tag information on variable
108	Variable_GetID	ID acquisition of variable
109	Variable_PutID	ID setting of variable
110	Variable_GetMicrosecond	Time stamp (millisecond) acquisition of variable
111	Variable_Release	Liberating of variable
112	Command_Execute	Execution of command
113	Command_Cancel	Cancellation of command
114	Command_GetTimeout	Acquisition at time-out time of command
115	Command_PutTimeout	Setting at time-out time of command
116	Command_GetState	State acquisition of command
117	Command_GetParameters	Acquisition of parameter of command
118	Command_PutParameters	Setting of parameter of command
119	Command_GetResult	Execution result acquisition of command
120	Command_GetAttribute	Attribute value acquisition of command
121	Command_GetHelp	Acquisition of help character string of command
122	Command_GetName	Acquisition of name of command
123	Command_GetTag	Acquisition of tag information on command
124	Command_PutTag	Setting of tag information on command
125	Command_GetID	ID acquisition of command
126	Command_PutID	ID setting of command
127	Command_Release	Liberating of command
128	Message_Reply	Response of event message
129	Message_Clear	Clearness of event message
130	Message_GetDateTime	Stamp acquisition of time of event message
131	Message_GetDescription	Acquisition of explanation of event message
132	Message_GetDestination	Destination acquisition of event message
133	Message_GetNumber	Acquisition of message number of event message
134	Message_GetSerialNumber	Acquisition of serial number of event message
135	Message_GetSource	Former transmission acquisition of event message
136	Message_GetValue	Value acquisition of event message
137	Message_Release	Liberating of event message

3.4. Return code

The return code is allotted in b-CAP as follows.

Table3 Allocation of return code

Return code	Explanation
0x00000000～0x8000FFFF	Predetermined return code and reservation area
0x80010000～0x8001FFFF	User definition error

When the error codes other than the following "Predetermined error code" are made, an arbitrary error code can be allocated within the range of the value of "User definition error".

Table4 Predetermined return code list

Return code	Error	Explanation
0x00000000	S_OK	Normal termination.
0x80004001	E_NOTIMPL	Unmounting.
0x80004004	E_ABORT	The function was interrupted.
0x80004005	E_FAIL	The function failed.
0x80070005	E_ACCESSDENIED	It is not possible to access it.
0x80070006	E_HANDLE	The steering wheel is illegal.
0x8007000E	E_OUTOFMEMORY	The memory is insufficient.
0x80070057	E_INVALIDARG	The argument is illegal.
0x8000FFFF	E_UNEXPECTED	A fatal error occurred.

3.5. VARIANT data format

The data format of the VARIANT type in e-CAP follows the method of expressing the VARIANT type of ORiN2 by the character string. The format expresses the data type and the data string by switching off the comma district as follows.

< data string >< data type >

Here, the integral value written by the VARTYPE type is shown in < data type >. Table3-5The data type for which [ni] can be used and the value are indicated.

Table3-5 Data type that can be used

Data type	Value	Meaning
VT_I2	2	Two byte integer type
VT_I4	3	Four byte integer type

VT_R4	4	Floating point of single precision type
VT_R8	5	Floating point of double precision type
VT_CY	6	Currency type
VT_DATE	7	Date type
VT_BSTR	8	Character string type
VT_BOOL	11	Boolean type
VT_VARIANT	12	VARIANT type
VT_UI1	17	Byte type
VT_ARRAY	8192	Array type

When the data type arranges it, it writes by VT_ARRAY and the logical add of the data type.

Data is written in the data string by the character string. The mark of the sequence data is delimited by "," (comma) and written.

EX1)	2,100	Type:VT_I2	Value: 100
EX2)	8,Sample	Type:VT_BSTR	Value:Sample
EX3)	12,(8,Sample)	Type:VT_VARIANT	Value:"Sample" (character string type)
EX4)	8194,100,200,300	Type:VT_I2|VTARRAY	Value: 100,200,300
EX5)	8200,Sample,Test	Type:VT_BSTR|VTARRAY	Value:"Sample","Test"
EX6)	8,Sample,Test	Type:VT_BSTR	Value:"Sample,Test"
EX7)	8204,(8,Sample),(2,100)	Type:VT_VARIANT|VTARRAY	Value:"Sample",100